Package 'ggformula'

October 6, 2025

Title Formula Interface to the Grammar of Graphics

Description Provides a formula interface to 'ggplot2' graphics.

Type Package

Version 1.0.0

License MIT + file LICENSE

LazyData TRUE

LazyLoad TRUE

Depends R (>= 4.1), ggplot2 (>= 3.3), scales, ggiraph, ggridges

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

Suggests tidyr, patchwork, glue, mosaicData, dplyr, lattice, mosaic, palmerpenguins, testthat, vdiffr, knitr, rmarkdown, lubridate, survival, broom, hexbin, maps, sf, purrr, ggthemes, covr, ggplot2movies, interp, quantreg, ggforce, quarto

VignetteBuilder quarto

RoxygenNote 7.3.3

Encoding UTF-8

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

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

Collate 'MIpop-doc.R' 'formula2aes.R' 'gf_aux.R' 'gf_dist.R' 'layer_factory.R' 'gf_function2d.R' 'gf_functions.R' 'gf_plot.R' 'ggridges.R' 'ggstrings.R' 'newplots.R' 'ggiraph.R' 'ggiraph-themes.R' 'ggiraph-documentation-with-examples.R' 'reexports.R' 'scales.R' 'utils.R' 'relabel.R' 'vdiffr.R' 'ggstance-deprecated.R' 'ggformula-package.R' 'zzz.R'

NeedsCompilation no

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

Maintainer Randall Pruim <rpruim@calvin.edu>

Repository CRAN

Date/Publication 2025-10-06 05:10:25 UTC

Contents

`discrete_breaks` *Discrete Breaks*

Description

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

Usage

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

Arguments

`resolution`	Resolution of the breaks

Value

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

Examples

```
x <- rbinom(100, 100, 0.4)
p <- gf_bar( ~ x)
p |> gf_refine(scale_x_continuous(breaks = discrete_breaks()))
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',
      birthyear   = '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 i 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

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

gf_hline(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...,
  yintercept,
  color,
  linetype,
  linewidth,
  alpha,
  xlab,
  ylab,
  title,
  subtitle,
```

```
  caption,
  position = "identity",
  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,
  position = "identity",
  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).

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().

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. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.

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.

position — A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following:

- The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position.
- A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter".
- For more information and other ways to specify the position, see the layer position documentation.

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(
    color = "red", slope = ~slope, intercept = ~intercept,
    data = data.frame(slope = -0.10, intercept = 33:35)
  )

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

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

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

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

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

---

gf_abline_interactive *Interactive reference lines*

Description

Creates an interactive plot using ggiraph. This function extends gf_abline() with interactive features like tooltips and clickable elements.

Arguments

object	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument.
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.
tooltip	A formula specifying a variable for tooltips, or a character vector.
data_id	A formula or character vector specifying data identifiers for interactive selection.
...	Additional arguments passed to the underlying geom.
alpha, color, size, shape, fill, group, stroke	Aesthetics passed to the geom.
xlab, ylab, title, subtitle, caption	Labels for the plot.
show.legend	Logical. Should this layer be included in the legends?
show.help	Logical. If TRUE, display some minimal help.
inherit	Logical. If TRUE, inherit aesthetics from previous layers.
environment	An environment in which to evaluate the formula.

Value

A gg object that can be displayed with gf_girafe().

Additional interactive features

- onclick: JavaScript code (as character string) executed when clicking elements.
- Additional ggiraph aesthetics may be available depending on the geom.

See Also

gf_abline(), gf_girafe()

Examples

```
mtcars |>
  gf_point(mpg ~ wt) |>
  gf_abline_interactive(
    slope = ~ -2,
    intercept = ~ 35,
    tooltip = ~ "slope: -2; intercept: 35",
  ) |>
  gf_girafe()
```

`gf_annotate` *Add an annotation layer to a graphic*

Description

This function adds geoms to a plot, but unlike a typical geom function, the properties of the geoms are not mapped from variables of a data frame, but are instead passed in as vectors. This is useful for adding small annotations (such as text labels) or if you have your data in vectors, and for some reason don't want to put them in a data frame.

Usage

```
gf_annotate(
  object,
  geom = "text",
  x = NULL,
  y = NULL,
  xmin = NULL,
  xmax = NULL,
  ymin = NULL,
  ymax = NULL,
  xend = NULL,
  yend = NULL,
  ...,
  na.rm = FALSE
)
```

Arguments

`object`	a gg object
`geom`	name of geom to use for annotation
`x`, `y`, `xmin`, `ymin`, `xmax`, `ymax`, `xend`, `yend`	positioning aesthetics - you must specify at least one of these.
`...`	Other arguments passed on to `layer()`'s `params` argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the `position` argument, or aesthetics that are required can *not* be passed through `....` Unknown arguments that are not part of the 4 categories below are ignored.

- Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, `colour = "red"` or `linewidth = 3`. The geom's documentation has an **Aesthetics** section that lists the available options. The 'required' aesthetics cannot be passed on to the `params`. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data.

- When constructing a layer using a `stat_*()` function, the `...` argument can be used to pass on parameters to the geom part of the layer. An example of this is `stat_density(geom = "area", outline.type = "both")`. The geom's documentation lists which parameters it can accept.
- Inversely, when constructing a layer using a `geom_*()` function, the `...` argument can be used to pass on parameters to the `stat` part of the layer. An example of this is `geom_area(stat = "density", adjust = 0.5)`. The stat's documentation lists which parameters it can accept.
- The `key_glyph` argument of `layer()` may also be passed on through `....` This can be one of the functions described as key glyphs, to change the display of the layer in the legend.

`na.rm` If `FALSE`, the default, missing values are removed with a warning. If `TRUE`, missing values are silently removed.

Details

Note that all position aesthetics are scaled (i.e. they will expand the limits of the plot so they are visible), but all other aesthetics are set. This means that layers created with this function will never affect the legend.

Unsupported geoms

Due to their special nature, reference line geoms `geom_abline()`, `geom_hline()`, and `geom_vline()` can't be used with `annotate()`. You can use these geoms directly for annotations.

See Also

The custom annotations section of the online ggplot2 book.

Examples

```
p <- gf_point(mpg ~wt, data = mtcars)
p |> gf_annotate("text", x = 4, y = 25, label = "Some text")
p |> gf_annotate("text", x = 2:5, y = 25, label = "Some text")
p |> gf_annotate("rect", xmin = 3, xmax = 4.2, ymin = 12, ymax = 21,
             alpha = .2)
p |> gf_annotate("segment", x = 2.5, xend = 4, y = 15, yend = 25,
             colour = "blue")
p |> gf_annotate("pointrange", x = 3.5, y = 20, ymin = 12, ymax = 28,
             colour = "red", size = 2.5, linewidth = 1.5)

p |> gf_annotate("text", x = 2:3, y = 20:21, label = c("my label", "label 2"))

p |> gf_annotate("text", x = 4, y = 25, label = "italic(R) ^ 2 == 0.75",
             parse = TRUE)
p |> gf_annotate("text", x = 4, y = 25,
             label = "paste(italic(R) ^ 2, \" = .75\")", parse = TRUE)
```

gf_area *Formula interface to geom_area()*

Description

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

Usage

```
gf_area(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...,
  alpha,
  color,
  fill,
  group,
  linetype,
  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()`.

`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 for this layer. When using a `stat_*()` function to construct a layer, the geom argument can be used to override the default coupling between stats and geoms. The geom argument accepts the following:

- A `Geom` ggproto subclass, for example `GeomPoint`.
- A string naming the geom. To give the geom as a string, strip the function name of the `geom_` prefix. For example, to use `geom_point()`, give the geom as `"point"`.
- For more information and other ways to specify the geom, see the layer geom documentation.

`stat` The statistical transformation to use on the data for this layer. When using a `geom_*()` function to construct a layer, the `stat` argument can be used to override the default coupling between geoms and stats. The `stat` argument accepts the following:

- A `Stat` ggproto subclass, for example `StatCount`.
- A string naming the stat. To give the stat as a string, strip the function name of the `stat_` prefix. For example, to use `stat_count()`, give the stat as `"count"`.
- For more information and other ways to specify the stat, see the layer stat documentation.

`position` A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The `position` argument accepts the following:

- The result of calling a position function, such as `position_jitter()`. This method allows for passing extra arguments to the position.

- A string naming the position adjustment. To give the position as a string, strip the function name of the `position_` prefix. For example, to use `position_jitter()`, give the position as `"jitter"`.
- For more information and other ways to specify the position, see the layer position documentation.

`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. To include legend keys for all levels, even when no data exists, use `TRUE`. If `NA`, all levels are shown in legend, but unobserved levels are omitted.

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

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

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

See Also

`ggplot2::geom_area()`

Examples

```
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_area_interactive *Interactive area plots*

Description

Creates an interactive plot using ggiraph. This function extends `gf_area()` with interactive features like tooltips and clickable elements.

Arguments

object	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument.
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.
tooltip	A formula specifying a variable for tooltips, or a character vector.
data_id	A formula or character vector specifying data identifiers for interactive selection.
...	Additional arguments passed to the underlying geom.
alpha, color, size, shape, fill, group, stroke	Aesthetics passed to the geom.
xlab, ylab, title, subtitle, caption	Labels for the plot.
show.legend	Logical. Should this layer be included in the legends?
show.help	Logical. If TRUE, display some minimal help.
inherit	Logical. If TRUE, inherit aesthetics from previous layers.
environment	An environment in which to evaluate the formula.

Value

A gg object that can be displayed with gf_girafe().

Additional interactive features

- onclick: JavaScript code (as character string) executed when clicking elements.
- Additional ggiraph aesthetics may be available depending on the geom.

See Also

gf_area(), gf_girafe()

Examples

```
Huron <-
  data.frame(
    year = 1875:1972,
    level = as.vector(LakeHuron)
  )

Huron |>
  gf_area_interactive(
    level ~ year,
    tooltip = ~ "This is the area.",
    data_id = "id:area",
    fill = "skyblue"
    ) |>
```

```
  gf_line_interactive(
    tooltip = ~ "This is the line.",
    data_id = "id:line"
  ) |>
  gf_girafe(
    list(
      options = list(opts_tooltip(css = "fill: steelblue;"))
    )
  )
```

---

gf_ash *Average Shifted Histograms*

Description

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

Usage

```
gf_ash(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...,
  alpha,
  color,
  group,
  linetype,
  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,
  ...
)

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

Arguments

object	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
gformula	A formula with shape ~x or y ~ x. y may be stat(density) or stat(count) or stat(ndensity) or stat(ncount). Faceting can be achieved by including | in the formula.
data	A data frame with the variables to be plotted.
...	Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with attribute = value, (b) ggplot2 aesthetics to be mapped with attribute = ~ expression, or (c) attributes of the layer as a whole, which are set with attribute = value.
alpha	Opacity (0 = invisible, 1 = opaque).
color	A color or a formula used for mapping color.
group	Used for grouping.
linetype	A linetype (numeric or "dashed", "dotted", etc.) or a formula used for mapping linetype.
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.
mapping	set of aesthetic mappings created by ggplot2::aes()] or ggplot2::aes_().
na.rm	If FALSE (the default), removes missing values with a warning. If TRUE silently removes missing values.
inherit.aes	A logical indicating whether default aesthetics are inherited.
binwidth	the width of the histogram bins. If NULL (the default) the binwidth will be chosen so that approximately 10 bins cover the data. adjust can be used to to increase or decrease binwidth.
adjust	a numeric adjustment to binwidth. Primarily useful when binwidth is not specified. Increasing adjust makes the plot smoother.

Value

a gg object

Specifying plot attributes

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

In formulas of the form A | B, B will be used to form facets using ggplot2::facet_wrap() or ggplot2::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(), gf_histogram().

Examples

```
data(penguins, package = "palmerpenguins")
gf_ash(~bill_length_mm, color = ~species, data = penguins)
gf_ash(~bill_length_mm, color = ~species, data = penguins, adjust = 2)
gf_ash(~bill_length_mm, color = ~species, data = penguins, binwidth = 1)
gf_ash(~bill_length_mm, color = ~species, data = penguins, binwidth = 1, adjust = 2)
ggplot(faithful, aes(x = eruptions)) +
  geom_histogram(aes(y = stat(density)),
    fill = "lightskyblue", colour = "gray50", alpha = 0.2
  ) +
  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.help = 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,
  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",
  stat = "count",
  position = "stack",
  show.legend = NA,
  show.help = NULL,
  inherit = TRUE,
  environment = parent.frame(),
  denom = ~PANEL
)
```

Arguments

`object`	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
`gformula`	A formula, typically with shape `~ x`. (`y ~ x` is also possible, but typically using one of `gf_col()`, `gf_props()`, or `gf_percents()` is preferable to using this formula shape.) Faceting can be achieved by including `|` in the formula.
`data`	The data to be displayed in this layer. There are three options: If `NULL`, the default, the data is inherited from the plot data as specified in the call to `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.

`width` Width of the bars.

`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` Override the default connection between `geom_bar()` and `stat_count()`. For more information about overriding these connections, see how the stat and geom arguments work.

`position` A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The `position` argument accepts the following:

- The result of calling a position function, such as `position_jitter()`. This method allows for passing extra arguments to the position.
- A string naming the position adjustment. To give the position as a string, strip the function name of the `position_` prefix. For example, to use `position_jitter()`, give the position as `"jitter"`.
- For more information and other ways to specify the position, see the layer position documentation.

`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. To include legend keys for all levels, even when no data exists, use `TRUE`. If `NA`, all levels are shown in legend, but unobserved levels are omitted.

`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 `ggplot2::facet_wrap()` or `ggplot2::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_percents() 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))
}
```

---

gf_barh *Deprecated horizontal plotting functions*

Description

These functions were wrappers around functions from ggstance from an era before ggplot2 supported horizonally oriented geoms. ggstance has not been updated to comply with the current version of ggplot2, and since the functionalilty 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_bar_interactive *Interactive bar charts*

Description

Creates an interactive plot using ggiraph. This function extends `gf_bar()` with interactive features like tooltips and clickable elements.

Arguments

`object`	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument.
`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.
`tooltip`	A formula specifying a variable for tooltips, or a character vector.
`data_id`	A formula or character vector specifying data identifiers for interactive selection.
`...`	Additional arguments passed to the underlying geom.
`alpha`, `color`, `size`, `shape`, `fill`, `group`, `stroke`	Aesthetics passed to the geom.
`xlab`, `ylab`, `title`, `subtitle`, `caption`	Labels for the plot.
`show.legend`	Logical. Should this layer be included in the legends?
`show.help`	Logical. If `TRUE`, display some minimal help.
`inherit`	Logical. If `TRUE`, inherit aesthetics from previous layers.
`environment`	An environment in which to evaluate the formula.

Value

A gg object that can be displayed with `gf_girafe()`.

Additional interactive features

- `onclick`: JavaScript code (as character string) executed when clicking elements.
- Additional ggiraph aesthetics may be available depending on the geom.

See Also

`gf_bar()`, `gf_girafe()`

Examples

```
diamonds |>
  gf_bar_interactive(
    ~color,
    fill = ~cut,
    tooltip = ~ stage(
      start = glue::glue("color: {color}; cut: {cut}"),
      after_stat = glue::glue("{tooltip}; count = {count}")
    ),
```

```
  data_id = ~ glue::glue("{cut} -- {color}"),
  size = 3
) |>
gf_girafe()
```

---

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. gf_bin_2d() is an alias.

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()
)

gf_bin_2d(
  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 ggplot2::facet_wrap() or ggplot2::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 ggplot2::facet_wrap() or ggplot2::facet_grid(). This provides an alternative to gf_facet_wrap() and gf_facet_grid() that is terser and may feel more familiar to users of **lattice**.

Evaluation

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

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

See Also

ggplot2::geom_bin2d(), gf_tile()

Examples

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

`gf_bin2d_interactive` *Interactive bin2d plots*

Description

Creates an interactive plot using ggiraph. This function extends `gf_bin2d()` with interactive features like tooltips and clickable elements.

Arguments

`object`	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument.
`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.
`tooltip`	A formula specifying a variable for tooltips, or a character vector.
`data_id`	A formula or character vector specifying data identifiers for interactive selection.
`...`	Additional arguments passed to the underlying geom.
`alpha`, `color`, `size`, `shape`, `fill`, `group`, `stroke`	Aesthetics passed to the geom.
`xlab`, `ylab`, `title`, `subtitle`, `caption`	Labels for the plot.
`show.legend`	Logical. Should this layer be included in the legends?
`show.help`	Logical. If `TRUE`, display some minimal help.
`inherit`	Logical. If `TRUE`, inherit aesthetics from previous layers.
`environment`	An environment in which to evaluate the formula.

Value

A gg object that can be displayed with `gf_girafe()`.

Additional interactive features

- `onclick`: JavaScript code (as character string) executed when clicking elements.
- Additional ggiraph aesthetics may be available depending on the geom.

See Also

`gf_bin2d()`, `gf_girafe()`

Examples

```
gf_bin2d_interactive(mpg ~ wt, data = mtcars,
                     tooltip = ~ paste("MPG:", mpg)) |>
  gf_girafe()
```

`gf_bin_2d_interactive` *Interactive bin_2d plots*

Description

Creates an interactive plot using ggiraph. This function extends `gf_bin_2d()` with interactive features like tooltips and clickable elements.

Arguments

`object`	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument.
`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.
`tooltip`	A formula specifying a variable for tooltips, or a character vector.
`data_id`	A formula or character vector specifying data identifiers for interactive selection.
`...`	Additional arguments passed to the underlying geom.
`alpha`, `color`, `size`, `shape`, `fill`, `group`, `stroke`	Aesthetics passed to the geom.
`xlab`, `ylab`, `title`, `subtitle`, `caption`	Labels for the plot.
`show.legend`	Logical. Should this layer be included in the legends?
`show.help`	Logical. If `TRUE`, display some minimal help.
`inherit`	Logical. If `TRUE`, inherit aesthetics from previous layers.
`environment`	An environment in which to evaluate the formula.

Value

A gg object that can be displayed with `gf_girafe()`.

Additional interactive features

- `onclick`: JavaScript code (as character string) executed when clicking elements.
- Additional ggiraph aesthetics may be available depending on the geom.

See Also

`gf_bin_2d()`, `gf_girafe()`

Examples

```
gf_bin_2d_interactive(mpg ~ wt, data = mtcars,
                      tooltip = ~ paste("MPG:", mpg)) |>
  gf_girafe()
```

gf_blank *Formula interface to geom_blank()*

Description

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

Usage

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

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

Arguments

object	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
gformula	A formula with shape y ~ x. Faceting can be achieved by including | in the formula.
data	A data frame with the variables to be plotted.
...	Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with attribute = value, (b) ggplot2 aesthetics to be mapped with attribute = ~ expression, or (c) attributes of the layer as a whole, which are set with attribute = value.
xlab	Label for x-axis. See also gf_labs().
ylab	Label for y-axis. See also gf_labs().
title, 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 ggplot2::facet_wrap() or ggplot2::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

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

---

gf_boxplot *Formula interface to geom_boxplot()*

Description

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

Usage

```
gf_boxplot(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...,
  alpha,
  color,
  fill,
  group,
  linetype,
  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 = 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.
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(). For more information about overriding these connections, see how the stat and geom arguments work.

position — A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following:

- The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position.
- A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter".
- For more information and other ways to specify the position, see the layer position documentation.

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. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.

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 `ggplot2::facet_wrap()` or `ggplot2::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

McGill, R., Tukey, J. W. and Larsen, W. A. (1978) Variations of box plots. The American Statistician 32, 12-16.

See Also

`ggplot2::geom_boxplot()`, `mosaic::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
) |>
  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_boxplot_interactive` *Interactive box plots*

Description

Creates an interactive plot using ggiraph. This function extends `gf_boxplot()` with interactive features like tooltips and clickable elements.

Arguments

`object`	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument.
`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.
`tooltip`	A formula specifying a variable for tooltips, or a character vector.
`data_id`	A formula or character vector specifying data identifiers for interactive selection.
`...`	Additional arguments passed to the underlying geom.
`alpha`, `color`, `size`, `shape`, `fill`, `group`, `stroke`	Aesthetics passed to the geom.
`xlab`, `ylab`, `title`, `subtitle`, `caption`	Labels for the plot.
`show.legend`	Logical. Should this layer be included in the legends?
`show.help`	Logical. If `TRUE`, display some minimal help.
`inherit`	Logical. If `TRUE`, inherit aesthetics from previous layers.
`environment`	An environment in which to evaluate the formula.

Value

A gg object that can be displayed with `gf_girafe()`.

Additional interactive features

- `onclick`: JavaScript code (as character string) executed when clicking elements.
- Additional ggiraph aesthetics may be available depending on the geom.

See Also

`gf_boxplot()`, `gf_girafe()`

Examples

```
mtcars |>
  gf_boxplot_interactive(
    mpg ~ factor(cyl),
    tooltip = ~ paste("Cylinders:", cyl)
  ) |>
  gf_girafe()
```

---

gf_col *Formula interface to geom_col()*

Description

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

Usage

```
gf_col(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...,
  alpha,
  color,
  fill,
  group,
  linetype,
  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 ~ 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 ggplot2::facet_wrap() or ggplot2::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

```
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_col_interactive *Interactive column charts*

Description

Creates an interactive plot using ggiraph. This function extends `gf_col()` with interactive features like tooltips and clickable elements.

Arguments

`object`	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument.
`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.
`tooltip`	A formula specifying a variable for tooltips, or a character vector.
`data_id`	A formula or character vector specifying data identifiers for interactive selection.
`...`	Additional arguments passed to the underlying geom.
`alpha`, `color`, `size`, `shape`, `fill`, `group`, `stroke`	Aesthetics passed to the geom.
`xlab`, `ylab`, `title`, `subtitle`, `caption`	Labels for the plot.
`show.legend`	Logical. Should this layer be included in the legends?
`show.help`	Logical. If `TRUE`, display some minimal help.
`inherit`	Logical. If `TRUE`, inherit aesthetics from previous layers.
`environment`	An environment in which to evaluate the formula.

Value

A gg object that can be displayed with `gf_girafe()`.

Additional interactive features

- `onclick`: JavaScript code (as character string) executed when clicking elements.
- Additional ggiraph aesthetics may be available depending on the geom.

See Also

`gf_col()`, `gf_girafe()`

Examples

```
if (require(dplyr)) {
  library(dplyr)
  diamonds |>
    group_by(color, cut) |>
    summarise(count = n()) |>
    gf_col_interactive(
      count ~ color,
      fill = ~cut,
      tooltip = ~ glue::glue("color: {color}, cut: {cut}, count: {count}"),
      data_id = ~ glue::glue("{cut} - {color}")
    ) |>
  gf_girafe()
}
```

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.

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()`. 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 for this layer. When using a `stat_*()` function to construct a layer, the `geom` argument can be used to override the default coupling between stats and geoms. The `geom` argument accepts the following: • A `Geom` ggproto subclass, for example `GeomPoint`. • A string naming the geom. To give the geom as a string, strip the function name of the `geom_` prefix. For example, to use `geom_point()`, give the geom as `"point"`. • For more information and other ways to specify the geom, see the layer geom documentation.
stat	The statistical transformation to use on the data for this layer. When using a `geom_*()` function to construct a layer, the `stat` argument can be used to override the default coupling between geoms and stats. The `stat` argument accepts the following: • A `Stat` ggproto subclass, for example `StatCount`.

- A string naming the stat. To give the stat as a string, strip the function name of the `stat_` prefix. For example, to use `stat_count()`, give the stat as `"count"`.
- For more information and other ways to specify the stat, see the layer stat documentation.

`position` A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The `position` argument accepts the following:

- The result of calling a position function, such as `position_jitter()`. This method allows for passing extra arguments to the position.
- A string naming the position adjustment. To give the position as a string, strip the function name of the `position_` prefix. For example, to use `position_jitter()`, give the position as `"jitter"`.
- For more information and other ways to specify the position, see the layer position documentation.

`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. To include legend keys for all levels, even when no data exists, use `TRUE`. If `NA`, all levels are shown in legend, but unobserved levels are omitted.

`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 `ggplot2::facet_wrap()` or `ggplot2::facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of **lattice**.

Evaluation

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

See Also

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

Examples

```
gf_density_2d(eruptions ~ waiting, data = faithful, alpha = 0.5, color = "navy") |>
  gf_contour(density ~ waiting + eruptions, data = 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_contour_filled_interactive

Interactive 2-demensional contour plots

Description

Creates an interactive plot using ggiraph. These functions extend `gf_contour()` and `gf_contour_filled()` with interactive features like tooltips and clickable elements.

Arguments

`object`	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument.
`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.
`tooltip`	A formula specifying a variable for tooltips, or a character vector.
`data_id`	A formula or character vector specifying data identifiers for interactive selection.
`...`	Additional arguments passed to the underlying geom.
`alpha`, `color`, `size`, `shape`, `fill`, `group`, `stroke`	Aesthetics passed to the geom.
`xlab`, `ylab`, `title`, `subtitle`, `caption`	Labels for the plot.
`show.legend`	Logical. Should this layer be included in the legends?
`show.help`	Logical. If `TRUE`, display some minimal help.
`inherit`	Logical. If `TRUE`, inherit aesthetics from previous layers.
`environment`	An environment in which to evaluate the formula.

Value

A gg object that can be displayed with `gf_girafe()`.

Examples

```
faithfuld |>
  gf_contour_interactive(
    density ~ waiting + eruptions,
    color = ~ after_stat(level),
    tooltip = ~ after_stat(paste0("density: ", level)),
    data_id = ~ after_stat(level),
    hover_css = "stroke: red;",
    hover_nearest = TRUE,
    bins = 10, show.legend = FALSE) |>
  gf_girafe()

faithfuld |>
  gf_contour_filled_interactive(
    density ~ waiting + eruptions,
    fill = ~ after_stat(level),
    tooltip = ~ after_stat(paste0("density: ", level)),
    data_id = ~ after_stat(level),
    hover_css = "fill: red; opacity: 0.5",
    hover_nearest = TRUE,
    bins = 10, show.legend = FALSE) |>
  gf_girafe()
```

---

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 ~ x. Faceting can be achieved by including | in the formula.
data	A data frame with the variables to be plotted.
...	Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with attribute = value, (b) ggplot2 aesthetics to be mapped with attribute = ~ expression, or (c) attributes of the layer as a whole, which are set with attribute = value.
alpha	Opacity (0 = invisible, 1 = opaque).
color	A color or a formula used for mapping color.
fill	A color for filling, or a formula used for mapping fill.
group	Used for grouping.
shape	An integer or letter shape or a formula used for mapping shape.
size	A numeric size or a formula used for mapping size.
stroke	A numeric size of the border or a formula used to map stroke.
xlab	Label for x-axis. See also gf_labs().
ylab	Label for y-axis. See also gf_labs().
title, 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 `ggplot2::facet_wrap()` or `ggplot2::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

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

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

---

gf_count_interactive *Interactive count plots*

Description

Creates an interactive plot using ggiraph. This function extends `gf_count()` with interactive features like tooltips and clickable elements.

Arguments

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

tooltip	A formula specifying a variable for tooltips, or a character vector.
data_id	A formula or character vector specifying data identifiers for interactive selection.
...	Additional arguments passed to the underlying geom.
alpha, color, size, shape, fill, group, stroke	Aesthetics passed to the geom.
xlab, ylab, title, subtitle, caption	Labels for the plot.
show.legend	Logical. Should this layer be included in the legends?
show.help	Logical. If TRUE, display some minimal help.
inherit	Logical. If TRUE, inherit aesthetics from previous layers.
environment	An environment in which to evaluate the formula.

Value

A gg object that can be displayed with gf_girafe().

Additional interactive features

- onclick: JavaScript code (as character string) executed when clicking elements.
- Additional ggiraph aesthetics may be available depending on the geom.

See Also

gf_count(), gf_density2d_interactive(), gf_girafe()

Examples

```
diamonds |>
  gf_count_interactive(
    clarity ~ cut,
    size = ~ after_stat(n),
    tooltip = ~ after_stat(paste0("count: ", n)),
    show.legend = FALSE
  ) |>
  gf_girafe()
```

---

gf_crossbar *Formula interface to geom_crossbar()*

Description

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

Usage

```
gf_crossbar(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...,
  alpha,
  color,
  group,
  middle.linetype,
  middle.colour = NULL,
  middle.color = NULL,
  middle.linewidth = NULL,
  box.colour = NULL,
  box.color = NULL,
  box.linetype = NULL,
  box.linewidth = NULL,
  na.rm = FALSE,
  inherit.aes = TRUE,
  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.

middle.linetype, middle.colour, middle.color, middle.linewidth
: Arguments to control the middle line.

box.colour, box.color, box.linetype, box.linewidth
: Arguments to control the box.

na.rm
: If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.

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. annotation_borders().

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. When using a geom_*() function to construct a layer, the stat argument can be used to override the default coupling between geoms and stats. The stat argument accepts the following:

  - A Stat ggproto subclass, for example StatCount.
  - A string naming the stat. To give the stat as a string, strip the function name of the stat_ prefix. For example, to use stat_count(), give the stat as "count".
  - For more information and other ways to specify the stat, see the layer stat documentation.

position
: A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following:

  - The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position.
  - A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter".
  - For more information and other ways to specify the position, see the layer position documentation.

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. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.
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 ggplot2::facet_wrap() or ggplot2::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 |>
    summarise(.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", middle.linewidth = 1, middle.color = "red") |>
  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_crossbar_interactive` *Interactive crossbar plots*

Description

Creates an interactive plot using ggiraph. This function extends `gf_crossbar()` with interactive features like tooltips and clickable elements.

Arguments

`object`	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument.
`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.
`tooltip`	A formula specifying a variable for tooltips, or a character vector.
`data_id`	A formula or character vector specifying data identifiers for interactive selection.
`...`	Additional arguments passed to the underlying geom.
`alpha`, `color`, `size`, `shape`, `fill`, `group`, `stroke`	Aesthetics passed to the geom.
`xlab`, `ylab`, `title`, `subtitle`, `caption`	Labels for the plot.
`show.legend`	Logical. Should this layer be included in the legends?
`show.help`	Logical. If `TRUE`, display some minimal help.
`inherit`	Logical. If `TRUE`, inherit aesthetics from previous layers.
`environment`	An environment in which to evaluate the formula.

Value

A gg object that can be displayed with gf_girafe().

Additional interactive features

- onclick: JavaScript code (as character string) executed when clicking elements.
- Additional ggiraph aesthetics may be available depending on the geom.

See Also

gf_crossbar(), gf_girafe()

Examples

```
diamonds |>
  dplyr::filter(carat < 1.1, carat > 0.9) |>
  dplyr::group_by(color, cut) |>
  dplyr::summarise(
    median_price = median(price) |> round(),
    lower = quantile(price, 0.25) |> round(),
    upper = quantile(price, 0.75) |> round(),
    iqr = upper - lower
  ) |>
  gf_crossbar_interactive(
    cut ~ median_price + lower + upper | color,
    color = ~ cut,
    tooltip = ~ paste0(
      "75th percentile: ", upper,
      "\nmedian: ", median_price,
      "\n25th percentile: ", lower
      )
  ) |>
  gf_girafe()
```

---

gf_curve *Formula interface to geom_curve()*

Description

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

Usage

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

Arguments

object	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
gformula	A formula with shape y + yend ~ 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. ~ 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. When using a geom_*() function to construct a layer, the stat argument can be used to override the default coupling between geoms and stats. The stat argument accepts the following:

- A Stat ggproto subclass, for example StatCount.
- A string naming the stat. To give the stat as a string, strip the function name of the stat_ prefix. For example, to use stat_count(), give the stat as "count".
- For more information and other ways to specify the stat, see the layer stat documentation.

position — A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following:

- The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position.
- A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter".
- For more information and other ways to specify the position, see the layer position documentation.

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. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.
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 ggplot2::facet_wrap() or ggplot2::facet_grid(). This provides an alternative to gf_facet_wrap() and gf_facet_grid() that is terser and may feel more familiar to users of **lattice**.

Evaluation

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

See Also

ggplot2::geom_curve()

Examples

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

gf_curve_interactive *Interactive curve plots*

Description

Creates an interactive plot using ggiraph. These functions extend gf_segment() and gf_curve() with interactive features like tooltips and clickable elements.

Creates an interactive plot using ggiraph. This function extends gf_segment() with interactive features like tooltips and clickable elements.

Arguments

object	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument.
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.
tooltip	A formula specifying a variable for tooltips, or a character vector.
data_id	A formula or character vector specifying data identifiers for interactive selection.
...	Additional arguments passed to the underlying geom.
alpha, color, size, shape, fill, group, stroke	Aesthetics passed to the geom.
xlab, ylab, title, subtitle, caption	Labels for the plot.
show.legend	Logical. Should this layer be included in the legends?
show.help	Logical. If TRUE, display some minimal help.
inherit	Logical. If TRUE, inherit aesthetics from previous layers.
environment	An environment in which to evaluate the formula.

Value

A gg object that can be displayed with gf_girafe().

A gg object that can be displayed with gf_girafe().

Additional interactive features

- onclick: JavaScript code (as character string) executed when clicking elements.
- Additional ggiraph aesthetics may be available depending on the geom.

- onclick: JavaScript code (as character string) executed when clicking elements.
- Additional ggiraph aesthetics may be available depending on the geom.

See Also

gf_curve(), gf_girafe()

gf_segment(), gf_girafe()

Examples

```
gf_curve_interactive(
  1 + 2 ~ 0 + 4, color = "red", curvature = - 0.2,
  tooltip = ~ "curvature: -0.2",
  data_id = 0.2
) |>
gf_curve_interactive(
  1 + 2 ~ 0 + 4, color = "blue", curvature = 0.4,
  tooltip = ~ "curvature: 0.4",
  data_id = 0.4) |>
gf_segment_interactive(
  1 + 2 ~ 0 + 4, color = "green",
  tooltip = ~ "curvature: 0",
  data_id = 0
) |>
gf_girafe(
  options = list(
    opts_hover(css = "stroke: black; stroke-width: 3;", nearest_distance = 10)
  )
)
```

---

gf_density *Formula interface to stat_density()*

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

```
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,
  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)).
...	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.
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.
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(). For more information about overriding these connections, see how the stat and geom arguments work.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.
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. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.
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 `ggplot2::facet_wrap()` or `ggplot2::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)
```

---

gf_density_2d *Formula interface to geom_density_2d() and geom_density_2d_filled()*

Description

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

Usage

```
gf_density_2d(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...,
  alpha,
  color,
  group,
  linetype,
  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,
  xlab,
  ylab,
  title,
  subtitle,
  caption,
  geom = "density2d",
  stat = "density2d",
  position = "identity",
  show.legend = NA,
  show.help = NULL,
  inherit = TRUE,
  environment = parent.frame()
)

gf_density2d_filled(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...,
```

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

Arguments

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

position — A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following:

- The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position.
- A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter".
- For more information and other ways to specify the position, see the layer position documentation.

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. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.

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 `ggplot2::facet_wrap()` or `ggplot2::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

```
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_density_2d_filled_interactive
Interactive 2-demensional density plots

Description

Creates an interactive plot using ggiraph. These functions extend gf_density2d(), gf_density_2d(), gf_density2d_filled(), and gf_density_2d_filled() with interactive features like tooltips and clickable elements.

Arguments

object	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument.
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.
tooltip	A formula specifying a variable for tooltips, or a character vector.
data_id	A formula or character vector specifying data identifiers for interactive selection.
...	Additional arguments passed to the underlying geom.
alpha, color, size, shape, fill, group, stroke	Aesthetics passed to the geom.
xlab, ylab, title, subtitle, caption	Labels for the plot.
show.legend	Logical. Should this layer be included in the legends?
show.help	Logical. If TRUE, display some minimal help.
inherit	Logical. If TRUE, inherit aesthetics from previous layers.
environment	An environment in which to evaluate the formula.

Value

A gg object that can be displayed with gf_girafe().

Additional interactive features

- onclick: JavaScript code (as character string) executed when clicking elements.
- Additional ggiraph aesthetics may be available depending on the geom.

See Also

gf_density_2d_filled(), gf_density_2d(), gf_contour_interactive(), gf_girafe()

Examples

```
faithful |>
  gf_density2d_filled_interactive(
    eruptions ~ waiting,
    tooltip = ~ after_stat(level),
    data_id = ~ after_stat(level),
    show.legend = FALSE
  ) |>
  gf_girafe()

faithful |>
  gf_density2d_interactive(
    eruptions ~ waiting,
    tooltip = ~ after_stat(level),
    data_id = ~ after_stat(level),
    show.legend = FALSE
  ) |>
  gf_girafe()
```

`gf_density_interactive` *Interactive density plots*

Description

Creates an interactive plot using ggiraph. This function extends `gf_density()` with interactive features like tooltips and clickable elements.

Arguments

`object`	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument.
`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.
`tooltip`	A formula specifying a variable for tooltips, or a character vector.
`data_id`	A formula or character vector specifying data identifiers for interactive selection.
`...`	Additional arguments passed to the underlying geom.
`alpha`, `color`, `size`, `shape`, `fill`, `group`, `stroke`	Aesthetics passed to the geom.
`xlab`, `ylab`, `title`, `subtitle`, `caption`	Labels for the plot.
`show.legend`	Logical. Should this layer be included in the legends?
`show.help`	Logical. If `TRUE`, display some minimal help.
`inherit`	Logical. If `TRUE`, inherit aesthetics from previous layers.
`environment`	An environment in which to evaluate the formula.

Value

A gg object that can be displayed with gf_girafe().

Additional interactive features

- onclick: JavaScript code (as character string) executed when clicking elements.
- Additional ggiraph aesthetics may be available depending on the geom.

See Also

gf_density(), gf_girafe()

Examples

```
diamonds |>
  gf_density_interactive(
    ~ carat,
    fill = ~ cut,
    color = ~ cut,
    data_id = ~ cut,
    tooltip = ~ cut) |>
  gf_girafe()
```

---

gf_dist *Plot distributions*

Description

Create a layer displaying a probability distribution.

Usage

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

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

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

---

gf_dotplot *Formula interface to geom_dotplot()*

Description

Scatterplots in ggformula.

Usage

```
gf_dotplot(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...,
  alpha,
  color,
  fill,
  group,
  binwidth = NULL,
  binaxis = "x",
  method = "dotdensity",
  binpositions = "bygroup",
  stackdir = "up",
  stackratio = 1,
  dotsize = 1,
  stackgroups = FALSE,
  origin = NULL,
  right = TRUE,
  width = 0.9,
  drop = FALSE,
  xlab,
  ylab,
  title,
  subtitle,
  caption,
  position = "identity",
  show.legend = NA,
  show.help = NULL,
  inherit = TRUE,
  environment = parent.frame()
)
```

Arguments

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

fill	A color for filling, or a formula used for mapping fill.
group	Used for grouping.
binwidth	When method is "dotdensity", this specifies maximum bin width. When method is "histodot", this specifies bin width. Defaults to 1/30 of the range of the data
binaxis	The axis to bin along, "x" (default) or "y"
method	"dotdensity" (default) for dot-density binning, or "histodot" for fixed bin widths (like stat_bin)
binpositions	When method is "dotdensity", "bygroup" (default) determines positions of the bins for each group separately. "all" determines positions of the bins with all the data taken together; this is used for aligning dot stacks across multiple groups.
stackdir	which direction to stack the dots. "up" (default), "down", "center", "centerwhole" (centered, but with dots aligned)
stackratio	how close to stack the dots. Default is 1, where dots just touch. Use smaller values for closer, overlapping dots.
dotsize	The diameter of the dots relative to binwidth, default 1.
stackgroups	should dots be stacked across groups? This has the effect that position = "stack" should have, but can't (because this geom has some odd properties).
origin	When method is "histodot", origin of first bin
right	When method is "histodot", should intervals be closed on the right (a, b], or not [a, b)
width	When binaxis is "y", the spacing of the dot stacks for dodging.
drop	If TRUE, remove all bins with zero counts
xlab	Label for x-axis. See also gf_labs().
ylab	Label for y-axis. See also gf_labs().
title, 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 ggplot2::facet_wrap() or ggplot2::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

Wilkinson, L. (1999) Dot plots. The American Statistician, 53(3), 276-281.

See Also

ggplot2::geom_dotplot()

Examples

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

---

gf_dotplot_interactive

Interactive dotplot plots

Description

Creates an interactive plot using ggiraph. This function extends gf_dotplot() with interactive features like tooltips and clickable elements.

Arguments

object	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument.
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.
tooltip	A formula specifying a variable for tooltips, or a character vector.
data_id	A formula or character vector specifying data identifiers for interactive selection.
...	Additional arguments passed to the underlying geom.
alpha, color, size, shape, fill, group, stroke	Aesthetics passed to the geom.
xlab, ylab, title, subtitle, caption	Labels for the plot.
show.legend	Logical. Should this layer be included in the legends?
show.help	Logical. If TRUE, display some minimal help.
inherit	Logical. If TRUE, inherit aesthetics from previous layers.
environment	An environment in which to evaluate the formula.

Value

A gg object that can be displayed with gf_girafe().

Additional interactive features

- onclick: JavaScript code (as character string) executed when clicking elements.
- Additional ggiraph aesthetics may be available depending on the geom.

See Also

gf_dotplot(), gf_girafe()

---

gf_ecdf *Formula interace to empirical cumulative distribution*

Description

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

Usage

```
gf_ecdf(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...,
  group,
  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()'s params argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the position argument, or aesthetics that are required can *not* be passed through .... Unknown arguments that are not part of the 4 categories below are ignored. • Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, colour = "red" or linewidth = 3. The geom's documentation has an **Aesthetics** section that lists the available options. The 'required' aesthetics cannot be passed on to the params. Please note that while passing unmapped aesthetics as vectors is

technically possible, the order and required length is not guaranteed to be parallel to the input data.

- When constructing a layer using a `stat_*()` function, the `...` argument can be used to pass on parameters to the `geom` part of the layer. An example of this is `stat_density(geom = "area", outline.type = "both")`. The geom's documentation lists which parameters it can accept.
- Inversely, when constructing a layer using a `geom_*()` function, the `...` argument can be used to pass on parameters to the `stat` part of the layer. An example of this is `geom_area(stat = "density", adjust = 0.5)`. The stat's documentation lists which parameters it can accept.
- The `key_glyph` argument of `layer()` may also be passed on through `....` This can be one of the functions described as key glyphs, to change the display of the layer in the legend.

`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 for this layer. When using a `stat_*()` function to construct a layer, the `geom` argument can be used to override the default coupling between stats and geoms. The `geom` argument accepts the following:

- A `Geom` ggproto subclass, for example `GeomPoint`.
- A string naming the geom. To give the geom as a string, strip the function name of the `geom_` prefix. For example, to use `geom_point()`, give the geom as `"point"`.
- For more information and other ways to specify the geom, see the layer geom documentation.

`stat` The statistical transformation to use on the data for this layer. When using a `geom_*()` function to construct a layer, the `stat` argument can be used to override the default coupling between geoms and stats. The `stat` argument accepts the following:

- A `Stat` ggproto subclass, for example `StatCount`.
- A string naming the stat. To give the stat as a string, strip the function name of the `stat_` prefix. For example, to use `stat_count()`, give the stat as `"count"`.
- For more information and other ways to specify the stat, see the layer stat documentation.

`position` A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The `position` argument accepts the following:

- The result of calling a position function, such as `position_jitter()`. This method allows for passing extra arguments to the position.
- A string naming the position adjustment. To give the position as a string, strip the function name of the `position_` prefix. For example, to use `position_jitter()`, give the position as `"jitter"`.
- For more information and other ways to specify the position, see the layer position documentation.

`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. To include legend keys for all levels, even when no data exists, use `TRUE`. If `NA`, all levels are shown in legend, but unobserved levels are omitted.

`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 *Formula interface to stat_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()

Examples

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

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

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

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

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

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

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

gf_empty *Create an "empty" plot*

Description

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

Usage

```
gf_empty(environment = parent.frame())
```

Arguments

environment	An environment passed to ggplot2::ggplot()

Value

A plot with now layers.

Examples

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

gf_errorbar *Formula interface to geom_errorbar()*

Description

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

Usage

```
gf_errorbar(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...,
  alpha,
  color,
  group,
  linetype,
  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 for this layer. When using a `stat_*()` function to construct a layer, the `geom` argument can be used to override the default coupling between stats and geoms. The `geom` argument accepts the following:

- A `Geom` ggproto subclass, for example `GeomPoint`.
- A string naming the geom. To give the geom as a string, strip the function name of the `geom_` prefix. For example, to use `geom_point()`, give the geom as `"point"`.
- For more information and other ways to specify the geom, see the layer geom documentation.

`stat` The statistical transformation to use on the data for this layer. When using a `geom_*()` function to construct a layer, the `stat` argument can be used to override the default coupling between geoms and stats. The `stat` argument accepts the following:

- A `Stat` ggproto subclass, for example `StatCount`.
- A string naming the stat. To give the stat as a string, strip the function name of the `stat_` prefix. For example, to use `stat_count()`, give the stat as `"count"`.
- For more information and other ways to specify the stat, see the layer stat documentation.

`position` A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The `position` argument accepts the following:

- The result of calling a position function, such as `position_jitter()`. This method allows for passing extra arguments to the position.
- A string naming the position adjustment. To give the position as a string, strip the function name of the `position_` prefix. For example, to use `position_jitter()`, give the position as `"jitter"`.
- For more information and other ways to specify the position, see the layer position documentation.

`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. To include legend keys for all levels, even when no data exists, use `TRUE`. If `NA`, all levels are shown in legend, but unobserved levels are omitted.

`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

```
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_errorbar_interactive

Interactive errorbar plots

Description

Creates an interactive plot using ggiraph. This function extends `gf_errorbar()` with interactive features like tooltips and clickable elements.

Arguments

`object`	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument.
`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.
`tooltip`	A formula specifying a variable for tooltips, or a character vector.

data_id	A formula or character vector specifying data identifiers for interactive selection.
...	Additional arguments passed to the underlying geom, plus any ggiraph::interactive_parameters(interactive parameters).
alpha, color, size, shape, fill, group, stroke	Aesthetics passed to the geom.
xlab, ylab, title, subtitle, caption	Labels for the plot.
show.legend	Logical. Should this layer be included in the legends?
show.help	Logical. If TRUE, display some minimal help.
inherit	Logical. If TRUE, inherit aesthetics from previous layers.
environment	An environment in which to evaluate the formula.

Value

A gg object that can be displayed with gf_girafe().

Additional interactive features

- onclick: JavaScript code (as character string) executed when clicking elements.
- Additional ggiraph aesthetics may be available depending on the geom.

See Also

gf_errorbar(), gf_girafe()

Examples

```
diamonds |>
  dplyr::filter(carat < 1.1, carat > 0.9) |>
  dplyr::group_by(color, cut) |>
  dplyr::summarise(
    median_price = median(price) |> round(),
    lower = quantile(price, 0.25) |> round(),
    upper = quantile(price, 0.75) |> round(),
    iqr = upper - lower
  ) |>
  gf_errorbar_interactive(
    cut ~ lower + upper | color,
    color = ~ cut,
    tooltip = ~ paste0(
      "75th percentile: ", upper,
      "\nmedian: ", median_price,
      "\n25th percentile: ", lower
      )
  ) |>
  gf_girafe()
```

gf_facet_wrap *Add facets to a plot*

Description

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

Usage

```
gf_facet_wrap(object, ...)

gf_facet_grid(object, ...)
```

Arguments

object	A ggplot object
...	Additional arguments passed to ggplot2::facet_wrap() or ggplot2::facet_grid(). This typically includes an unnamed formula argument describing the facets. scales and space are additional useful arguments. See the examples.

See Also

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

Examples

```
gf_histogram(~avg_drinks, data = mosaicData::HELPrct, bins =25) |>
  gf_facet_grid(~substance)
gf_histogram(~avg_drinks, data = mosaicData::HELPrct, bins =25) |>
  gf_facet_grid(~substance, scales = "free")
gf_histogram(~avg_drinks, data = mosaicData::HELPrct, bins =25) |>
  gf_facet_grid(~substance, scales = "free", space = "free")
gf_line(births ~ date, data = mosaicData::Births, color = ~wday) |>
  gf_facet_wrap(~year, scales = "free_x", nrow = 5) |>
  gf_theme(
    axis.title.x = element_blank(),
    axis.text.x = element_blank(),
    axis.ticks.x = element_blank()
  ) |>
  gf_labs(color = "Day")
```

gf_fitdistr *Plot density function based on fit to data*

Description

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

Usage

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

Arguments

object	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See examples.
gformula	A formula with shape ~ x used to specify the data to be fit to a family of distributions.
data	A data frame containing the variable to be fitted.
...	Additional arguments

dist	A quoted name of a distribution function. See mosaicCore::fit_distr_fun() for more details about allowable distributions.
start	Starting value(s) for the search for MLE. (See MASS::fitdistr.)
alpha	Opacity (0 = invisible, 1 = opaque).
color	A color or a formula used for mapping color.
fill	A color for filling, or a formula used for mapping fill.
group	Used for grouping.
linetype	A linetype (numeric or "dashed", "dotted", etc.) or a formula used for mapping linetype.
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 ggplot2::facet_wrap() or ggplot2::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)
fitted_density <- mosaicCore::fit_distr_fun(~g, data = Dat, dist = "dgamma")
gf_dhistogram(~g, data = Dat) |>
  gf_fun(fitted_density(x) ~ x, inherit = FALSE)

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

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

---

gf_freqpoly *Formula interface to geom_freqpoly()*

Description

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

Usage

```
gf_freqpoly(
  object = NULL,
```

```
  gformula = NULL,
  data = NULL,
  ...,
  alpha,
  color,
  group,
  linetype,
  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.
gformula	A formula with shape ~ x or 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.

`binwidth` The width of the bins. Can be specified as a numeric value or as a function that takes x after scale transformation as input and returns a single numeric value. 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.

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

`center`, `boundary`

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.

`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()`. For more information at overriding these connections, see how the stat and geom arguments work.

`position` A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The `position` argument accepts the following:

- The result of calling a position function, such as `position_jitter()`. This method allows for passing extra arguments to the position.
- A string naming the position adjustment. To give the position as a string, strip the function name of the `position_` prefix. For example, to use `position_jitter()`, give the position as `"jitter"`.
- For more information and other ways to specify the position, see the layer position documentation.

`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. To include legend keys for all levels, even when no data exists, use `TRUE`. If `NA`, all levels are shown in legend, but unobserved levels are omitted.

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 ggplot2::facet_wrap() or ggplot2::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_freqpoly_interactive

Interactive freqpoly plots

Description

Creates an interactive plot using ggiraph. This function extends gf_freqpoly() with interactive features like tooltips and clickable elements.

Arguments

object	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument.
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.
tooltip	A formula specifying a variable for tooltips, or a character vector.
data_id	A formula or character vector specifying data identifiers for interactive selection.
...	Additional arguments passed to the underlying geom.
alpha, color, size, shape, fill, group, stroke	Aesthetics passed to the geom.
xlab, ylab, title, subtitle, caption	Labels for the plot.
show.legend	Logical. Should this layer be included in the legends?
show.help	Logical. If TRUE, display some minimal help.
inherit	Logical. If TRUE, inherit aesthetics from previous layers.
environment	An environment in which to evaluate the formula.

Value

A gg object that can be displayed with gf_girafe().

Additional interactive features

- onclick: JavaScript code (as character string) executed when clicking elements.
- Additional ggiraph aesthetics may be available depending on the geom.

See Also

gf_freqpoly(), gf_girafe()

Examples

```
gf_freqpoly_interactive(mpg ~ wt, data = mtcars,
                        tooltip = ~ paste("MPG:", mpg)) |>
  gf_girafe()
```

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, ..., 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

```
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")
```

gf_function_2d *Plot functions of two variables*

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_fun_2d(
  object = NULL,
  formula = NULL,
  xlim = NULL,
  ylim = NULL,
  tile = TRUE,
  contour = TRUE,
  ...,
  resolution = 50
)

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

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

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

Arguments

object	An R object, typically of class "gg".
fun	A function of two variables to be plotted.
xlim	x limits for generating points to be plotted.
ylim	y limits for generating points to be plotted.

...	additional arguments passed to gf_tile() or gf_contour().
tile	A logical indicating whether the tile layer should be drawn.
contour	A logical indicating whether the contour layer should be drawn.
resolution	A numeric vector of length 1 or 2 specifying the number of grid points at which the function is evaluated (in each dimension).
formula	A formula describing a function of two variables to be plotted. See mosaic::makeFun() for details regarding the conversion from a formula to a function.

Value

A gg plot.

Examples

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

---

gf_girafe *Render interactive ggformula plots*

Description

Converts a ggplot object with interactive elements into an interactive HTML widget using ggiraph. This function is a wrapper around ggiraph::girafe() tailored for ggformula interactive plots.

Usage

```
gf_girafe(ggobj, code, ...)
```

Arguments

ggobj	A ggplot object, typically created with a gf_*_interactive() function.
code	R code to execute. This parameter is optional and rarely used in typical workflows.
...	Additional arguments passed to ggiraph::girafe(), such as width_svg, height_svg, options, etc.

Details

This function takes a ggplot object containing interactive elements (created with `gf_*_interactive()` functions) and renders it as an interactive plot. The resulting widget supports features like tooltips, hover effects, and clickable elements.

Value

An interactive HTML widget that can be displayed in RStudio Viewer, R Markdown documents, or Shiny applications.

See Also

`ggiraph::girafe()`, `gf_point_interactive()`, and other `gf_*_interactive()` functions

Examples

```
library(ggformula)

# Basic interactive plot
gf_point_interactive(mpg ~ wt, data = mtcars,
                     tooltip = ~ paste("Car:", rownames(mtcars))) |>
  gf_girafe()

# With custom sizing
gf_histogram_interactive(~ mpg, data = mtcars,
                         tooltip = ~ paste("Count:", after_stat(count))) |>
  gf_girafe(width_svg = 8, height_svg = 6)
```

---

gf_guides *Guides for ggformula*

Description

Guides for each scale can be set scale-by-scale with the `guide` argument, or en masse with `gf_guides()`.

Usage

```
gf_guides(object, ...)
```

Arguments

`object`	a gg object
`...`	arguments passed to `ggplot2::guides()`.

Value

a modified gg object

See Also

ggplot2::guides()

Examples

```
dat <-
  data.frame(
    x = 1:5, y = 1:5,
    p = 1:5, q = factor(1:5), r = factor(1:5)
 )

p <-
  dat |>
  gf_point( y ~ x, colour = ~ p, size = ~ q, shape = ~r)

# without guide specification
p

# Show colorbar guide for colour.
# All these examples below have a same effect.

p |> gf_guides(colour = "colorbar", size = "legend", shape = "legend")
p |> gf_guides(colour = guide_colorbar(), size = guide_legend(),
           shape = guide_legend())
p +
  scale_colour_continuous(guide = "colorbar") +
  scale_size_discrete(guide = "legend") +
  scale_shape(guide = "legend")

# Remove some guides
p |> gf_guides(colour = "none")
p |> gf_guides(colour = "colorbar",size = "none")

# Guides are integrated where possible

p |>
  gf_guides(
    colour = guide_legend("title"),
    size = guide_legend("title"),
    shape = guide_legend("title")
  )
g <- guide_legend("title")
p |> gf_guides(colour = g, size = g, shape = g)

p |> gf_theme(legend.position = "bottom")

# position of guides

# Set order for multiple guides
mpg |>
  gf_point(cty ~ displ, size = ~hwy, shape = ~ drv) |>
  gf_guides(
```

```
    colour = guide_colourbar(order = 1),
    shape = guide_legend(order = 2),
    size = guide_legend(order = 3)
  )
```

---

gf_hex *Formula interface to geom_hex()*

Description

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

Usage

```
gf_hex(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...,
  bins,
  binwidth,
  alpha,
  color,
  fill,
  group,
  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 — Number of bins. Overridden by binwidth. Defaults to 30.

binwidth — The width of the bins. Can be specified as a numeric value or as a function that takes x after scale transformation as input and returns a single numeric value. When specifying a function along with a grouping structure, the function will be called once per group. The default is to use the number of bins in bins, covering the range of the data. You should always override this value, exploring multiple widths to find the best to illustrate the stories in your data.

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

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

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

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

group — Used for grouping.

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

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, stat — Override the default connection between geom_hex() and stat_bin_hex(). For more information about overriding these connections, see how the stat and geom arguments work.

position — A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following:

- The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position.
- A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter".

- For more information and other ways to specify the position, see the layer position documentation.

`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. To include legend keys for all levels, even when no data exists, use `TRUE`. If `NA`, all levels are shown in legend, but unobserved levels are omitted.

`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 `ggplot2::facet_wrap()` or `ggplot2::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::HELPrct, bins = 15) |>
  gf_density2d(avg_drinks ~ age, data = mosaicData::HELPrct, color = "red", alpha = 0.5)
```

gf_hex_interactive *Interactive hex plots*

Description

Creates an interactive plot using ggiraph. This function extends `gf_hex()` with interactive features like tooltips and clickable elements.

Arguments

`object`	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument.
`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.
`tooltip`	A formula specifying a variable for tooltips, or a character vector.
`data_id`	A formula or character vector specifying data identifiers for interactive selection.
`...`	Additional arguments passed to the underlying geom.
`alpha`, `color`, `size`, `shape`, `fill`, `group`, `stroke`	Aesthetics passed to the geom.
`xlab`, `ylab`, `title`, `subtitle`, `caption`	Labels for the plot.
`show.legend`	Logical. Should this layer be included in the legends?
`show.help`	Logical. If `TRUE`, display some minimal help.
`inherit`	Logical. If `TRUE`, inherit aesthetics from previous layers.
`environment`	An environment in which to evaluate the formula.

Value

A gg object that can be displayed with `gf_girafe()`.

Additional interactive features

- `onclick`: JavaScript code (as character string) executed when clicking elements.
- Additional ggiraph aesthetics may be available depending on the geom.

See Also

`gf_hex()`, `gf_girafe()`

Examples

```
gf_hex_interactive(mpg ~ wt, data = mtcars,
                   tooltip = ~ paste("MPG:", mpg)) |>
  gf_girafe()
```

gf_histogram *Formula interface to geom_histogram()*

Description

Count and density histograms in ggformula.

Usage

```
gf_histogram(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...,
  bins,
  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,
  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).
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	Number of bins. Overridden by binwidth. Defaults to 30.
binwidth	The width of the bins. Can be specified as a numeric value or as a function that takes x after scale transformation as input and returns a single numeric value. When specifying a function along with a grouping structure, the function will be called once per group. The default is to use the number of bins in bins, covering the range of the data. You should always override this value, exploring multiple widths to find the best to illustrate the stories in your data. The bin width of a date variable is the number of days in each time; the bin width of a time variable is the number of seconds.
alpha	Opacity (0 = invisible, 1 = opaque).
color	A color or a formula used for mapping color.
fill	A color for filling, or a formula used for mapping fill.

group	Used for grouping.
linetype	A linetype (numeric or "dashed", "dotted", etc.) or a formula used for mapping linetype.
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, stat	Use to override the default connection between geom_histogram()/geom_freqpoly() and stat_bin(). For more information at overriding these connections, see how the stat and geom arguments work.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.
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. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.
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 ggplot2::facet_wrap() or ggplot2::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

```
x <- rnorm(1000)
gf_histogram(~x, bins = 30)
gf_dhistogram(~x, bins = 30)
gf_dhistogram(~x, binwidth = 0.5, center = 0, color = "black", bins = 30)
gf_dhistogram(~x, binwidth = 0.5, boundary = 0, color = "black", bins = 30)
gf_dhistogram(x ~ ., binwidth = 0.5, boundary = 0, color = "black", bins = 30)
gf_dhistogram(~x, bins = 30) |>
  gf_fitdistr(dist = "dnorm") # see help for gf_fitdistr() for more info.

gf_histogram(~x, fill = ~ (abs(x) <= 2), boundary = 2, binwidth = 0.25)

data(penguins, package = "palmerpenguins")
gf_histogram(~ bill_length_mm | species, data = penguins, binwidth = 0.25)
gf_histogram(~age,
  data = mosaicData::HELPrct, binwidth = 5,
  fill = "skyblue", color = "black"
)
# bins can be adjusted left/right using center or boundary
gf_histogram(~age,
  data = mosaicData::HELPrct,
  binwidth = 5, fill = "skyblue", color = "black", center = 42.5
)
gf_histogram(~age,
  data = mosaicData::HELPrct,
  binwidth = 5, fill = "skyblue", color = "black", boundary = 40
)
gf_histogram(age ~ .,
  data = mosaicData::HELPrct,
  binwidth = 5, fill = "skyblue", color = "black", boundary = 40
)
```

---

gf_histogram_interactive

Interactive histograms

Description

Creates an interactive plot using ggiraph. This function extends `gf_histogram()` with interactive features like tooltips and clickable elements.

Arguments

object	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument.
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.
tooltip	A formula specifying a variable for tooltips, or a character vector.
data_id	A formula or character vector specifying data identifiers for interactive selection.
...	Additional arguments passed to the underlying geom.
alpha, color, size, shape, fill, group, stroke	Aesthetics passed to the geom.
xlab, ylab, title, subtitle, caption	Labels for the plot.
show.legend	Logical. Should this layer be included in the legends?
show.help	Logical. If TRUE, display some minimal help.
inherit	Logical. If TRUE, inherit aesthetics from previous layers.
environment	An environment in which to evaluate the formula.

Value

A gg object that can be displayed with gf_girafe().

Additional interactive features

- onclick: JavaScript code (as character string) executed when clicking elements.
- Additional ggiraph aesthetics may be available depending on the geom.

See Also

gf_histogram(), gf_girafe()

Examples

```
# Interactive histogram with bin information
mtcars |>
  gf_histogram_interactive(
    ~ mpg,
    tooltip = ~ paste0('Min: ', round(after_stat(xmin), 1),
                    '; Max: ', round(after_stat(xmax),1),
                    '; Count: ', after_stat(count)),
    bins = 15) |>
  gf_girafe()
```

gf_hline_interactive *Interactive horizontal lines*

Description

Creates an interactive plot using ggiraph. This function extends gf_hline() with interactive features like tooltips and clickable elements.

Arguments

object	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument.
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.
tooltip	A formula specifying a variable for tooltips, or a character vector.
data_id	A formula or character vector specifying data identifiers for interactive selection.
...	Additional arguments passed to the underlying geom.
alpha, color, size, shape, fill, group, stroke	Aesthetics passed to the geom.
xlab, ylab, title, subtitle, caption	Labels for the plot.
show.legend	Logical. Should this layer be included in the legends?
show.help	Logical. If TRUE, display some minimal help.
inherit	Logical. If TRUE, inherit aesthetics from previous layers.
environment	An environment in which to evaluate the formula.

Value

A gg object that can be displayed with gf_girafe().

Additional interactive features

- onclick: JavaScript code (as character string) executed when clicking elements.
- Additional ggiraph aesthetics may be available depending on the geom.

See Also

gf_hline(), gf_girafe()

Examples

```
# Interactive horizontal reference line
gf_point_interactive(mpg ~ wt, data = mtcars, alpha = 0.7) |>
  gf_hline_interactive(yintercept = ~ mean(mpg),
                       tooltip = ~ paste("Mean MPG:", round(mean(mpg), 1)),
                       color = "red", linetype = "dashed") |>
  gf_girafe()
```

---

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 `ggplot2::facet_wrap()` or `ggplot2::facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of **lattice**.

Evaluation

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

See Also

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

Examples

```
gf_jitter()
# without jitter
gf_point(age ~ sex, alpha = 0.25, data = mosaicData::HELPrct)
# jitter only horizontally
gf_jitter(age ~ sex, alpha = 0.25, data = mosaicData::HELPrct, width = 0.2, height = 0)
# alternative way to get jitter
gf_point(age ~ sex,
  alpha = 0.25, data = mosaicData::HELPrct,
  position = "jitter", width = 0.2, height = 0
)
```

`gf_jitter_interactive` *Interactive jitter plots*

Description

Creates an interactive plot using ggiraph. This function extends `gf_jitter()` with interactive features like tooltips and clickable elements.

Arguments

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

tooltip	A formula specifying a variable for tooltips, or a character vector.
data_id	A formula or character vector specifying data identifiers for interactive selection.
...	Additional arguments passed to the underlying geom.
alpha, color, size, shape, fill, group, stroke	Aesthetics passed to the geom.
xlab, ylab, title, subtitle, caption	Labels for the plot.
show.legend	Logical. Should this layer be included in the legends?
show.help	Logical. If TRUE, display some minimal help.
inherit	Logical. If TRUE, inherit aesthetics from previous layers.
environment	An environment in which to evaluate the formula.

Value

A gg object that can be displayed with gf_girafe().

Additional interactive features

- onclick: JavaScript code (as character string) executed when clicking elements.
- Additional ggiraph aesthetics may be available depending on the geom.

See Also

gf_jitter(), gf_girafe()

Examples

```
# Interactive jittered points
gf_jitter_interactive(mpg ~ factor(cyl), data = mtcars,
                      tooltip = ~ paste0(rownames(mtcars), ": ", mpg, "mpg"),
                      width = 0.2) |>
  gf_girafe()
```

---

gf_labeller_interactive

Create interactive labeller

Description

Create interactive labeller

Usage

```
gf_labeller_interactive(..., .mapping)
```

Arguments

`...`	Arguments of the form `name = ~ expr` are used to create `.mapping` (if `.mapping` is missing). Other arguments (or all arguments if `.mapping` is not missing) are passed through to `ggplot2::labeller()`.
`.mapping`	An aesthetic mapping as could be created with `ggplot2::aes()` or `ggplot2::aes_()`. If missing (the typical use case), `.mapping` is created from the arguments in `...` that have the form `name = ~ expr`.

Value

a labeller

`gf_label_interactive` *Interactive text labels*

Description

Creates an interactive plot using ggiraph. This function extends `gf_label()` with interactive features like tooltips and clickable elements.

Arguments

`object`	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument.
`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.
`tooltip`	A formula specifying a variable for tooltips, or a character vector.
`data_id`	A formula or character vector specifying data identifiers for interactive selection.
`...`	Additional arguments passed to the underlying geom.
`alpha, color, size, shape, fill, group, stroke`	Aesthetics passed to the geom.
`xlab, ylab, title, subtitle, caption`	Labels for the plot.
`show.legend`	Logical. Should this layer be included in the legends?
`show.help`	Logical. If `TRUE`, display some minimal help.
`inherit`	Logical. If `TRUE`, inherit aesthetics from previous layers.
`environment`	An environment in which to evaluate the formula.

Value

A gg object that can be displayed with `gf_girafe()`.

Additional interactive features

- onclick: JavaScript code (as character string) executed when clicking elements.
- Additional ggiraph aesthetics may be available depending on the geom.

See Also

gf_label(), gf_girafe()

Examples

```
mtcars |>
  gf_label_interactive(
    mpg ~ wt,
    label = rownames(mtcars),
    size = 3,
    tooltip = ~ paste("MPG:", mpg)) |>
  gf_girafe()
```

---

gf_labs *Non-layer functions for gf plots*

Description

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

Usage

```
gf_labs(object, ...)

gf_lims(object, ...)

gf_refine(object, ...)
```

Arguments

object	a gg object
...	additional arguments passed through to the similarly named function in **ggplot2**.

Details

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

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",
```

```
  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.
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 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 `ggplot2::facet_wrap()` or `ggplot2::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,
  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 = 1,
  size,
  fun.y = NULL,
  fun.ymax = NULL,
  fun.ymin = NULL,
  fun.args = list(),
  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)`).
`...`	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 — Set or map color.

group — Use to set or map group.

linetype, linewidth — Set or map style of the line.

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 for this layer. When using a stat_*() function to construct a layer, the geom argument can be used to override the default coupling between stats and geoms. The geom argument accepts the following:

- A Geom ggproto subclass, for example GeomPoint.
- A string naming the geom. To give the geom as a string, strip the function name of the geom_ prefix. For example, to use geom_point(), give the geom as "point".
- For more information and other ways to specify the geom, see the layer geom documentation.

stat — The statistical transformation to use on the data for this layer. When using a geom_*() function to construct a layer, the stat argument can be used to override the default coupling between geoms and stats. The stat argument accepts the following:

- A Stat ggproto subclass, for example StatCount.
- A string naming the stat. To give the stat as a string, strip the function name of the stat_ prefix. For example, to use stat_count(), give the stat as "count".
- For more information and other ways to specify the stat, see the layer stat documentation.

position — A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following:

- The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position.
- A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter".
- For more information and other ways to specify the position, see the layer position documentation.

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. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.
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()).
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(
      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_linerange_interactive` *Interactive linerange plots*

Description

Creates an interactive plot using ggiraph. This function extends `gf_linerange()` with interactive features like tooltips and clickable elements.

Arguments

`object`	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument.
`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.
tooltip	A formula specifying a variable for tooltips, or a character vector.
data_id	A formula or character vector specifying data identifiers for interactive selection.
...	Additional arguments passed to the underlying geom.
alpha, color, size, shape, fill, group, stroke	Aesthetics passed to the geom.
xlab, ylab, title, subtitle, caption	Labels for the plot.
show.legend	Logical. Should this layer be included in the legends?
show.help	Logical. If TRUE, display some minimal help.
inherit	Logical. If TRUE, inherit aesthetics from previous layers.
environment	An environment in which to evaluate the formula.

Value

A gg object that can be displayed with gf_girafe().

Additional interactive features

- onclick: JavaScript code (as character string) executed when clicking elements.
- Additional ggiraph aesthetics may be available depending on the geom.

See Also

gf_linerange(), gf_girafe()

Examples

```
diamonds |>
  dplyr::filter(carat < 1.1, carat > 0.9) |>
  dplyr::group_by(color, cut) |>
  dplyr::summarise(
    median_price = median(price) |> round(),
    lower = quantile(price, 0.25) |> round(),
    upper = quantile(price, 0.75) |> round(),
    iqr = upper - lower
  ) |>
  gf_linerange_interactive(
    cut ~ lower + upper | color,
    color = ~ cut,
    tooltip = ~ paste0(
      "75th percentile: ", upper,
      "\nmedian: ", median_price,
      "\n25th percentile: ", lower
      )
  ) |>
  gf_girafe()
```

gf_line_interactive *Interactive line plots*

Description

Creates an interactive plot using ggiraph. This function extends gf_line() with interactive features like tooltips and clickable elements.

Arguments

object	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument.
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.
tooltip	A formula specifying a variable for tooltips, or a character vector.
data_id	A formula or character vector specifying data identifiers for interactive selection.
...	Additional arguments passed to the underlying geom.
alpha, color, size, shape, fill, group, stroke	Aesthetics passed to the geom.
xlab, ylab, title, subtitle, caption	Labels for the plot.
show.legend	Logical. Should this layer be included in the legends?
show.help	Logical. If TRUE, display some minimal help.
inherit	Logical. If TRUE, inherit aesthetics from previous layers.
environment	An environment in which to evaluate the formula.

Value

A gg object that can be displayed with gf_girafe().

Additional interactive features

- onclick: JavaScript code (as character string) executed when clicking elements.
- Additional ggiraph aesthetics may be available depending on the geom.

See Also

gf_line(), gf_girafe()

Examples

```
if (require(mosaicData)) {
  Weather |>
  gf_line_interactive(
    high_temp ~ date,
    color = ~city,
    show.legend = FALSE,
    tooltip = ~city,
    data_id = ~city
  ) |>
  gf_girafe(
    width = 8, height = 3,
    options = list(
      opts_hover_inv(css = "opacity:0.4;"),
      opts_hover(css = "stroke-width:2;", nearest_distance = 40),
      opts_tooltip(use_cursor_pos = FALSE, offx = 0, offy = -10)
    )
  )
}
```

`gf_path_interactive` *Interactive path plots*

Description

Creates an interactive plot using ggiraph. This function extends `gf_path()` with interactive features like tooltips and clickable elements.

Arguments

`object`	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument.
`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.
`tooltip`	A formula specifying a variable for tooltips, or a character vector.
`data_id`	A formula or character vector specifying data identifiers for interactive selection.
`...`	Additional arguments passed to the underlying geom.
`alpha`, `color`, `size`, `shape`, `fill`, `group`, `stroke`	Aesthetics passed to the geom.
`xlab`, `ylab`, `title`, `subtitle`, `caption`	Labels for the plot.
`show.legend`	Logical. Should this layer be included in the legends?
`show.help`	Logical. If `TRUE`, display some minimal help.
`inherit`	Logical. If `TRUE`, inherit aesthetics from previous layers.
`environment`	An environment in which to evaluate the formula.

Value

A gg object that can be displayed with `gf_girafe()`.

Additional interactive features

- `onclick`: JavaScript code (as character string) executed when clicking elements.
- Additional ggiraph aesthetics may be available depending on the geom.

See Also

`gf_path()`, `gf_girafe()`

Examples

```
gf_path_interactive(mpg ~ wt, data = mtcars,
                    tooltip = ~ paste("MPG:", mpg)) |>
  gf_girafe()
```

---

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: `aesthetic = ~ expression`. See examples.

Value

a gg object

Examples

```
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.

alpha	Opacity (0 = invisible, 1 = opaque).
color	A color or a formula used for mapping color.
size	A numeric size or a formula used for mapping size.
shape	An integer or letter shape or a formula used for mapping shape.
fill	A color for filling, or a formula used for mapping fill.
group	Used for grouping.
stroke	A numeric size of the border or a formula used to map stroke.
xlab	Label for x-axis. See also gf_labs().
ylab	Label for y-axis. See also gf_labs().
title, subtitle, caption	Title, sub-title, and caption for the plot. See also gf_labs().
geom	A character string naming the geom used to make the layer.
stat	A character string naming the stat used to make the layer.
position	Either a character string naming the position function used for the layer or a position object returned from a call to a position function.
show.legend	A logical indicating whether this layer should be included in the legends. NA, the default, includes layer in the legends if any of the attributes of the layer are mapped.
show.help	If TRUE, display some minimal help.
inherit	A logical indicating whether default attributes are inherited.
environment	An environment in which to look for variables not found in data.

Value

a gg object

Specifying plot attributes

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

In formulas of the form A | B, B will be used to form facets using ggplot2::facet_wrap() or ggplot2::facet_grid(). This provides an alternative to gf_facet_wrap() and gf_facet_grid() that is terser and may feel more familiar to users of **lattice**.

Evaluation

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

See Also

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

Examples

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

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

---

gf_pointrange_interactive

Interactive pointrange plots

Description

Creates an interactive plot using ggiraph. This function extends `gf_pointrange()` with interactive features like tooltips and clickable elements.

Arguments

`object`	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument.
`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.
`tooltip`	A formula specifying a variable for tooltips, or a character vector.
`data_id`	A formula or character vector specifying data identifiers for interactive selection.
`...`	Additional arguments passed to the underlying geom.
`alpha`, `color`, `size`, `shape`, `fill`, `group`, `stroke`	Aesthetics passed to the geom.

xlab, ylab, title, subtitle, caption	Labels for the plot.
show.legend	Logical. Should this layer be included in the legends?
show.help	Logical. If TRUE, display some minimal help.
inherit	Logical. If TRUE, inherit aesthetics from previous layers.
environment	An environment in which to evaluate the formula.

Value

A gg object that can be displayed with gf_girafe().

Additional interactive features

- onclick: JavaScript code (as character string) executed when clicking elements.
- Additional ggiraph aesthetics may be available depending on the geom.

See Also

gf_pointrange(), gf_girafe()

Examples

```
diamonds |>
  dplyr::filter(carat < 1.1, carat > 0.9) |>
  dplyr::group_by(color, cut) |>
  dplyr::summarise(
    median_price = median(price) |> round(),
    lower = quantile(price, 0.25) |> round(),
    upper = quantile(price, 0.75) |> round(),
    iqr = upper - lower
  ) |>
  gf_pointrange_interactive(
    cut ~ median_price + lower + upper | color,
    color = ~ cut,
    tooltip = ~ paste0(
      "75th percentile: ", upper,
      "\nmedian: ", median_price,
      "\n25th percentile: ", lower
      )
  ) |>
  gf_girafe()
```

`gf_point_interactive` *Interactive scatter plots*

Description

Creates an interactive plot using ggiraph. This function extends `gf_point()` with interactive features like tooltips and clickable elements.

Arguments

`object`	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument.
`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.
`tooltip`	A formula specifying a variable for tooltips, or a character vector.
`data_id`	A formula or character vector specifying data identifiers for interactive selection.
`...`	Additional arguments passed to the underlying geom.
`alpha`, `color`, `size`, `shape`, `fill`, `group`, `stroke`	Aesthetics passed to the geom.
`xlab`, `ylab`, `title`, `subtitle`, `caption`	Labels for the plot.
`show.legend`	Logical. Should this layer be included in the legends?
`show.help`	Logical. If `TRUE`, display some minimal help.
`inherit`	Logical. If `TRUE`, inherit aesthetics from previous layers.
`environment`	An environment in which to evaluate the formula.

Value

A gg object that can be displayed with `gf_girafe()`.

Additional interactive features

- `onclick`: JavaScript code (as character string) executed when clicking elements.
- Additional ggiraph aesthetics may be available depending on the geom.

See Also

`gf_point()`, `gf_girafe()`

Examples

```
gf_point_interactive(mpg ~ wt, data = mtcars,
                     tooltip = ~ paste("Model:", rownames(mtcars))) |>
  gf_girafe()

# With color mapping and data_id for selection
gf_point_interactive(mpg ~ wt, data = mtcars,
                     color = ~ factor(cyl),
                     tooltip = ~ paste(rownames(mtcars), ":", mpg, "mpg"),
                     data_id = ~ rownames(mtcars)) |>
  gf_girafe()
```

---

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

Arguments

object	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
gformula	A formula with shape y ~ 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.

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 ggplot2::facet_wrap() or ggplot2::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_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_polygon_interactive

Interactive polygon plots

Description

Creates an interactive plot using ggiraph. This function extends `gf_polygon()` with interactive features like tooltips and clickable elements.

Arguments

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

tooltip	A formula specifying a variable for tooltips, or a character vector.
data_id	A formula or character vector specifying data identifiers for interactive selection.
...	Additional arguments passed to the underlying geom.
alpha, color, size, shape, fill, group, stroke	Aesthetics passed to the geom.
xlab, ylab, title, subtitle, caption	Labels for the plot.
show.legend	Logical. Should this layer be included in the legends?
show.help	Logical. If TRUE, display some minimal help.
inherit	Logical. If TRUE, inherit aesthetics from previous layers.
environment	An environment in which to evaluate the formula.

Value

A gg object that can be displayed with gf_girafe().

Additional interactive features

- onclick: JavaScript code (as character string) executed when clicking elements.
- Additional ggiraph aesthetics may be available depending on the geom.

See Also

gf_polygon(), gf_girafe()

Examples

```
gf_polygon_interactive(mpg ~ wt, data = mtcars,
                       tooltip = ~ paste("MPG:", mpg)) |>
  gf_girafe()
```

---

gf_qq *Formula interface to geom_qq()*

Description

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

Usage

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

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

gf_qqstep(
```

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

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(). For more information at overriding these connections, see how the stat and geom arguments work.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.
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. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.
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 ggplot2::facet_wrap() or ggplot2::facet_grid(). This provides an alternative to gf_facet_wrap() and gf_facet_grid() that is terser and may feel more familiar to users of **lattice**.

Evaluation

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

See Also

ggplot2::geom_qq()

Examples

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

---

gf_quantile *Formula interface to geom_quantile()*

Description

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

Usage

```
gf_quantile(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...,
  alpha,
  color,
  group,
  linetype,
  linewidth,
  lineend = "butt",
  linejoin = "round",
  linemitre = 1,
  quantiles,
  formula,
  method,
  method.args,
  xlab,
  ylab,
  title,
  subtitle,
  caption,
  geom = "quantile",
  stat = "quantile",
  position = "identity",
```

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

Arguments

object	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
gformula	A formula with shape y ~ x. Faceting can be achieved by including | in the formula.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
...	Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with attribute = value, (b) ggplot2 aesthetics to be mapped with attribute = ~ expression, or (c) attributes of the layer as a whole, which are set with attribute = value.
alpha	Opacity (0 = invisible, 1 = opaque).
color	A color or a formula used for mapping color.
group	Used for grouping.
linetype	A linetype (numeric or "dashed", "dotted", etc.) or a formula used for mapping linetype.
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).
quantiles	conditional quantiles of y to calculate and display
formula	formula relating y variables to x variables
method	Quantile regression method to use. Available options are "rq" (for quantreg::rq()) and "rqss" (for quantreg::rqss()).
method.args	List of additional arguments passed on to the modelling function defined by method.
xlab	Label for x-axis. See also gf_labs().
ylab	Label for y-axis. See also gf_labs().

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

`geom`, `stat`
: Use to override the default connection between `geom_quantile()` and `stat_quantile()`. For more information about overriding these connections, see how the stat and geom arguments work.

`position`
: A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The `position` argument accepts the following:

  - The result of calling a position function, such as `position_jitter()`. This method allows for passing extra arguments to the position.
  - A string naming the position adjustment. To give the position as a string, strip the function name of the `position_` prefix. For example, to use `position_jitter()`, give the position as `"jitter"`.
  - For more information and other ways to specify the position, see the layer position documentation.

`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. To include legend keys for all levels, even when no data exists, use `TRUE`. If `NA`, all levels are shown in legend, but unobserved levels are omitted.

`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 `ggplot2::facet_wrap()` or `ggplot2::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

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

`gf_quantile_interactive` *Interactive quantile plots*

Description

Creates an interactive plot using ggiraph. This function extends `gf_quantile()` with interactive features like tooltips and clickable elements.

Arguments

`object`	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument.
`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.
`tooltip`	A formula specifying a variable for tooltips, or a character vector.
`data_id`	A formula or character vector specifying data identifiers for interactive selection.
`...`	Additional arguments passed to the underlying geom.
`alpha`, `color`, `size`, `shape`, `fill`, `group`, `stroke`	Aesthetics passed to the geom.
`xlab`, `ylab`, `title`, `subtitle`, `caption`	Labels for the plot.
`show.legend`	Logical. Should this layer be included in the legends?
`show.help`	Logical. If `TRUE`, display some minimal help.
`inherit`	Logical. If `TRUE`, inherit aesthetics from previous layers.
`environment`	An environment in which to evaluate the formula.

Value

A gg object that can be displayed with `gf_girafe()`.

Additional interactive features

- `onclick`: JavaScript code (as character string) executed when clicking elements.
- Additional ggiraph aesthetics may be available depending on the geom.

See Also

`gf_quantile()`, `gf_girafe()`

Examples

```
gf_quantile_interactive(mpg ~ wt, data = mtcars,
                        tooltip = ~ paste("MPG:", mpg)) |>
  gf_girafe()
```

---

gf_raster *Formula interface to geom_raster()*

Description

Formula interface to geom_raster()

Usage

```
gf_raster(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...,
  alpha,
  color,
  fill,
  group,
  linetype,
  linewidth,
  hjust = 0.5,
  vjust = 0.5,
  interpolate = FALSE,
  xlab,
  ylab,
  title,
  subtitle,
  caption,
  geom = "raster",
  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 or fill ~ x + y
data	A data frame with the variables to be plotted.
...	Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with attribute = value, (b) ggplot2 aesthetics to be mapped with attribute = ~ expression, or (c) attributes of the layer as a whole, which are set with attribute = value.
alpha	Opacity (0 = invisible, 1 = opaque).
color	A color or a formula used for mapping color.
fill	A color for filling, or a formula used for mapping fill.
group	Used for grouping.
linetype	A linetype (numeric or "dashed", "dotted", etc.) or a formula used for mapping linetype.
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 ggplot2::facet_wrap() or ggplot2::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_raster_interactive *Interactive raster plots*

Description

Creates an interactive plot using ggiraph. This function extends `gf_raster()` with interactive features like tooltips and clickable elements.

Arguments

`object`	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument.
`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.
`tooltip`	A formula specifying a variable for tooltips, or a character vector.
`data_id`	A formula or character vector specifying data identifiers for interactive selection.
`...`	Additional arguments passed to the underlying geom.
`alpha`, `color`, `size`, `shape`, `fill`, `group`, `stroke`	Aesthetics passed to the geom.
`xlab`, `ylab`, `title`, `subtitle`, `caption`	Labels for the plot.
`show.legend`	Logical. Should this layer be included in the legends?
`show.help`	Logical. If `TRUE`, display some minimal help.
`inherit`	Logical. If `TRUE`, inherit aesthetics from previous layers.
`environment`	An environment in which to evaluate the formula.

Value

A gg object that can be displayed with gf_girafe().

Additional interactive features

- onclick: JavaScript code (as character string) executed when clicking elements.
- Additional ggiraph aesthetics may be available depending on the geom.

See Also

gf_raster(), gf_girafe()

Examples

```
gf_raster_interactive(mpg ~ wt, data = mtcars,
                      tooltip = ~ paste("MPG:", mpg)) |>
  gf_girafe()
```

---

gf_rect *Formula interface to geom_rect()*

Description

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

Usage

```
gf_rect(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...,
  alpha,
  color,
  fill,
  group,
  linetype,
  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 `ggplot2::facet_wrap()` or `ggplot2::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_rect_interactive *Interactive rect plots*

Description

Creates an interactive plot using ggiraph. This function extends `gf_rect()` with interactive features like tooltips and clickable elements.

Arguments

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

tooltip	A formula specifying a variable for tooltips, or a character vector.
data_id	A formula or character vector specifying data identifiers for interactive selection.
...	Additional arguments passed to the underlying geom.
alpha, color, size, shape, fill, group, stroke	Aesthetics passed to the geom.
xlab, ylab, title, subtitle, caption	Labels for the plot.
show.legend	Logical. Should this layer be included in the legends?
show.help	Logical. If TRUE, display some minimal help.
inherit	Logical. If TRUE, inherit aesthetics from previous layers.
environment	An environment in which to evaluate the formula.

Value

A gg object that can be displayed with gf_girafe().

Additional interactive features

- onclick: JavaScript code (as character string) executed when clicking elements.
- Additional ggiraph aesthetics may be available depending on the geom.

See Also

gf_rect(), gf_girafe()

Examples

```
rect_data <-
  data.frame(
    x1 = c(1, 3, 1, 5, 4),
    x2 = c(2, 4, 3, 6, 6),
    y1 = c(1, 1, 4, 1, 3),
    y2 = c(2, 2, 5, 3, 5),
    t = c('a', 'a', 'a', 'b', 'b'),
    r = c(1, 2, 3, 4, 5),
    tooltip = c("ID 1", "ID 2", "ID 3", "ID 4", "ID 5"),
    uid = c("ID 1", "ID 2", "ID 3", "ID 4", "ID 5"),
    oc = rep("alert(this.getAttribute(\"data-id\"))", 5)
  )

p <- rect_data |>
  gf_rect_interactive(
    y1 + y2 ~ x1 + x2,
    fill = t,
    tooltip = ~ tooltip,
    onclick = ~ oc,
    data_id = ~ uid,
    color = "black",
    alpha = 0.5,
```

```
    linejoin = "bevel",
    lineend = "round"
  ) |>
  gf_text(
    (y1 + (y2 - y1) / 2) ~ (x1 + (x2 - x1) / 2),
    label = ~ r,
    size = 4
    )

if (interactive()) {
 p |> gf_girafe()
}
```

`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

```
# 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

```
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 for this layer. When using a stat_*() function to construct a layer, the geom argument can be used to override the default coupling between stats and geoms. The geom argument accepts the following: • A Geom ggproto subclass, for example GeomPoint. • A string naming the geom. To give the geom as a string, strip the function name of the geom_ prefix. For example, to use geom_point(), give the geom as "point". • For more information and other ways to specify the geom, see the layer geom documentation.
stat	The statistical transformation to use on the data for this layer. When using a geom_*() function to construct a layer, the stat argument can be used to override the default coupling between geoms and stats. The stat argument accepts the following:

- A `Stat` ggproto subclass, for example `StatCount`.
- A string naming the stat. To give the stat as a string, strip the function name of the `stat_` prefix. For example, to use `stat_count()`, give the stat as `"count"`.
- For more information and other ways to specify the stat, see the layer stat documentation.

`position` A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The `position` argument accepts the following:

- The result of calling a position function, such as `position_jitter()`. This method allows for passing extra arguments to the position.
- A string naming the position adjustment. To give the position as a string, strip the function name of the `position_` prefix. For example, to use `position_jitter()`, give the position as `"jitter"`.
- For more information and other ways to specify the position, see the layer position documentation.

`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. To include legend keys for all levels, even when no data exists, use `TRUE`. If `NA`, all levels are shown in legend, but unobserved levels are omitted.

`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_ribbon_interactive *Interactive ribbon plots*

Description

Creates an interactive plot using ggiraph. This function extends gf_ribbon() with interactive features like tooltips and clickable elements.

Arguments

object	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument.
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.
tooltip	A formula specifying a variable for tooltips, or a character vector.
data_id	A formula or character vector specifying data identifiers for interactive selection.
...	Additional arguments passed to the underlying geom.
alpha, color, size, shape, fill, group, stroke	Aesthetics passed to the geom.
xlab, ylab, title, subtitle, caption	Labels for the plot.
show.legend	Logical. Should this layer be included in the legends?
show.help	Logical. If TRUE, display some minimal help.
inherit	Logical. If TRUE, inherit aesthetics from previous layers.
environment	An environment in which to evaluate the formula.

Value

A gg object that can be displayed with gf_girafe().

Additional interactive features

- onclick: JavaScript code (as character string) executed when clicking elements.
- Additional ggiraph aesthetics may be available depending on the geom.

See Also

gf_ribbon(), gf_girafe()

Examples

```
Huron <-
  data.frame(
    year = 1875:1972,
    level = as.vector(LakeHuron)
  )

Huron |>
  gf_ribbon_interactive(
    (level - 1) + (level + 1) ~ year,
    tooltip = ~ "This is the ribbon.",
    fill = "skyblue",
    data_id = "id:ribbon"
    ) |>
  gf_line_interactive(
    level  ~ year,
    tooltip = ~ "This is the line.",
    data_id = "id:line"
  ) |>
  gf_girafe()
```

---

gf_ridgeline *Formula interface to ggridges plots*

Description

Formula interface to ggridges plots

Usage

```
gf_ridgeline(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...,
  height,
  scale = 1,
  min_height = 0,
  color,
  fill,
  alpha,
  group,
  linetype,
  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,
  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",
  stat = "density_ridges",
  position = "points_sina",
  show.legend = NA,
  show.help = NULL,
  inherit = TRUE,
  environment = parent.frame()
)
```

Arguments

object	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
gformula	A formula with shape ~ x. Faceting can be achieved by including | in the formula.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
...	Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with attribute = value, (b) ggplot2 aesthetics to be mapped with attribute = ~ expression, or (c) attributes of the layer as a whole, which are set with attribute = value.
height	The height of each ridgeline at the respective x value. Automatically calculated and provided by ggridges::stat_density_ridges() if the default stat is not changed.
scale	A scaling factor to scale the height of the ridgelines relative to the spacing between them. A value of 1 indicates that the maximum point of any ridgeline touches the baseline right above, assuming even spacing between baselines.
min_height	A height cutoff on the drawn ridgelines. All values that fall below this cutoff will be removed. The main purpose of this cutoff is to remove long tails right at the baseline level, but other uses are possible. The cutoff is applied before any height scaling is applied via the scale aesthetic. Default is 0, so negative values are removed.
color	A color or a formula used for mapping color.
fill	A color for filling, or a formula used for mapping fill.
alpha	Opacity (0 = invisible, 1 = opaque).
group	Used for grouping.
linetype	A linetype (numeric or "dashed", "dotted", etc.) or a formula used for mapping linetype.
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 ggridges::geom_ridgeline().
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(). For more information about overriding these connections, see how the stat and geom arguments work.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.
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. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.
show.help	If TRUE, display some minimal help.
inherit	A logical indicating whether default attributes are inherited.
environment	An environment in which to look for variables not found in data.
rel_min_height	Lines with heights below this cutoff will be removed. The cutoff is measured relative to the overall maximum, so rel_min_height = 0.01 would remove everything. Default is 0, so nothing is removed.
panel_scaling	If TRUE, the default, relative scaling is calculated separately for each panel. If FALSE, relative scaling is calculated globally.
gradient_lwd	A parameter to needed to remove rendering artifacts inside the rendered gradients. Should ideally be 0, but often needs to be around 0.5 or higher.

Details

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

See Also

ggridges::geom_density_ridges()

ggridges::geom_ridgeline()

ggridges::geom_density_ridges_gradient()

Examples

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

```
  gf_density_ridges(clarity ~ price | cut,
    scale = 2, fill = ~ clarity, alpha = 0.6, show.legend = FALSE) |>
  gf_theme(theme_ridges()) |>
  gf_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,
  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,
  geom = "rug",
  stat = "identity",
  position = "identity",
  show.legend = NA,
  show.help = NULL,
  inherit = FALSE,
  environment = parent.frame()
)
```

Arguments

object	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See 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. When using a geom_*() function to construct a layer, the stat argument can be used to override the default coupling between geoms and stats. The stat argument accepts the following:
  - A Stat ggproto subclass, for example StatCount.
  - A string naming the stat. To give the stat as a string, strip the function name of the stat_ prefix. For example, to use stat_count(), give the stat as "count".
  - For more information and other ways to specify the stat, see the layer stat documentation.

position
: A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following:
  - The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position.
  - A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter".
  - For more information and other ways to specify the position, see the layer position documentation.

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. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.

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 ggplot2::facet_wrap() or ggplot2::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, bins = 25) |>
  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, bins = 25) |>
  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_dhistogram(~bill_depth_mm, data = penguins) |>
 gf_rug(0 ~ bill_depth_mm, data = penguins, color = "green", sides = "b", position = "jitter")

# the choice of y value can affect how the plot looks.
gf_dhistogram(~bill_depth_mm, data = penguins) |>
```

```
gf_rug(0.5 ~ bill_depth_mm, data = penguins, color = "green", sides = "b", position = "jitter")
```

---

gf_segment *Formula interface to geom_segment()*

Description

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

Usage

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

Arguments

object	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
gformula	A formula with shape y + yend ~ 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. ~ 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.

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. When using a geom_*() function to construct a layer, the stat argument can be used to override the default coupling between geoms and stats. The stat argument accepts the following:

- A Stat ggproto subclass, for example StatCount.
- A string naming the stat. To give the stat as a string, strip the function name of the stat_ prefix. For example, to use stat_count(), give the stat as "count".
- For more information and other ways to specify the stat, see the layer stat documentation.

position — A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following:

- The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position.
- A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter".

- For more information and other ways to specify the position, see the layer position documentation.

`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. To include legend keys for all levels, even when no data exists, use `TRUE`. If `NA`, all levels are shown in legend, but unobserved levels are omitted.

`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 `ggplot2::facet_wrap()` or `ggplot2::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")
```

gf_sf *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.
data	A data frame with the variables to be plotted.
...	Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with attribute = value, (b) ggplot2 aesthetics to be mapped with attribute = ~ expression, (c) attributes of the layer as a whole, which are set with attribute = value, or (d) arguments for the geom, stat, or position function.
alpha	Opacity (0 = invisible, 1 = opaque).

color	A color or a formula used for mapping color.
fill	A color for filling, or a formula used for mapping fill.
group	Used for grouping.
linetype	A linetype (numeric or "dashed", "dotted", etc.) or a formula used for mapping linetype.
linewidth	A numerical line width or a formula used for mapping linewidth.
geometry	A column of class sfc containing simple features data. (Another option is that data may contain a column named geometry.) geometry is never inherited.
xlab	Label for x-axis. See also gf_labs().
ylab	Label for y-axis. See also gf_labs().
title, subtitle, caption	Title, sub-title, and caption for the plot. See also gf_labs().
stat	A character string naming the stat used to make the layer.
position	Either a character string naming the position function used for the layer or a position object returned from a call to a position function.
show.legend	A logical indicating whether this layer should be included in the legends. NA, the default, includes layer in the legends if any of the attributes of the layer are mapped.
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 ggplot2::facet_wrap() or ggplot2::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

```
if (requireNamespace('maps', quietly = TRUE)) {
  world1 <- sf::st_as_sf(maps::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)
}
```

---

gf_sf_interactive *Interactive sf plots*

Description

Creates an interactive plot using ggiraph. This function extends `gf_sf()` with interactive features like tooltips and clickable elements.

Arguments

`object`	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument.
`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.
`tooltip`	A formula specifying a variable for tooltips, or a character vector.
`data_id`	A formula or character vector specifying data identifiers for interactive selection.
`...`	Additional arguments passed to the underlying geom.
`alpha`, `color`, `size`, `shape`, `fill`, `group`, `stroke`	Aesthetics passed to the geom.
`xlab`, `ylab`, `title`, `subtitle`, `caption`	Labels for the plot.
`show.legend`	Logical. Should this layer be included in the legends?
`show.help`	Logical. If `TRUE`, display some minimal help.
`inherit`	Logical. If `TRUE`, inherit aesthetics from previous layers.
`environment`	An environment in which to evaluate the formula.

Value

A gg object that can be displayed with `gf_girafe()`.

Additional interactive features

- onclick: JavaScript code (as character string) executed when clicking elements.
- Additional ggiraph aesthetics may be available depending on the geom.

See Also

gf_sf(), gf_girafe()

---

gf_sina *Formula interface to geom_sina()*

Description

The sina plot is a data visualization chart suitable for plotting any single variable in a multiclass dataset. It is an enhanced jitter strip chart, where the width of the jitter is controlled by the density distribution of the data within each class.

Usage

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

Arguments

object	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
gformula	A formula with shape y ~ 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()'s params argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the position argument, or aesthetics that are required can *not* be passed through .... Unknown arguments that are not part of the 4 categories below are ignored. • Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, colour = "red" or linewidth = 3. The geom's documentation has an **Aesthetics** section that lists the available options. The 'required' aesthetics cannot be passed on to the params. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data. • When constructing a layer using a stat_*() function, the ... argument can be used to pass on parameters to the geom part of the layer. An example of this is stat_density(geom = "area", outline.type = "both"). The geom's documentation lists which parameters it can accept. • Inversely, when constructing a layer using a geom_*() function, the ... argument can be used to pass on parameters to the stat part of the layer. An example of this is geom_area(stat = "density", adjust = 0.5). The stat's documentation lists which parameters it can accept. • The key_glyph argument of layer() may also be passed on through .... This can be one of the functions described as key glyphs, to change the display of the layer in the legend.
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 for this layer. When using a stat_*() function to construct a layer, the geom argument can be used to override the default coupling between stats and geoms. The geom argument accepts the following:

- A Geom ggproto subclass, for example GeomPoint.
- A string naming the geom. To give the geom as a string, strip the function name of the geom_ prefix. For example, to use geom_point(), give the geom as "point".
- For more information and other ways to specify the geom, see the layer geom documentation.

stat
: The statistical transformation to use on the data for this layer. When using a geom_*() function to construct a layer, the stat argument can be used the override the default coupling between geoms and stats. The stat argument accepts the following:

- A Stat ggproto subclass, for example StatCount.
- A string naming the stat. To give the stat as a string, strip the function name of the stat_ prefix. For example, to use stat_count(), give the stat as "count".
- For more information and other ways to specify the stat, see the layer stat documentation.

position
: A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following:

- The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position.
- A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter".
- For more information and other ways to specify the position, see the layer position documentation.

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 `ggplot2::facet_wrap()` or `ggplot2::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::HELPrct)

## End(Not run)
```

---

gf_smooth *Formula interface to geom_smooth()*

Description

LOESS and linear model smoothers in `ggformula`.

Usage

```
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()
)

gf_lm(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...,
  alpha = 0.3,
  linewidth = 1,
  linetype,
  lm.args = list(),
  interval = "none",
  level = 0.95,
  fullrange = TRUE,
  xlab,
  ylab,
  title,
  subtitle,
  caption,
  geom = "lm",
  stat = "lm",
  position = "identity",
  show.legend = NA,
  show.help = NULL,
  inherit = TRUE,
  environment = parent.frame()
)
```

Arguments

`object`	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
`gformula`	A formula with shape `y ~ 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 ~ s(x, bs = "cs") with method = "REML". Somewhat anecdotally, loess gives a better appearance, but is $O(N^2)$ in memory, so does not work for larger datasets.

If you have fewer than 1,000 observations but want to use the same gam() model that method = NULL would use, then set method = "gam", formula = y ~ s(x, bs = "cs").

formula — Formula to use in smoothing function, eg. y ~ x, y ~ poly(x, 2), y ~ log(x). NULL by default, in which case method = NULL implies formula = y ~ x when there are fewer than 1,000 observations and formula = y ~ s(x, bs = "cs") otherwise.

se — Display confidence band 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 band 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().

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

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

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

show.legend	A logical indicating whether this layer should be included in the legends. NA, the default, includes layer in the legends if any of the attributes of the layer are mapped.
show.help	If TRUE, display some minimal help.
inherit	A logical indicating whether default attributes are inherited.
environment	An environment in which to look for variables not found in data.
alpha	Opacity (0 = invisible, 1 = opaque).
linewidth	A numerical line width or a formula used for mapping linewidth.
linetype	A linetype (numeric or "dashed", "dotted", etc.) or a formula used for mapping linetype.
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 ggplot2::facet_wrap() or ggplot2::facet_grid(). This provides an alternative to gf_facet_wrap() and gf_facet_grid() that is terser and may feel more familiar to users of **lattice**.

Evaluation

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

See Also

ggplot2::geom_smooth(), gf_spline()

Examples

```
gf_smooth()
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()

  clotting <- 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_smooth_interactive` *Interactive smoothed conditional means*

Description

Creates an interactive plot using ggiraph. This function extends `gf_smooth()` with interactive features like tooltips and clickable elements.

Arguments

`object`	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument.
`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.
`tooltip`	A formula specifying a variable for tooltips, or a character vector.
`data_id`	A formula or character vector specifying data identifiers for interactive selection.
`...`	Additional arguments passed to the underlying geom.
`alpha`, `color`, `size`, `shape`, `fill`, `group`, `stroke`	Aesthetics passed to the geom.
`xlab`, `ylab`, `title`, `subtitle`, `caption`	Labels for the plot.
`show.legend`	Logical. Should this layer be included in the legends?
`show.help`	Logical. If `TRUE`, display some minimal help.
`inherit`	Logical. If `TRUE`, inherit aesthetics from previous layers.
`environment`	An environment in which to evaluate the formula.

Value

A gg object that can be displayed with `gf_girafe()`.

Additional interactive features

- `onclick`: JavaScript code (as character string) executed when clicking elements.
- Additional ggiraph aesthetics may be available depending on the geom.

See Also

`gf_smooth()`, `gf_girafe()`

Examples

```
# Interactive smooth line with confidence band
mtcars |>
  gf_point_interactive(mpg ~ wt, alpha = 0.5) |>
 gf_smooth_interactive(tooltip = ~ "loess line with confidence band", se = TRUE, alpha = 0.5) |>
  gf_girafe()
```

---

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 ~ 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 `ggplot2::facet_wrap()` or `ggplot2::facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of **lattice**.

Evaluation

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

See Also

`geom_spline()`, `gf_smooth()`, `gf_lm()`

Examples

```
gf_spline(births ~ date, color = ~wday, data = mosaicData::Births78)
gf_spline(births ~ date, color = ~wday, data = mosaicData::Births78, df = 20)
gf_spline(births ~ date, color = ~wday, data = mosaicData::Births78, df = 4)
```

---

gf_spoke *Formula interface to geom_spoke()*

Description

This is a polar parameterisation of geom_segment. It is useful when you have variables that describe direction and distance.

Usage

```
gf_spoke(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...,
  angle,
  radius,
  alpha,
  color,
  group,
  linetype,
  linewidth,
```

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

Arguments

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

- A Stat ggproto subclass, for example StatCount.
- A string naming the stat. To give the stat as a string, strip the function name of the stat_ prefix. For example, to use stat_count(), give the stat as "count".
- For more information and other ways to specify the stat, see the layer stat documentation.

position — A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following:

- The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position.
- A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter".
- For more information and other ways to specify the position, see the layer position documentation.

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. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.

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 ggplot2::facet_wrap() or ggplot2::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

```
SomeData <- expand.grid(x = 1:10, y = 1:10)
SomeData$angle <- runif(100, 0, 2 * pi)
SomeData$speed <- runif(100, 0, sqrt(0.1 * SomeData$x))

gf_point(y ~ x, data = SomeData) |>
  gf_spoke(y ~ x, angle = ~angle, radius = 0.5)

gf_point(y ~ x, data = SomeData) |>
  gf_spoke(y ~ x, angle = ~angle, radius = ~speed)
```

`gf_spoke_interactive` *Interactive spoke plots*

Description

Creates an interactive plot using ggiraph. This function extends `gf_spoke()` with interactive features like tooltips and clickable elements.

Arguments

`object`	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument.
`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.
`tooltip`	A formula specifying a variable for tooltips, or a character vector.
`data_id`	A formula or character vector specifying data identifiers for interactive selection.
`...`	Additional arguments passed to the underlying geom.
`alpha`, `color`, `size`, `shape`, `fill`, `group`, `stroke`	Aesthetics passed to the geom.
`xlab`, `ylab`, `title`, `subtitle`, `caption`	Labels for the plot.
`show.legend`	Logical. Should this layer be included in the legends?
`show.help`	Logical. If `TRUE`, display some minimal help.
`inherit`	Logical. If `TRUE`, inherit aesthetics from previous layers.
`environment`	An environment in which to evaluate the formula.

Value

A gg object that can be displayed with gf_girafe().

Additional interactive features

- onclick: JavaScript code (as character string) executed when clicking elements.
- Additional ggiraph aesthetics may be available depending on the geom.

See Also

gf_spoke(), gf_girafe()

Examples

```
if (require(dplyr)) {
  expand.grid(x = 0:10, y = 0:10) |>
    mutate(
      direction = round(x * y / 100 * 2 * pi, 1),
      size = (20 + x + y) / 50
      ) |>
    gf_spoke_interactive(
      y ~ x, angle = ~ direction, radius = ~ size,
      tooltip = ~ paste(
        "angle:", round(direction / 2 / pi * 360, 1),
        "degrees; size =", size),
      data_id = ~ paste(x, "-", y),
      hover_nearest = TRUE
      ) |>
    gf_point() |>
    gf_girafe(
      options = list(
        opts_hover(css = "stroke: red; stroke-width: 2;", nearest_distance = 10)
      )
    )
}
```

---

gf_step *Formula interface to geom_step()*

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. When using a geom_*() function to construct a layer, the stat argument can be used to override the default coupling between geoms and stats. The stat argument accepts the following: • A Stat ggproto subclass, for example StatCount. • A string naming the stat. To give the stat as a string, strip the function name of the stat_ prefix. For example, to use stat_count(), give the stat as "count". • For more information and other ways to specify the stat, see the layer stat documentation.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.
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. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.
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 `ggplot2::facet_wrap()` or `ggplot2::facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of **lattice**.

Evaluation

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

See Also

`ggplot2::geom_step()`

Examples

```
gf_step(births ~ date, data = mosaicData::Births78, color = ~wday)

# Roll your own Kaplan-Meier plot

if (require(survival) && require(broom)) {
  # fit a survival model
  surv_fit <- survfit(coxph(Surv(time, status) ~ age + sex, lung))
  surv_fit
  # use broom::tidy() to create a tidy data frame for plotting
  surv_df <- tidy(surv_fit)
  head(surv_df)
  # now create a plot
  surv_df |>
    gf_step(estimate ~ time) |>
    gf_ribbon(conf.low + conf.high ~ time, alpha = 0.2)
}
```

---

gf_step_interactive *Interactive step plots*

Description

Creates an interactive plot using ggiraph. This function extends `gf_step()` with interactive features like tooltips and clickable elements.

Arguments

object	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument.
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.
tooltip	A formula specifying a variable for tooltips, or a character vector.
data_id	A formula or character vector specifying data identifiers for interactive selection.
...	Additional arguments passed to the underlying geom.
alpha, color, size, shape, fill, group, stroke	Aesthetics passed to the geom.
xlab, ylab, title, subtitle, caption	Labels for the plot.
show.legend	Logical. Should this layer be included in the legends?
show.help	Logical. If TRUE, display some minimal help.
inherit	Logical. If TRUE, inherit aesthetics from previous layers.
environment	An environment in which to evaluate the formula.

Value

A gg object that can be displayed with gf_girafe().

Additional interactive features

- onclick: JavaScript code (as character string) executed when clicking elements.
- Additional ggiraph aesthetics may be available depending on the geom.

See Also

gf_step(), gf_girafe()

Examples

```
if (require(dplyr)) {
  mtcars |>
    group_by(cyl) |>
    mutate(ecdf = ecdf(mpg)(mpg)) |>
    gf_step_interactive(
      ecdf ~ mpg,
      group = ~ cyl,
      color = ~ factor(cyl),
      tooltip = ~ paste(cyl, "cylinders"),
      data_id = ~ mpg,
      hover_nearest = TRUE) |>
    gf_labs(color = "cylinders") |>
    gf_girafe()
}
```

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,
  size,
  parse,
  nudge_x = 0,
  nudge_y = 0,
  label.padding = unit(0.25, "lines"),
  label.r = unit(0.15, "lines"),
  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 the the specified location.
lineheight	Line height.
size	A numeric size or a formula used for mapping size.
parse	If TRUE, the labels will be parsed into expressions and displayed as described in ?plotmath.
nudge_x, nudge_y	Passed to ggplot2::position_nudge() to nudge text or labels horizontally or vertically.
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. When using a geom_*() function to construct a layer, the stat argument can be used to override the default coupling between geoms and stats. The stat argument accepts the following: • A Stat ggproto subclass, for example StatCount. • A string naming the stat. To give the stat as a string, strip the function name of the stat_ prefix. For example, to use stat_count(), give the stat as "count". • For more information and other ways to specify the stat, see the layer stat documentation.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following:

- The result of calling a position function, such as `position_jitter()`. This method allows for passing extra arguments to the position.
- A string naming the position adjustment. To give the position as a string, strip the function name of the `position_` prefix. For example, to use `position_jitter()`, give the position as `"jitter"`.
- For more information and other ways to specify the position, see the layer position documentation.

`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. To include legend keys for all levels, even when no data exists, use `TRUE`. If `NA`, all levels are shown in legend, but unobserved levels are omitted.

`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.

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 `ggplot2::facet_wrap()` or `ggplot2::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_text_interactive` *Interactive text annotations*

Description

Creates an interactive plot using ggiraph. This function extends `gf_text()` with interactive features like tooltips and clickable elements.

Arguments

`object`	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument.
`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.
`tooltip`	A formula specifying a variable for tooltips, or a character vector.
`data_id`	A formula or character vector specifying data identifiers for interactive selection.
`...`	Additional arguments passed to the underlying geom.
`alpha`, `color`, `size`, `shape`, `fill`, `group`, `stroke`	Aesthetics passed to the geom.
`xlab`, `ylab`, `title`, `subtitle`, `caption`	Labels for the plot.
`show.legend`	Logical. Should this layer be included in the legends?
`show.help`	Logical. If `TRUE`, display some minimal help.
`inherit`	Logical. If `TRUE`, inherit aesthetics from previous layers.
`environment`	An environment in which to evaluate the formula.

Value

A gg object that can be displayed with gf_girafe().

Additional interactive features

- onclick: JavaScript code (as character string) executed when clicking elements.
- Additional ggiraph aesthetics may be available depending on the geom.

See Also

gf_text(), gf_girafe()

Examples

```
gf_point_interactive(
  mpg ~ wt, data = mtcars, alpha = 0.4, size = 3,
  tooltip = ~ rownames(mtcars),
  data_id = 1:nrow(mtcars)
) |>
  gf_text_interactive(mpg ~ wt, data = mtcars[1:5, ],
                      label = ~ rownames(mtcars)[1:5],
                      size = 3,
                      angle = 20,
                      data_id = 1:5,
                      tooltip = ~ paste(rownames(mtcars)[1:5], "\nmpg:", mpg, "wt: ", wt)
  ) |>
  gf_girafe()
```

---

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 ggplot2::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_tile()` uses the center of the tile and its size (`x`, `y`, `width`, `height`), while `geom_rect()` can use those or the locations of the corners (`xmin`, `xmax`, `ymin` and `ymax`). `geom_raster()` is a high performance special case for when all the tiles are the same size, and no pattern fills are applied.

Usage

```
gf_tile(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...,
  alpha,
  color,
  fill,
  group,
  linetype,
  linewidth,
  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 ggplot2::facet_wrap() or ggplot2::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 `ggplot2::facet_wrap()` or `ggplot2::facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of **lattice**.

Evaluation

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

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

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_tile_interactive *Interactive tile plots*

Description

Creates an interactive plot using ggiraph. This function extends `gf_tile()` with interactive features like tooltips and clickable elements.

Arguments

`object`	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument.
`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.
`tooltip`	A formula specifying a variable for tooltips, or a character vector.
`data_id`	A formula or character vector specifying data identifiers for interactive selection.
`...`	Additional arguments passed to the underlying geom.
`alpha`, `color`, `size`, `shape`, `fill`, `group`, `stroke`	Aesthetics passed to the geom.

xlab, ylab, title, subtitle, caption	Labels for the plot.
show.legend	Logical. Should this layer be included in the legends?
show.help	Logical. If TRUE, display some minimal help.
inherit	Logical. If TRUE, inherit aesthetics from previous layers.
environment	An environment in which to evaluate the formula.

Value

A gg object that can be displayed with gf_girafe().

Additional interactive features

- onclick: JavaScript code (as character string) executed when clicking elements.
- Additional ggiraph aesthetics may be available depending on the geom.

See Also

gf_tile(), gf_girafe()

Examples

```
expand.grid(x = 1:10, y = 1:10) |>
  gf_tile_interactive(
    (x+y) ~ x + y,
    tooltip = ~ paste("x + y =", x + y)
  ) |>
  gf_labs(fill = "sum") |>
  gf_girafe()
```

---

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.
`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.
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	**[Deprecated]** Previous specification of drawing quantiles.
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(). Note that automatic calculation of the bandwidth does not take weights into account.
adjust	A multiplicate 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.
kernel	Kernel. See list of available kernels in density().
xlab	Label for x-axis. See also gf_labs().
ylab	Label for y-axis. See also gf_labs().
title, subtitle, caption	Title, sub-title, and caption for the plot. See also gf_labs().
geom, stat	Use to override the default connection between geom_violin() and stat_ydensity(). For more information about overriding these connections, see how the stat and geom arguments work.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.
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. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.

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 ggplot2::facet_wrap() or ggplot2::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

Hintze, J. L., Nelson, R. D. (1998) Violin Plots: A Box Plot-Density Trace Synergism. The American Statistician 52, 181-184.

See Also

ggplot2::geom_violin()

Examples

```
gf_violin(age ~ substance, data = mosaicData::HELPrct)
gf_violin(age ~ substance, data = mosaicData::HELPrct, fill = ~sex)
```

---

gf_violin_interactive *Interactive violin plots*

Description

Creates an interactive plot using ggiraph. This function extends gf_violin() with interactive features like tooltips and clickable elements.

Arguments

object	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument.
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.
tooltip	A formula specifying a variable for tooltips, or a character vector.
data_id	A formula or character vector specifying data identifiers for interactive selection.
...	Additional arguments passed to the underlying geom.
alpha, color, size, shape, fill, group, stroke	Aesthetics passed to the geom.
xlab, ylab, title, subtitle, caption	Labels for the plot.
show.legend	Logical. Should this layer be included in the legends?
show.help	Logical. If TRUE, display some minimal help.
inherit	Logical. If TRUE, inherit aesthetics from previous layers.
environment	An environment in which to evaluate the formula.

Value

A gg object that can be displayed with gf_girafe().

Additional interactive features

- onclick: JavaScript code (as character string) executed when clicking elements.
- Additional ggiraph aesthetics may be available depending on the geom.

See Also

gf_violin(), gf_girafe()

Examples

```
p <-
  mtcars |>
  gf_violin_interactive(
    mpg ~ factor(cyl),
    alpha = 0.5,
    fill = "skyblue",
    tooltip = ~ paste("Cylinders:", cyl)
  )

if (require(ggforce)) {
  p |> gf_sina(color = "red", alpha = 0.8) |> gf_girafe()
} else {
  p |> gf_girafe()
}
```

gf_vline_interactive *Interactive vertical lines*

Description

Creates an interactive plot using ggiraph. This function extends gf_vline() with interactive features like tooltips and clickable elements.

Arguments

object	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument.
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.
tooltip	A formula specifying a variable for tooltips, or a character vector.
data_id	A formula or character vector specifying data identifiers for interactive selection.
...	Additional arguments passed to the underlying geom.
alpha, color, size, shape, fill, group, stroke	Aesthetics passed to the geom.
xlab, ylab, title, subtitle, caption	Labels for the plot.
show.legend	Logical. Should this layer be included in the legends?
show.help	Logical. If TRUE, display some minimal help.
inherit	Logical. If TRUE, inherit aesthetics from previous layers.
environment	An environment in which to evaluate the formula.

Value

A gg object that can be displayed with gf_girafe().

Additional interactive features

- onclick: JavaScript code (as character string) executed when clicking elements.
- Additional ggiraph aesthetics may be available depending on the geom.

See Also

gf_vline(), gf_girafe()

Examples

```
gf_point(mpg ~ wt, data = mtcars, alpha = 0.7) |>
  gf_vline_interactive(xintercept = ~ mean(wt),
                       tooltip = ~ paste("Mean Weight:", round(mean(wt), 1)),
                       color = "blue", linetype = "dashed",
                       data_id = 1,
                       hover_nearest = TRUE) |>
  gf_girafe(
    options =
      list(
        opts_hover(nearest_distance = 10, css = "stroke: red; stroke-width: 3")
  ))
```

---

interactive_facets *Interactive facets*

Description

To create interactive facets, use `gf_facet_wrap_interactive()` or `gf_facet_grid_interactive()` and use `gf_labeller_interactive()` to create the `labeller`.

Usage

```
gf_facet_wrap_interactive(
  object,
  ...,
  labeller,
  interactive_on = c("text", "rect", "both")
)

gf_facet_grid_interactive(
  object,
  ...,
  labeller,
  interactive_on = c("text", "rect", "both")
)
```

Arguments

`object`	a ggplot graphic
`...`	additional arguments passed to `labeller` and to the ggplot2 faceting function (`ggplot2::facet_wrap()` or `ggplot2::facet_grid()`).
`labeller`	a labeller created using `gf_labeller_interactive()`
`interactive_on`	one of "text" (strip text is made interactive), "rect" (strip rectangles are made interactive), or "both". Can be abbreviated.

See Also

ggplot2::facet_wrap()

ggplot2::facet_grid()

gf_labeller_interactive()

Examples

```
mosaicData::Weather |>
gf_line_interactive(
  high_temp ~ date,
  color = ~city,
  show.legend = FALSE,
  tooltip = ~city,
  data_id = ~city
) |>
  gf_facet_wrap_interactive(
    ~year,
    ncol = 1,
    scales = "free_x",
    labeller = gf_labeller_interactive(
      data_id = ~year,
      tooltip = ~ glue::glue("This is the year {year}")
    )
  ) |>
  gf_theme(theme_facets_interactive()) |>
  gf_girafe(
    options = list(
      opts_hover_inv(css = "opacity:0.2;"),
      opts_hover(css = "stroke-width:2;", nearest_distance = 40),
      opts_tooltip(use_cursor_pos = FALSE, offx = 0, offy = -30)
    )
  )
```

---

interactive_layer_factory

Create an interactive ggformula layer function

Description

Primarily intended for package developers, this function factory is used to create layer functions in the ggformula package.

Usage

```
interactive_layer_factory(geom_fun)
```

Arguments

geom_fun	A character string naming an interactive geom (example: "geom_point_interactive")

---

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",
  interactive = FALSE,
  layer_func_interactive = "geom_point",
  pre = {
 },
  aes_form = y ~ x,
  extras = alist(),
  note = NULL,
  aesthetics = aes(),
  inherit.aes = TRUE,
  check.aes = TRUE,
  data = NULL,
  layer_fun = if (interactive) {
     quo(layer_interactive)
 } else {

    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).
interactive	A logical indicating whether this is being used to create an interactive layer.
layer_func_interactive	The function used to create the layer when ‘interactive“ is TRUE (or a quosure that evaluates to such a function).

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 emited when aesthetics provided don't match what is expected.
data	A data frame or NULL or NA.
layer_fun	function used to create a layer. The default value is anticipated to work in most (all?) cases.
...	Additional arguments.

Value

A function.

MIpop *Population of Michigan counties*

Description

Population of Michigan counties

Usage

```
data(MIpop)
```

Format

A data frame with populations of Michigan counties.

rank Population rank.

county County name.

population Population (2010 census).

percs_by_group *Compute groupwise proportions and percents*

Description

Transform a vector of counts and a vector of groups into a vector of proportions or percentages within groups.

Usage

```
percs_by_group(x, group)

props_by_group(x, group)
```

Arguments

x	A vector of counts
group	A vector to determine groups.

Examples

```
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.

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()
ggplot2::stat_qq()
gf_qq()
stat_lm()
gf_lm()
geom_lm()
gf_lm()
stat_ash()
gf_ash()

---

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

---

stat_lm *Linear Model Displays*

Description

Adds linear model fits to plots. geom_lm() and stat_lm() are essentially equivalent. Use geom_lm() unless you want a non-standard geom.

Usage

```
stat_lm(
  mapping = NULL,
  data = NULL,
  geom = "lm",
  position = "identity",
  interval = c("none", "prediction", "confidence"),
  level = 0.95,
  formula = y ~ x,
  lm.args = list(),
  backtrans = identity,
  ...,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
```

```
)

geom_lm(
  mapping = NULL,
  data = NULL,
  stat = "lm",
  position = "identity",
  interval = c("none", "prediction", "confidence"),
  level = 0.95,
  formula = y ~ x,
  lm.args = list(),
  backtrans = identity,
  ...,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). 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	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.
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()`'s `params` argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the `position` argument, or aesthetics that are required can *not* be passed through `....` Unknown arguments that are not part of the 4 categories below are ignored.

- Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, `colour = "red"` or `linewidth = 3`. The geom's documentation has an **Aesthetics** section that lists the available options. The 'required' aesthetics cannot be passed on to the `params`. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data.
- When constructing a layer using a `stat_*()` function, the `...` argument can be used to pass on parameters to the `geom` part of the layer. An example of this is `stat_density(geom = "area", outline.type = "both")`. The geom's documentation lists which parameters it can accept.
- Inversely, when constructing a layer using a `geom_*()` function, the `...` argument can be used to pass on parameters to the `stat` part of the layer. An example of this is `geom_area(stat = "density", adjust = 0.5)`. The stat's documentation lists which parameters it can accept.
- The `key_glyph` argument of `layer()` may also be passed on through `....` This can be one of the functions described as key glyphs, to change the display of the layer in the legend.

`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. To include legend keys for all levels, even when no data exists, use `TRUE`. If `NA`, all levels are shown in legend, but unobserved levels are omitted.

`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. `annotation_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

```
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() +
  ggplot2::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() +
  ggplot2::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

```
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 ggplot2::aes()] or ggplot2::aes_string()].
data	A data frame.
geom	A geom.
position	A position object.
...	Additional arguments
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

```
data(penguins, package = "palmerpenguins")
ggplot(data = penguins, aes(sample = bill_length_mm)) +
  geom_qq() +
  stat_qqline(alpha = 0.7, color = "red", linetype = "dashed") +
  ggplot2::facet_wrap(~species)
```

---

stat_spline *Geoms and stats for spline smoothing*

Description

Similar to ggplot2::geom_smooth, this adds spline fits to plots.

Usage

```
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,
```

```
  all.knots = FALSE,
  nknots = stats::.nknots.smspl,
  df.offset = 0,
  penalty = 1,
  control.spar = list(),
  tol = NULL,
  ...
)

geom_spline(
  mapping = NULL,
  data = NULL,
  stat = "spline",
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  weight = NULL,
  df = NULL,
  spar = NULL,
  cv = FALSE,
  all.knots = FALSE,
  nknots = stats::.nknots.smspl,
  df.offset = 0,
  penalty = 1,
  control.spar = list(),
  tol = NULL,
  ...
)
```

Arguments

mapping	An aesthetic mapping produced with ggplot2::aes()] or ggplot2::aes_string()].
data	A data frame.
geom	A geom.
position	A position object.
na.rm	A logical indicating whether a warning should be issued when missing values are removed before plotting.
show.legend	A logical indicating whether legends should be included for this layer. If NA, legends will be included for each aesthetic that is mapped.
inherit.aes	A logical indicating whether aesthetics should be inherited. When FALSE, the supplied mapping will be the only aesthetics used.
weight	An optional vector of weights. See smooth.spline().
df	desired equivalent degrees of freedom. See smooth.spline() for details.
spar	A smoothing parameter, typically in (0,1]. See smooth.spline() for details.
cv	A logical. See smooth.spline() for details.

all.knots	A logical. See smooth.spline() for details.
nknots	An integer or function giving the number of knots to use when all.knots = FALSE. See smooth.spline() for details.
df.offset	A numerical value used to increase the degrees of freedom when using GVC. See smooth.spline() for details.
penalty	the coefficient of the penalty for degrees of freedom in the GVC criterion. See smooth.spline() for details.
control.spar	An optional list used to control root finding when the parameter spar is computed. See smooth.spline() for details.
tol	A tolerance for sameness or uniqueness of the x values. The values are binned into bins of size tol and values which fall into the same bin are regarded as the same. Must be strictly positive (and finite). When NULL, IQR(x) * 10e-6 is used.
...	Additional arguments
stat	A stat.

Examples

```
if (require(mosaicData)) {
  ggplot(Births) + geom_spline(aes(x = date, y = births, colour = wday))
  ggplot(Births) + geom_spline(aes(x = date, y = births, colour = wday), nknots = 10)
}
```

---

theme_facets_interactive

Interactive theme for ggiraph facets

Description

A theme that enables interactive strip text and backgrounds for faceted plots using ggiraph. This theme ensures that facet labels can receive hover events and tooltips.

Usage

```
theme_facets_interactive(
  base_theme = theme_bw(),
  interactive_text = TRUE,
  interactive_rects = TRUE,
  strip_text_color = NULL,
  strip_background_color = NULL,
  strip_text_size = NULL
)
```

Arguments

base_theme	A theme that will will be modified
interactive_text	A logical indicating whether text elements of strips should be interactive.
interactive_rects	A logical indicating whether rect elements of strips should be interactive.
strip_text_color	Color for strip text (or NULL to retain settings from base_theme)
strip_background_color	Color for strip background (or NULL to retain settings from base_theme)
strip_text_size	Size for strip text (or NULL to retain settings from base_theme)

Value

A ggplot2 theme object

Index