

# Package ‘ggformula’

June 22, 2017

**Title** Formula Interface to the Grammar of Graphics

**Description** Provides a formula interface to 'ggplot2' graphics.

**Type** Package

**Version** 0.4.0

**Date** 2017-06-20

**License** MIT + file LICENSE

**LazyData** TRUE

**LazyLoad** TRUE

**Depends** R (>= 3.1), ggplot2

**Imports** rlang, mosaic, magrittr, tibble, stringr, glue

**RoxygenNote** 6.0.1

**Suggests** dplyr, testthat, mosaicData, knitr, statisticalModeling,  
rmarkdown, weatherData, lubridate

**VignetteBuilder** knitr

**NeedsCompilation** no

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

**Maintainer** Pruum Randall <rpruum@calvin.edu>

**Repository** CRAN

**Date/Publication** 2017-06-21 23:44:17 UTC

## R topics documented:

df_stats . . . . .	3
gf_abline . . . . .	4
gf_area . . . . .	5
gf_bar . . . . .	7
gf_boxplot . . . . .	9
gf_col . . . . .	10
gf_contour . . . . .	12

gf_count . . . . .	13
gf_crossbar . . . . .	14
gf_curve . . . . .	16
gf_dens . . . . .	18
gf_density . . . . .	19
gf_density2d . . . . .	21
gf_density_2d . . . . .	22
gf_dotplot . . . . .	23
gf_errorbar . . . . .	25
gf_errorbarh . . . . .	27
gf_frame . . . . .	28
gf_freqpoly . . . . .	30
gf_function . . . . .	31
gf_hex . . . . .	32
gf_histogram . . . . .	33
gf_jitter . . . . .	35
gf_label . . . . .	36
gf_labs . . . . .	38
gf_line . . . . .	38
gf_linerange . . . . .	40
gf_path . . . . .	42
gf_point . . . . .	43
gf_pointrange . . . . .	45
gf_qq . . . . .	46
gf_quantile . . . . .	48
gf_raster . . . . .	49
gf_rect . . . . .	51
gf_ribbon . . . . .	52
gf_rug . . . . .	54
gf_segment . . . . .	55
gf_smooth . . . . .	57
gf_spline . . . . .	58
gf_spoke . . . . .	60
gf_step . . . . .	61
gf_text . . . . .	63
gf_theme . . . . .	64
gf_tile . . . . .	65
gf_violin . . . . .	66
ggformula . . . . .	67

df\_stats

*Calculate basic statistics on a quantitative variable***Description**

Creates a data frame of statistics calculated on one variable, possibly for each group formed by combinations of additional variables. The resulting data frame has one column for each of the statistics requested as well as columns for any grouping variables.

**Usage**

```
df_stats(formula, data, ..., drop = TRUE, fargs = list(),
         long_names = TRUE, nice_names = FALSE)
```

**Arguments**

formula	A formula indicating which variables are to be used. See details.
data	A data frame or list containing the variables.
...	Functions used to compute the statistics. If this is empty, <a href="#">favstats()</a> is used. Functions used must accept a vector of values and return either a (possibly named) single value, a (possibly named) vector of values, or a data frame with one row.
drop	A logical indicating whether combinations of the grouping variables that do not occur in data should be dropped from the result.
fargs	Arguments passed to the functions in ...
long_names	A logical indicating whether the default names should include the name of the variable being summarized as well as the summarizing function name in the default case when names are not derived from the names of the returned object or an argument name.
nice_names	A logical indicating whether <a href="#">make.names()</a> should be used to force names of the returned data frame to be syntactically valid.

**Details**

Use a one-sided formula to compute summary statistics for the left hand side expression over the entire data. Use a two-sided formula to compute summary statistics for the left hand expression for each combination of levels of the expressions occurring on the right hand side. This is most useful when the left hand side is quantitative and each expression on the right hand side has relatively few unique values. A function like [ntiles\(\)](#) is often useful to create a few groups of roughly equal size determined by ranges of a quantitative variable. See the examples.

Note that unlike `dplyr::summarise()`, `df_stats()` ignores any grouping defined in data if data is a grouped tibble.

Names of columns in the resulting data frame are determined as follows. For named arguments in ..., the argument name is used. For unnamed arguments, if the statistic function returns a result with names, those names are used. Else, a name is computed from the expression in ... and the

name of the variable being summarized. For functions that produce multiple outputs without names, consecutive integers are appended to the names. See the examples.

### Value

A data frame.

### Examples

```
df_stats( ~ hp, data = mtcars)
df_stats( ~ hp, data = mtcars, mean, median)
df_stats( hp ~ cyl, data = mtcars, mean, median, range)
# magrittr style piping is also supported
mtcars %>% df_stats(hp ~ cyl)
gf_violin(hp ~ cyl, data = mtcars, group = ~ cyl) %>%
  gf_point(mean_hp ~ cyl, data = df_stats(hp ~ cyl, data = mtcars, mean))
```

---

gf\_abline

*Reference lines – horizontal, vertical, and diagonal.*

---

### Description

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

### Usage

```
gf_abline(object = NULL, gformula = NULL, data = NULL, geom = type,
  verbose = FALSE, add = inherits(object, c("gg", "ggplot")), ...,
  position = NULL, show.help = NULL)
```

```
gf_hline(object = NULL, gformula = NULL, data = NULL, geom = type,
  verbose = FALSE, add = inherits(object, c("gg", "ggplot")), ...,
  position = NULL, show.help = NULL)
```

```
gf_vline(object = NULL, gformula = NULL, data = NULL, geom = type,
  verbose = FALSE, add = inherits(object, c("gg", "ggplot")), ...,
  position = NULL, show.help = NULL)
```

```
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	A data frame with the variables to be plotted.
geom	A way to specify ggplot geoms that are not aliased to gf functions.
verbose	If TRUE print the ggplot2 command in the console.
add	If TRUE then construct just the layer with no frame. The result can be added to an existing frame.
...	Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with <code>attribute = value</code> , (b) ggplot2 aesthetics to be mapped with <code>attribute = ~expression</code> , or (c) attributes of the layer as a whole, which are set with <code>attribute = value</code> . Available attributes include <code>slope</code> , <code>intercept</code>
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
show.help	If TRUE, display some minimal help.
coef	A numeric vector of length at least 2, treated as intercept and slope. Additional components, if any, are ignored (with a warning).
model	An object with a method for <code>coef()</code> that returns a numeric vector, the first two elements of which are intercept and slope. This is equivalent to <code>coef = coef(model)</code> .

### See Also

[geom\\_abline\(\)](#), [geom\\_vline\(\)](#), [geom\\_hline\(\)](#)

### Examples

```
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 = -0.10, intercept = 33:36) %>%
  gf_hline(color = "navy", yintercept = c(20, 25)) %>%
  gf_vline(color = "brown", xintercept = c(200, 300))
```

---

gf\_area

*Formula interface to geom\_area()*

---

### Description

**ggformula** functions provide a formula interface to ggplot2 layer functions. For plots with just one layer, the formula interface is more compact and is consistent with modeling and **mosaic** notation. The functions generate a ggplot command string which can be displayed by setting `verbose = TRUE` as an argument.

### Usage

```
gf_area(object = NULL, gformula = NULL, data = NULL, geom = type,
  verbose = FALSE, add = inherits(object, c("gg", "ggplot")), ...,
  position = NULL, show.help = NULL)
```

## Arguments

object	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
gformula	A formula with shape $y \sim x$ . Faceting can be achieved by including <code> </code> in the formula.
data	A data frame with the variables to be plotted.
geom	A way to specify ggplot geoms that are not aliased to gf functions.
verbose	If TRUE print the ggplot2 command in the console.
add	If TRUE then construct just the layer with no frame. The result can be added to an existing frame.
...	Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with <code>attribute = value</code> , (b) ggplot2 aesthetics to be mapped with <code>attribute = ~expression</code> , or (c) attributes of the layer as a whole, which are set with <code>attribute = value</code> . Available attributes include <code>alpha</code> , <code>color</code> , <code>fill</code> , <code>group</code> , <code>linetype</code> , <code>size</code>
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
show.help	If TRUE, display some minimal help.

## Details

Positional aesthetics are typically specified using a formula (see the `gformula` argument). Additional formula terms of the form `+ attribute::value` map `attribute` to `value`. Additional terms of the form `+ attribute:value` will map `attribute` to `value` if `value` is the name of a variable in `data`, else `attribute` will be set to the constant `value`. Alternatively (and preferably) attributes can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`. In formulas of the form `A | B`, `B` will be used to form facets using `facet_wrap()` or `facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of **lattice**.

Evaluation 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.

## Value

a gg object

## See Also

[geom\\_area\(\)](#)

## Examples

```
if (require(weatherData) && require(dplyr)) {
  Temps <- NewYork2013 %>%
    mutate(date = lubridate::date(Time),
           month = lubridate::month(Time)) %>%
```

```

    filter(month <= 4) %>%
    group_by(date) %>%
    summarise(
      hi = max(Temperature, na.rm = TRUE),
      lo = min(Temperature, na.rm = TRUE)
    )
    gf_linerange(lo + hi ~ date, color = ~hi, data = Temps)
    gf_ribbon(lo + hi ~ date, data = Temps, color = "navy", alpha = 0.3)
    gf_area(hi ~ date, data = Temps, color = "navy", alpha = 0.3)

    Temps2 <- NewYork2013 %>% mutate(city = "NYC") %>%
      bind_rows(Mumbai2013 %>% mutate(city = "Mumbai")) %>%
      bind_rows(London2013 %>% mutate(city = "London")) %>%
      mutate(date = lubridate::date(Time),
             month = lubridate::month(Time)) %>%
      group_by(city, date) %>%
      summarise(
        hi = max(Temperature, na.rm = TRUE),
        lo = min(Temperature, na.rm = TRUE),
        mid = (hi + lo)/2
      )
      gf_ribbon(lo + hi ~ date, data = Temps2, alpha = 0.3) %>%
      gf_facet_grid(city ~ .)

      gf_linerange(lo + hi ~ date, color = ~ mid, data = Temps2) %>%
      gf_facet_grid(city ~ .) %>%
      gf_refine(scale_colour_gradientn(colors = rev(rainbow(5))))
  }

```

gf\_bar

*Formula interface to geom\_bar()*

## Description

**ggformula** functions provide a formula interface to ggplot2 layer functions. For plots with just one layer, the formula interface is more compact and is consistent with modeling and **mosaic** notation. The functions generate a ggplot command string which can be displayed by setting `verbose = TRUE` as an argument.

## Usage

```
gf_bar(object = NULL, gformula = NULL, data = NULL, geom = type,
       verbose = FALSE, add = inherits(object, c("gg", "ggplot")), ...,
       position = NULL, show.help = NULL)
```

```
gf_counts(object = NULL, gformula = NULL, data = NULL, geom = type,
          verbose = FALSE, add = inherits(object, c("gg", "ggplot")), ...,
          position = NULL, show.help = NULL)
```

## Arguments

object	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
gformula	A formula with shape <code>~x</code> . Faceting can be achieved by including <code> </code> in the formula.
data	A data frame with the variables to be plotted.
geom	A way to specify ggplot geoms that are not aliased to gf functions.
verbose	If TRUE print the ggplot2 command in the console.
add	If TRUE then construct just the layer with no frame. The result can be added to an existing frame.
...	Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with <code>attribute = value</code> , (b) ggplot2 aesthetics to be mapped with <code>attribute = ~expression</code> , or (c) attributes of the layer as a whole, which are set with <code>attribute = value</code> . Available attributes include <code>alpha</code> , <code>color</code> , <code>fill</code> , <code>group</code> , <code>linetype</code> , <code>size</code> , <code>width</code> , <code>binwidth</code>
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
show.help	If TRUE, display some minimal help.

## Details

Positional aesthetics are typically specified using a formula (see the `gformula` argument). Additional formula terms of the form `+ attribute::value` map `attribute` to `value`. Additional terms of the form `+ attribute:value` will map `attribute` to `value` if `value` is the name of a variable in `data`, else `attribute` will be set to the constant `value`. Alternatively (and preferably) attributes can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`. In formulas of the form `A | B`, `B` will be used to form facets using `facet_wrap()` or `facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of **lattice**.

Evaluation 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.

## Value

a gg object

## See Also

[geom\\_bar\(\)](#)

## Examples

```
if (require(mosaicData)) {
  gf_bar( ~ substance, data = HELPrct)
  gf_bar( ~ substance, data = HELPrct, fill = ~sex)
```



```

gf_bar( ~ substance, data = HELPrct, fill = ~sex, position = position_dodge())
# gf_counts() is another name for gf_bar()
gf_counts( ~ substance, data = HELPrct, fill = ~sex, position = position_dodge())
}

```

---

gf\_boxplot

*Formula interface to geom\_boxplot()*


---

## Description

**ggformula** functions provide a formula interface to ggplot2 layer functions. For plots with just one layer, the formula interface is more compact and is consistent with modeling and **mosaic** notation. The functions generate a ggplot command string which can be displayed by setting `verbose = TRUE` as an argument.

## Usage

```

gf_boxplot(object = NULL, gformula = NULL, data = NULL, geom = type,
  verbose = FALSE, add = inherits(object, c("gg", "ggplot")), ...,
  position = NULL, show.help = NULL)

```

## Arguments

object	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
gformula	A formula with shape <code>y ~ x</code> . Faceting can be achieved by including <code> </code> in the formula.
data	A data frame with the variables to be plotted.
geom	A way to specify ggplot geoms that are not aliased to gf functions.
verbose	If TRUE print the ggplot2 command in the console.
add	If TRUE then construct just the layer with no frame. The result can be added to an existing frame.
...	Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with <code>attribute = value</code> , (b) ggplot2 aesthetics to be mapped with <code>attribute = ~expression</code> , or (c) attributes of the layer as a whole, which are set with <code>attribute = value</code> . Available attributes include <code>alpha</code> , <code>color</code> , <code>fill</code> , <code>group</code> , <code>linetype</code> , <code>shape</code> , <code>size</code> , <code>weight</code> , <code>coef</code> , <code>outlier.color</code> , <code>outlier.fill</code> , <code>outlier.shape</code> , <code>outlier.size</code> , <code>outlier.stroke</code> , <code>outlier.alpha</code> , <code>notch</code> , <code>notchwidth</code> , <code>varwidth</code>
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
show.help	If TRUE, display some minimal help.

## Details

Positional aesthetics are typically specified using a formula (see the `gformula` argument). Additional formula terms of the form `+ attribute::value` map `attribute` to `value`. Additional terms of the form `+ attribute:value` will map `attribute` to `value` if `value` is the name of a variable in data, else `attribute` will be set to the constant value. Alternatively (and preferably) attributes can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`. In formulas of the form `A | B`, `B` will be used to form facets using `facet_wrap()` or `facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of **lattice**.

## Value

a gg object 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\\_boxplot\(\)](#)

## Examples

```
if (require(mosaicData)) {
  gf_boxplot(age ~ substance, data = HELPrct)
  gf_boxplot(age ~ substance, data = HELPrct, varwidth = TRUE)
  gf_boxplot(age ~ substance, data = HELPrct, color = ~sex)
  gf_boxplot(age ~ substance, data = HELPrct, color = ~sex, outlier.color = "gray50")
  # longer whiskers
  gf_boxplot(age ~ substance, data = HELPrct, color = ~sex, coef = 2)
  gf_boxplot(age ~ substance, data = HELPrct, color = ~sex, position = position_dodge(width = 0.9))
}
```

---

gf\_col

*Formula interface to geom\_col()*

---

## Description

**ggformula** functions provide a formula interface to **ggplot2** layer functions. For plots with just one layer, the formula interface is more compact and is consistent with modeling and **mosaic** notation. The functions generate a **ggplot** command string which can be displayed by setting `verbose = TRUE` as an argument.

## Usage

```
gf_col(object = NULL, gformula = NULL, data = NULL, geom = type,
  verbose = FALSE, add = inherits(object, c("gg", "ggplot")), ...,
  position = NULL, show.help = NULL)
```

## Arguments

object	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
gformula	A formula with shape $y \sim x$ . Faceting can be achieved by including <code> </code> in the formula.
data	A data frame with the variables to be plotted.
geom	A way to specify ggplot geoms that are not aliased to gf functions.
verbose	If TRUE print the ggplot2 command in the console.
add	If TRUE then construct just the layer with no frame. The result can be added to an existing frame.
...	Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with <code>attribute = value</code> , (b) ggplot2 aesthetics to be mapped with <code>attribute = ~expression</code> , or (c) attributes of the layer as a whole, which are set with <code>attribute = value</code> . Available attributes include <code>alpha</code> , <code>color</code> , <code>fill</code> , <code>group</code> , <code>linetype</code> , <code>size</code>
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
show.help	If TRUE, display some minimal help.

## Details

Positional aesthetics are typically specified using a formula (see the `gformula` argument). Additional formula terms of the form `+ attribute::value` map `attribute` to `value`. Additional terms of the form `+ attribute:value` will map `attribute` to `value` if `value` is the name of a variable in `data`, else `attribute` will be set to the constant `value`. Alternatively (and preferably) attributes can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`. In formulas of the form `A | B`, `B` will be used to form facets using `facet_wrap()` or `facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of **lattice**.

Evaluation 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.

## Value

a gg object

## See Also

[geom\\_col\(\)](#)

## Examples

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

gf\_contour

*Formula interface to geom\_contour()*

## Description

**ggformula** functions provide a formula interface to ggplot2 layer functions. For plots with just one layer, the formula interface is more compact and is consistent with modeling and **mosaic** notation. The functions generate a ggplot command string which can be displayed by setting `verbose = TRUE` as an argument.

## Usage

```
gf_contour(object = NULL, gformula = NULL, data = NULL, geom = type,
  verbose = FALSE, add = inherits(object, c("gg", "ggplot")), ...,
  position = NULL, show.help = NULL)
```

## Arguments

<code>object</code>	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.
<code>gformula</code>	A formula with shape $z \sim x + y$ . Faceting can be achieved by including <code> </code> in the formula.
<code>data</code>	A data frame with the variables to be plotted.
<code>geom</code>	A way to specify ggplot geoms that are not aliased to gf functions.
<code>verbose</code>	If TRUE print the ggplot2 command in the console.
<code>add</code>	If TRUE then construct just the layer with no frame. The result can be added to an existing frame.
<code>...</code>	Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with <code>attribute = value</code> , (b) ggplot2 aesthetics to be mapped with <code>attribute = ~expression</code> , or (c) attributes of the layer as a whole, which are set with <code>attribute = value</code> . Available attributes include
<code>position</code>	Position adjustment, either as a string, or the result of a call to a position adjustment function.
<code>show.help</code>	If TRUE, display some minimal help.

## Details

Positional aesthetics are typically specified using a formula (see the `gformula` argument). Additional formula terms of the form `+ attribute::value` map `attribute` to `value`. Additional terms of the form `+ attribute:value` will map `attribute` to `value` if `value` is the name of a variable in `data`, else `attribute` will be set to the constant `value`. Alternatively (and preferably) attributes can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`. In formulas of the form `A | B`, `B` will be used to form facets using `facet_wrap()` or `facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of **lattice**.

Evaluation 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.

## Value

a gg object

## See Also

[geom\\_contour\(\)](#)

## 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\_count

*Formula interface to geom\_count()*

---

## Description

**ggformula** functions provide a formula interface to ggplot2 layer functions. For plots with just one layer, the formula interface is more compact and is consistent with modeling and **mosaic** notation. The functions generate a ggplot command string which can be displayed by setting `verbose = TRUE` as an argument.

## Usage

```
gf_count(object = NULL, gformula = NULL, data = NULL, geom = type,
  verbose = FALSE, add = inherits(object, c("gg", "ggplot")), ...,
  position = NULL, show.help = NULL)
```

## Arguments

object	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
gformula	A formula with shape <code>y ~ x</code> . Faceting can be achieved by including <code> </code> in the formula.
data	A data frame with the variables to be plotted.
geom	A way to specify ggplot geoms that are not aliased to gf functions.
verbose	If TRUE print the ggplot2 command in the console.
add	If TRUE then construct just the layer with no frame. The result can be added to an existing frame.

...	Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with <code>attribute = value</code> , (b) ggplot2 aesthetics to be mapped with <code>attribute = ~expression</code> , or (c) attributes of the layer as a whole, which are set with <code>attribute = value</code> . Available attributes include <code>alpha</code> , <code>color</code> , <code>fill</code> , <code>group</code> , <code>shape</code> , <code>size</code> , <code>stroke</code> .
<code>position</code>	Position adjustment, either as a string, or the result of a call to a position adjustment function.
<code>show.help</code>	If TRUE, display some minimal help.

## Details

Positional aesthetics are typically specified using a formula (see the `gformula` argument). Additional formula terms of the form `+ attribute::value` map `attribute` to `value`. Additional terms of the form `+ attribute:value` will map `attribute` to `value` if `value` is the name of a variable in data, else `attribute` will be set to the constant `value`. Alternatively (and preferably) attributes can be set can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`. In formulas of the form `A | B`, `B` will be used to form facets using `facet_wrap()` or `facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of **lattice**.

Evaluation 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.

## Value

a gg object

## See Also

[geom\\_count\(\)](#)

## Examples

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

```
gf_count(hwy ~ cty, data = mpg, alpha = 0.5) %>%
  gf_refine(scale_size_area())
```

---

gf\_crossbar

*Formula interface to geom\_crossbar()*

---

## Description

**ggformula** functions provide a formula interface to ggplot2 layer functions. For plots with just one layer, the formula interface is more compact and is consistent with modeling and **mosaic** notation. The functions generate a ggplot command string which can be displayed by setting `verbose = TRUE` as an argument.

**Usage**

```
gf_crossbar(object = NULL, gformula = NULL, data = NULL, geom = type,
  verbose = FALSE, add = inherits(object, c("gg", "ggplot")), ...,
  position = NULL, show.help = NULL)
```

**Arguments**

object	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
gformula	A formula with shape <code>y + ymin + ymax ~ x</code> . Faceting can be achieved by including <code> </code> in the formula.
data	A data frame with the variables to be plotted.
geom	A way to specify ggplot geoms that are not aliased to gf functions.
verbose	If TRUE print the ggplot2 command in the console.
add	If TRUE then construct just the layer with no frame. The result can be added to an existing frame.
...	Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with <code>attribute = value</code> , (b) ggplot2 aesthetics to be mapped with <code>attribute = ~expression</code> , or (c) attributes of the layer as a whole, which are set with <code>attribute = value</code> . Available attributes include <code>alpha</code> , <code>color</code> , <code>group</code> , <code>linetype</code> , <code>size</code> , <code>fatten</code>
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
show.help	If TRUE, display some minimal help.

**Details**

Positional aesthetics are typically specified using a formula (see the `gformula` argument). Additional formula terms of the form `+ attribute::value` map `attribute` to `value`. Additional terms of the form `+ attribute:value` will map `attribute` to `value` if `value` is the name of a variable in `data`, else `attribute` will be set to the constant value. Alternatively (and preferably) attributes can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`. In formulas of the form `A | B`, `B` will be used to form facets using `facet_wrap()` or `facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of **lattice**.

Evaluation 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.

**Value**

a gg object

**See Also**

[geom\\_crossbar\(\)](#)

## 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) %>%
    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) %>%
    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\_curve

*Formula interface to geom\_curve()*


---

## Description

**ggformula** functions provide a formula interface to ggplot2 layer functions. For plots with just one layer, the formula interface is more compact and is consistent with modeling and **mosaic** notation. The functions generate a ggplot command string which can be displayed by setting `verbose = TRUE` as an argument.

## Usage

```
gf_curve(object = NULL, gformula = NULL, data = NULL, geom = type,
  verbose = FALSE, add = inherits(object, c("gg", "ggplot")), ...,
  position = NULL, show.help = NULL)
```

## Arguments

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



gformula	A formula with shape <code>y + yend ~ x + xend</code> . Faceting can be achieved by including <code> </code> in the formula.
data	A data frame with the variables to be plotted.
geom	A way to specify ggplot geoms that are not aliased to gf functions.
verbose	If TRUE print the ggplot2 command in the console.
add	If TRUE then construct just the layer with no frame. The result can be added to an existing frame.
...	Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with <code>attribute = value</code> , (b) ggplot2 aesthetics to be mapped with <code>attribute = ~expression</code> , or (c) attributes of the layer as a whole, which are set with <code>attribute = value</code> . Available attributes include <code>alpha</code> , <code>color</code> , <code>group</code> , <code>linetype</code> , <code>size</code> , <code>curvature</code> , <code>angle</code> , <code>ncp</code> , <code>arrow</code> , <code>lineend</code>
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
show.help	If TRUE, display some minimal help.

## Details

Positional aesthetics are typically specified using a formula (see the `gformula` argument). Additional formula terms of the form `+ attribute::value` map `attribute` to `value`. Additional terms of the form `+ attribute:value` will map `attribute` to `value` if `value` is the name of a variable in `data`, else `attribute` will be set to the constant `value`. Alternatively (and preferably) attributes can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`. In formulas of the form `A | B`, `B` will be used to form facets using `facet_wrap()` or `facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of **lattice**.

Evaluation 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.

## Value

a gg object

## See Also

[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\_dens

*Formula interface to geom\_line() and stat\_density()***Description**

**ggformula** functions provide a formula interface to ggplot2 layer functions. For plots with just one layer, the formula interface is more compact and is consistent with modeling and **mosaic** notation. The functions generate a ggplot command string which can be displayed by setting `verbose = TRUE` as an argument.

**Usage**

```
gf_dens(object = NULL, gformula = NULL, data = NULL, geom = type,
        verbose = FALSE, add = inherits(object, c("gg", "ggplot")), ...,
        position = NULL, show.help = NULL)
```

**Arguments**

<code>object</code>	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.
<code>gformula</code>	A formula with shape <code>~x</code> . Faceting can be achieved by including <code> </code> in the formula.
<code>data</code>	A data frame with the variables to be plotted.
<code>geom</code>	A way to specify ggplot geoms that are not aliased to gf functions.
<code>verbose</code>	If TRUE print the ggplot2 command in the console.
<code>add</code>	If TRUE then construct just the layer with no frame. The result can be added to an existing frame.
<code>...</code>	Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with <code>attribute = value</code> , (b) ggplot2 aesthetics to be mapped with <code>attribute = ~expression</code> , or (c) attributes of the layer as a whole, which are set with <code>attribute = value</code> . Available attributes include <code>stat</code> , <code>alpha</code> , <code>color</code> , <code>fill</code> , <code>group</code> , <code>linetype</code> , <code>size</code> , <code>weight</code>
<code>position</code>	Position adjustment, either as a string, or the result of a call to a position adjustment function.
<code>show.help</code>	If TRUE, display some minimal help.

**Details**

Positional aesthetics are typically specified using a formula (see the `gformula` argument). Additional formula terms of the form `+ attribute::value` map `attribute` to `value`. Additional terms of the form `+ attribute:value` will map `attribute` to `value` if `value` is the name of a variable in `data`, else `attribute` will be set to the constant `value`. Alternatively (and preferably) attributes can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`. In formulas of the form `A | B`, `B` will be used to form

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

Evaluation 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.

## Value

a gg object

## See Also

`geom_line()`

## Examples

```
gf_dens()
gf_density(~ Sepal.Length, color = ~Species, data = iris)
gf_dens(~ Sepal.Length, color = ~Species, data = iris)
gf_freqpoly(~ Sepal.Length, color = ~Species, data = iris)
# Chaining in the data
iris %>% gf_dens(~ Sepal.Length, color = ~Species)
```

---

gf\_density

*Formula interface to geom\_density()*

---

## Description

**ggformula** functions provide a formula interface to ggplot2 layer functions. For plots with just one layer, the formula interface is more compact and is consistent with modeling and **mosaic** notation. The functions generate a ggplot command string which can be displayed by setting `verbose = TRUE` as an argument.

## Usage

```
gf_density(object = NULL, gformula = NULL, data = NULL, geom = type,
  verbose = FALSE, add = inherits(object, c("gg", "ggplot")), ...,
  position = NULL, show.help = NULL)
```

## Arguments

object	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
gformula	A formula with shape <code>~x</code> . Faceting can be achieved by including <code> </code> in the formula.
data	A data frame with the variables to be plotted.
geom	A way to specify ggplot geoms that are not aliased to gf functions.

verbose	If TRUE print the ggplot2 command in the console.
add	If TRUE then construct just the layer with no frame. The result can be added to an existing frame.
...	Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with <code>attribute = value</code> , (b) ggplot2 aesthetics to be mapped with <code>attribute = ~expression</code> , or (c) attributes of the layer as a whole, which are set with <code>attribute = value</code> . Available attributes include <code>alpha</code> , <code>color</code> , <code>fill</code> , <code>group</code> , <code>linetype</code> , <code>size</code> , <code>weight</code>
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
show.help	If TRUE, display some minimal help.

## Details

Positional aesthetics are typically specified using a formula (see the `gformula` argument). Additional formula terms of the form `+ attribute::value` map `attribute` to `value`. Additional terms of the form `+ attribute:value` will map `attribute` to `value` if `value` is the name of a variable in data, else `attribute` will be set to the constant value. Alternatively (and preferably) attributes can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`. In formulas of the form `A | B`, `B` will be used to form facets using `facet_wrap()` or `facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of **lattice**.

Evaluation 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.

## Value

a gg object

## See Also

[geom\\_density\(\)](#)

## Examples

```
gf_dens()
gf_density(~ Sepal.Length, color = ~Species, data = iris)
gf_dens(~ Sepal.Length, color = ~Species, data = iris)
gf_freqpoly(~ Sepal.Length, color = ~Species, data = iris)
# Chaining in the data
iris %>% gf_dens(~ Sepal.Length, color = ~Species)
```

---

gf_density2d	<i>Formula interface to geom_density2d()</i>
--------------	----------------------------------------------

---

## Description

**ggformula** functions provide a formula interface to ggplot2 layer functions. For plots with just one layer, the formula interface is more compact and is consistent with modeling and **mosaic** notation. The functions generate a ggplot command string which can be displayed by setting `verbose = TRUE` as an argument.

## Usage

```
gf_density2d(object = NULL, gformula = NULL, data = NULL, geom = type,
  verbose = FALSE, add = inherits(object, c("gg", "ggplot")), ...,
  position = NULL, show.help = NULL)
```

## Arguments

<code>object</code>	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.
<code>gformula</code>	A formula with shape <code>y ~ x</code> . Faceting can be achieved by including <code> </code> in the formula.
<code>data</code>	A data frame with the variables to be plotted.
<code>geom</code>	A way to specify ggplot geoms that are not aliased to gf functions.
<code>verbose</code>	If TRUE print the ggplot2 command in the console.
<code>add</code>	If TRUE then construct just the layer with no frame. The result can be added to an existing frame.
<code>...</code>	Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with <code>attribute = value</code> , (b) ggplot2 aesthetics to be mapped with <code>attribute = ~expression</code> , or (c) attributes of the layer as a whole, which are set with <code>attribute = value</code> . Available attributes include <code>alpha</code> , <code>color</code> , <code>group</code> , <code>linetype</code> , <code>size</code> , <code>contour</code> , <code>n</code> , <code>h</code> , <code>lineend</code> , <code>linejoin</code> , <code>linemitre</code>
<code>position</code>	Position adjustment, either as a string, or the result of a call to a position adjustment function.
<code>show.help</code>	If TRUE, display some minimal help.

## Details

Positional aesthetics are typically specified using a formula (see the `gformula` argument). Additional formula terms of the form `+ attribute::value` map `attribute` to `value`. Additional terms of the form `+ attribute:value` will map `attribute` to `value` if `value` is the name of a variable in `data`, else `attribute` will be set to the constant `value`. Alternatively (and preferably) attributes can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`. In formulas of the form `A | B`, `B` will be used to form

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

Evaluation 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.

## Value

a gg object

## See Also

`geom_density2d()`

## Examples

```
if (require(mosaicData)) {
  gf_jitter(i1 ~ age, alpha = 0.2, data = HELPrct, width = 0.4, height = 0.4) %>%
  gf_density2d(i1 ~ age, data = HELPrct)
}
```

---

gf\_density\_2d

*Formula interface to geom\_density\_2d()*

---

## Description

**ggformula** functions provide a formula interface to ggplot2 layer functions. For plots with just one layer, the formula interface is more compact and is consistent with modeling and **mosaic** notation. The functions generate a ggplot command string which can be displayed by setting `verbose = TRUE` as an argument.

## Usage

```
gf_density_2d(object = NULL, gformula = NULL, data = NULL, geom = type,
  verbose = FALSE, add = inherits(object, c("gg", "ggplot")), ...,
  position = NULL, show.help = NULL)
```

## Arguments

<code>object</code>	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.
<code>gformula</code>	A formula with shape <code>y ~ x</code> . Faceting can be achieved by including <code> </code> in the formula.
<code>data</code>	A data frame with the variables to be plotted.
<code>geom</code>	A way to specify ggplot geoms that are not aliased to gf functions.
<code>verbose</code>	If TRUE print the ggplot2 command in the console.

add	If TRUE then construct just the layer with no frame. The result can be added to an existing frame.
...	Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with <code>attribute = value</code> , (b) ggplot2 aesthetics to be mapped with <code>attribute = ~expression</code> , or (c) attributes of the layer as a whole, which are set with <code>attribute = value</code> . Available attributes include <code>alpha</code> , <code>color</code> , <code>group</code> , <code>linetype</code> , <code>size</code> , <code>contour</code> , <code>n</code> , <code>h</code> , <code>lineend</code> , <code>linejoin</code> , <code>linemitre</code>
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
show.help	If TRUE, display some minimal help.

## Details

Positional aesthetics are typically specified using a formula (see the `gformula` argument). Additional formula terms of the form `+ attribute::value` map `attribute` to `value`. Additional terms of the form `+ attribute:value` will map `attribute` to `value` if `value` is the name of a variable in data, else `attribute` will be set to the constant value. Alternatively (and preferably) attributes can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`. In formulas of the form `A | B`, `B` will be used to form facets using `facet_wrap()` or `facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of **lattice**.

Evaluation 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.

## Value

a gg object

## See Also

[geom\\_density\\_2d\(\)](#)

## Examples

```
if (require(mosaicData)) {
  gf_jitter(i1 ~ age, alpha = 0.2, data = HELPrct, width = 0.4, height = 0.4) %>%
  gf_density_2d(i1 ~ age, data = HELPrct)
}
```

## Description

**ggformula** functions provide a formula interface to ggplot2 layer functions. For plots with just one layer, the formula interface is more compact and is consistent with modeling and **mosaic** notation. The functions generate a ggplot command string which can be displayed by setting `verbose = TRUE` as an argument.

## Usage

```
gf_dotplot(object = NULL, gformula = NULL, data = NULL, geom = type,
  verbose = FALSE, add = inherits(object, c("gg", "ggplot")), ...,
  position = NULL, show.help = NULL)
```

## Arguments

<code>object</code>	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.
<code>gformula</code>	A formula with shape <code>~x</code> . Faceting can be achieved by including <code> </code> in the formula.
<code>data</code>	A data frame with the variables to be plotted.
<code>geom</code>	A way to specify ggplot geoms that are not aliased to gf functions.
<code>verbose</code>	If TRUE print the ggplot2 command in the console.
<code>add</code>	If TRUE then construct just the layer with no frame. The result can be added to an existing frame.
<code>...</code>	Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with <code>attribute = value</code> , (b) ggplot2 aesthetics to be mapped with <code>attribute = ~expression</code> , or (c) attributes of the layer as a whole, which are set with <code>attribute = value</code> . Available attributes include <code>alpha</code> , <code>color</code> , <code>fill</code> , <code>group</code> , <code>binwidth</code> , <code>binaxis</code> , <code>method</code> , <code>binpositions</code> , <code>stackdir</code> , <code>stackratio</code> , <code>dotsize</code> , <code>stackgroups</code> , <code>origin</code> , <code>right</code> , <code>width</code> , <code>drop</code>
<code>position</code>	Position adjustment, either as a string, or the result of a call to a position adjustment function.
<code>show.help</code>	If TRUE, display some minimal help.

## Details

Positional aesthetics are typically specified using a formula (see the `gformula` argument). Additional formula terms of the form `+ attribute::value` map `attribute` to `value`. Additional terms of the form `+ attribute:value` will map `attribute` to `value` if `value` is the name of a variable in `data`, else `attribute` will be set to the constant value. Alternatively (and preferably) attributes can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`. In formulas of the form `A | B`, `B` will be used to form facets using `facet_wrap()` or `facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of **lattice**.

Evaluation 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.



**Value**

a gg object

**See Also**

[geom\\_dotplot\(\)](#)

**Examples**

```
gf_dotplot(~ Sepal.Length, fill = ~Species, data = iris)
```

---

gf\_errorbar

*Formula interface to geom\_errorbar()*


---

**Description**

**ggformula** functions provide a formula interface to ggplot2 layer functions. For plots with just one layer, the formula interface is more compact and is consistent with modeling and **mosaic** notation. The functions generate a ggplot command string which can be displayed by setting `verbose = TRUE` as an argument.

**Usage**

```
gf_errorbar(object = NULL, gformula = NULL, data = NULL, geom = type,
  verbose = FALSE, add = inherits(object, c("gg", "ggplot")), ...,
  position = NULL, show.help = NULL)
```

**Arguments**

object	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
gformula	A formula with shape <code>ymin + ymax ~ x</code> . Faceting can be achieved by including <code> </code> in the formula.
data	A data frame with the variables to be plotted.
geom	A way to specify ggplot geoms that are not aliased to gf functions.
verbose	If TRUE print the ggplot2 command in the console.
add	If TRUE then construct just the layer with no frame. The result can be added to an existing frame.
...	Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with <code>attribute = value</code> , (b) ggplot2 aesthetics to be mapped with <code>attribute = ~expression</code> , or (c) attributes of the layer as a whole, which are set with <code>attribute = value</code> . Available attributes include <code>alpha</code> , <code>color</code> , <code>group</code> , <code>linetype</code> , <code>size</code>
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
show.help	If TRUE, display some minimal help.

## Details

Positional aesthetics are typically specified using a formula (see the `gformula` argument). Additional formula terms of the form `+ attribute::value` map `attribute` to `value`. Additional terms of the form `+ attribute:value` will map `attribute` to `value` if `value` is the name of a variable in data, else `attribute` will be set to the constant value. Alternatively (and preferably) attributes can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`. In formulas of the form `A | B`, `B` will be used to form facets using `facet_wrap()` or `facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of **lattice**.

Evaluation 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.

## Value

a gg object

## See Also

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

---

gf_errorbarh	<i>Formula interface to geom_errorbarh()</i>
--------------	----------------------------------------------

---

## Description

**ggformula** functions provide a formula interface to ggplot2 layer functions. For plots with just one layer, the formula interface is more compact and is consistent with modeling and **mosaic** notation. The functions generate a ggplot command string which can be displayed by setting `verbose = TRUE` as an argument.

## Usage

```
gf_errorbarh(object = NULL, gformula = NULL, data = NULL, geom = type,
  verbose = FALSE, add = inherits(object, c("gg", "ggplot")), ...,
  position = NULL, show.help = NULL)
```

## Arguments

<code>object</code>	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.
<code>gformula</code>	A formula with shape <code>y ~ x + xmin + xmax</code> . Faceting can be achieved by including <code> </code> in the formula.
<code>data</code>	A data frame with the variables to be plotted.
<code>geom</code>	A way to specify ggplot geoms that are not aliased to gf functions.
<code>verbose</code>	If TRUE print the ggplot2 command in the console.
<code>add</code>	If TRUE then construct just the layer with no frame. The result can be added to an existing frame.
<code>...</code>	Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with <code>attribute = value</code> , (b) ggplot2 aesthetics to be mapped with <code>attribute = ~expression</code> , or (c) attributes of the layer as a whole, which are set with <code>attribute = value</code> . Available attributes include <code>alpha</code> , <code>color</code> , <code>group</code> , <code>linetype</code> , <code>size</code>
<code>position</code>	Position adjustment, either as a string, or the result of a call to a position adjustment function.
<code>show.help</code>	If TRUE, display some minimal help.

## Details

Positional aesthetics are typically specified using a formula (see the `gformula` argument). Additional formula terms of the form `+ attribute::value` map `attribute` to `value`. Additional terms of the form `+ attribute:value` will map `attribute` to `value` if `value` is the name of a variable in `data`, else `attribute` will be set to the constant value. Alternatively (and preferably) attributes can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`. In formulas of the form `A | B`, `B` will be used to form facets using `facet_wrap()` or `facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of **lattice**.

Evaluation 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\\_errorbarh\(\)](#)

### Examples

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

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

---

gf\_frame

*Formula interface to geom\_blank()*

---

### Description

**ggformula** functions provide a formula interface to **ggplot2** layer functions. For plots with just one layer, the formula interface is more compact and is consistent with modeling and **mosaic** notation. The functions generate a **ggplot** command string which can be displayed by setting `verbose = TRUE` as an argument.

### Usage

```
gf_frame(object = NULL, gformula = NULL, data = NULL, geom = type,
  verbose = FALSE, add = inherits(object, c("gg", "ggplot")), ...,
  position = NULL, show.help = NULL)
```

**Arguments**

object	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
gformula	A formula with shape $y \sim x$ . Faceting can be achieved by including <code> </code> in the formula.
data	A data frame with the variables to be plotted.
geom	A way to specify ggplot2 geoms that are not aliased to gf functions.
verbose	If TRUE print the ggplot2 command in the console.
add	If TRUE then construct just the layer with no frame. The result can be added to an existing frame.
...	Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with <code>attribute = value</code> , (b) ggplot2 aesthetics to be mapped with <code>attribute = ~expression</code> , or (c) attributes of the layer as a whole, which are set with <code>attribute = value</code> . Available attributes include
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
show.help	If TRUE, display some minimal help.

**Details**

Positional aesthetics are typically specified using a formula (see the `gformula` argument). Additional formula terms of the form `+ attribute::value` map `attribute` to `value`. Additional terms of the form `+ attribute:value` will map `attribute` to `value` if `value` is the name of a variable in `data`, else `attribute` will be set to the constant `value`. Alternatively (and preferably) attributes can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`. In formulas of the form `A | B`, `B` will be used to form facets using `facet_wrap()` or `facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of **lattice**.

Evaluation 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.

**Value**

a gg object

**See Also**

[geom\\_blank\(\)](#)

**Examples**

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

gf\_freqpoly

*Formula interface to geom\_freqpoly()***Description**

**ggformula** functions provide a formula interface to ggplot2 layer functions. For plots with just one layer, the formula interface is more compact and is consistent with modeling and **mosaic** notation. The functions generate a ggplot command string which can be displayed by setting `verbose = TRUE` as an argument.

**Usage**

```
gf_freqpoly(object = NULL, gformula = NULL, data = NULL, geom = type,
  verbose = FALSE, add = inherits(object, c("gg", "ggplot")), ...,
  position = NULL, show.help = NULL)
```

**Arguments**

<code>object</code>	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.
<code>gformula</code>	A formula with shape <code>~x</code> . Faceting can be achieved by including <code> </code> in the formula.
<code>data</code>	A data frame with the variables to be plotted.
<code>geom</code>	A way to specify ggplot geoms that are not aliased to gf functions.
<code>verbose</code>	If TRUE print the ggplot2 command in the console.
<code>add</code>	If TRUE then construct just the layer with no frame. The result can be added to an existing frame.
<code>...</code>	Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with <code>attribute = value</code> , (b) ggplot2 aesthetics to be mapped with <code>attribute = ~expression</code> , or (c) attributes of the layer as a whole, which are set with <code>attribute = value</code> . Available attributes include <code>alpha</code> , <code>color</code> , <code>group</code> , <code>linetype</code> , <code>size</code> , <code>binwidth</code> , <code>bins</code> , <code>center</code> , <code>boundary</code> ,
<code>position</code>	Position adjustment, either as a string, or the result of a call to a position adjustment function.
<code>show.help</code>	If TRUE, display some minimal help.

**Details**

Positional aesthetics are typically specified using a formula (see the `gformula` argument). Additional formula terms of the form `+ attribute::value` map `attribute` to `value`. Additional terms of the form `+ attribute:value` will map `attribute` to `value` if `value` is the name of a variable in `data`, else `attribute` will be set to the constant `value`. Alternatively (and preferably) attributes can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`. In formulas of the form `A | B`, `B` will be used to form

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

Evaluation 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.

## Value

a gg object

## See Also

`geom_freqpoly()`

## Examples

```
gf_histogram(~ Sepal.Length | Species, alpha = 0.2, data = iris, bins = 20) %>%
  gf_freqpoly(~ Sepal.Length, data = iris, color = ~Species, bins = 20)
gf_freqpoly(~ Sepal.Length, color = ~Species, data = iris, bins = 20)
gf_dens(~ Sepal.Length, data = iris, color = "navy") %>%
  gf_freqpoly(~ Sepal.Length, y = ~..density.., data = iris, color = "red", bins = 20)
```

---

gf\_function

*Layers displaying graphs of functions*

---

## Description

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

## Usage

```
gf_function(object, fun, ...)
```

```
gf_fun(object, formula, ...)
```

## 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.
...	Other arguments such as <code>position="dodge"</code> .
formula	A formula describing a function. See examples.

## Examples

```
if (require(mosaicData)) {
  gf_histogram(..density.. ~ age, data = HELPrct, binwidth = 3, alpha = 0.6) %>%
    gf_function(fun = dnorm,
               args = list(mean = mean(HELPrct$age), sd = sd(HELPrct$age)),
               color = "red")
}
gf_point(Sepal.Length ~ Sepal.Width, data = iris) %>%
gf_fun(5 + 3 * cos(10 * x) ~ x)
# Utility bill is quadratic in month?
if (require(mosaic)) {
  f <- makeFun(lm(totalbill ~ poly(month, 2), data = Utilities))
  gf_point(totalbill ~ month, data = Utilities, alpha = 0.6) %>%
    gf_fun(f(m) ~ m, color = "red")
}
```

---

gf\_hex

*Formula interface to geom\_hex()*


---

## Description

**ggformula** functions provide a formula interface to ggplot2 layer functions. For plots with just one layer, the formula interface is more compact and is consistent with modeling and **mosaic** notation. The functions generate a ggplot command string which can be displayed by setting `verbose = TRUE` as an argument.

## Usage

```
gf_hex(object = NULL, gformula = NULL, data = NULL, geom = type,
       verbose = FALSE, add = inherits(object, c("gg", "ggplot")), ...,
       position = NULL, show.help = NULL)
```

## Arguments

object	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
gformula	A formula with shape <code>y ~ x</code> . Faceting can be achieved by including <code> </code> in the formula.
data	A data frame with the variables to be plotted.
geom	A way to specify ggplot geoms that are not aliased to gf functions.
verbose	If TRUE print the ggplot2 command in the console.
add	If TRUE then construct just the layer with no frame. The result can be added to an existing frame.
...	Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with <code>attribute = value</code> , (b) ggplot2 aesthetics to be mapped with <code>attribute = ~expression</code> , or (c) attributes of the layer as a whole, which are set with <code>attribute = value</code> . Available attributes include <code>bins</code> , <code>binwidth</code> , <code>alpha</code> , <code>color</code> , <code>fill</code> , <code>group</code> , <code>size</code>



position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
show.help	If TRUE, display some minimal help.

### Details

Positional aesthetics are typically specified using a formula (see the `gformula` argument). Additional formula terms of the form `+ attribute::value` map `attribute` to `value`. Additional terms of the form `+ attribute:value` will map `attribute` to `value` if `value` is the name of a variable in data, else `attribute` will be set to the constant value. Alternatively (and preferably) attributes can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`. In formulas of the form `A | B`, `B` will be used to form facets using `facet_wrap()` or `facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of **lattice**.

Evaluation 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.

### Value

a gg object

### See Also

[geom\\_hex\(\)](#)

### Examples

```
if (require(mosaicData)) {
  gf_hex(i1 ~ age, data = HELPrct, bins = 15) %>%
  gf_density2d(i1 ~ age, data = HELPrct, color = "red", alpha = 0.5)
}
```

---

gf\_histogram

*Formula interface to geom\_histogram()*

---

### Description

**ggformula** functions provide a formula interface to **ggplot2** layer functions. For plots with just one layer, the formula interface is more compact and is consistent with modeling and **mosaic** notation. The functions generate a **ggplot** command string which can be displayed by setting `verbose = TRUE` as an argument.

### Usage

```
gf_histogram(object = NULL, gformula = NULL, data = NULL, geom = type,
  verbose = FALSE, add = inherits(object, c("gg", "ggplot")), ...,
  position = NULL, show.help = NULL)
```

## Arguments

<code>object</code>	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.
<code>gformula</code>	A formula with shape <code>~x</code> or <code>y ~ x</code> . <code>y</code> may be <code>..density..</code> or <code>..count..</code> or <code>..ndensity..</code> or <code>..ncount...</code> . Faceting can be achieved by including <code> </code> in the formula.
<code>data</code>	A data frame with the variables to be plotted.
<code>geom</code>	A way to specify ggplot geoms that are not aliased to gf functions.
<code>verbose</code>	If TRUE print the ggplot2 command in the console.
<code>add</code>	If TRUE then construct just the layer with no frame. The result can be added to an existing frame.
<code>...</code>	Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with <code>attribute = value</code> , (b) ggplot2 aesthetics to be mapped with <code>attribute = ~expression</code> , or (c) attributes of the layer as a whole, which are set with <code>attribute = value</code> . Available attributes include <code>alpha</code> , <code>color</code> , <code>fill</code> , <code>group</code> , <code>linetype</code> , <code>size</code>
<code>position</code>	Position adjustment, either as a string, or the result of a call to a position adjustment function.
<code>show.help</code>	If TRUE, display some minimal help.

## Details

Positional aesthetics are typically specified using a formula (see the `gformula` argument). Additional formula terms of the form `+ attribute::value` map `attribute` to `value`. Additional terms of the form `+ attribute:value` will map `attribute` to `value` if `value` is the name of a variable in `data`, else `attribute` will be set to the constant value. Alternatively (and preferably) attributes can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`. In formulas of the form `A | B`, `B` will be used to form facets using `facet_wrap()` or `facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of **lattice**.

Evaluation 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.

## Value

a gg object

## See Also

[geom\\_histogram\(\)](#)

## Examples

```
x <- rnorm(1000)
gf_histogram( ~ x, bins = 30)
gf_histogram( ..density.. ~ x, bins = 30)
gf_histogram(~ Sepal.Length | Species, data = iris, binwidth = 0.25)
```

gf\_jitter

*Formula interface to geom\_jitter()***Description**

**ggformula** functions provide a formula interface to ggplot2 layer functions. For plots with just one layer, the formula interface is more compact and is consistent with modeling and **mosaic** notation. The functions generate a ggplot command string which can be displayed by setting `verbose = TRUE` as an argument.

**Usage**

```
gf_jitter(object = NULL, gformula = NULL, data = NULL, geom = type,
  verbose = FALSE, add = inherits(object, c("gg", "ggplot")), ...,
  position = NULL, show.help = NULL)
```

**Arguments**

<code>object</code>	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.
<code>gformula</code>	A formula with shape <code>y ~ x</code> . Faceting can be achieved by including <code> </code> in the formula.
<code>data</code>	A data frame with the variables to be plotted.
<code>geom</code>	A way to specify ggplot geoms that are not aliased to gf functions.
<code>verbose</code>	If TRUE print the ggplot2 command in the console.
<code>add</code>	If TRUE then construct just the layer with no frame. The result can be added to an existing frame.
<code>...</code>	Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with <code>attribute = value</code> , (b) ggplot2 aesthetics to be mapped with <code>attribute = ~expression</code> , or (c) attributes of the layer as a whole, which are set with <code>attribute = value</code> . Available attributes include <code>alpha</code> , <code>color</code> , <code>size</code> , <code>shape</code> , <code>fill</code> , <code>group</code> , <code>stroke</code> , <code>width</code> , <code>height</code>
<code>position</code>	Position adjustment, either as a string, or the result of a call to a position adjustment function.
<code>show.help</code>	If TRUE, display some minimal help.

**Details**

Positional aesthetics are typically specified using a formula (see the `gformula` argument). Additional formula terms of the form `+ attribute::value` map `attribute` to `value`. Additional terms of the form `+ attribute:value` will map `attribute` to `value` if `value` is the name of a variable in `data`, else `attribute` will be set to the constant `value`. Alternatively (and preferably) attributes can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`. In formulas of the form `A | B`, `B` will be used to form

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

Evaluation 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.

## Value

a gg object

## See Also

`geom_jitter()`

## Examples

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

---

gf\_label

*Formula interface to geom\_label()*

---

## Description

**ggformula** functions provide a formula interface to ggplot2 layer functions. For plots with just one layer, the formula interface is more compact and is consistent with modeling and **mosaic** notation. The functions generate a ggplot command string which can be displayed by setting `verbose = TRUE` as an argument.

## Usage

```
gf_label(object = NULL, gformula = NULL, data = NULL, geom = type,
  verbose = FALSE, add = inherits(object, c("gg", "ggplot")), ...,
  position = NULL, show.help = NULL)
```

## Arguments

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

<code>data</code>	A data frame with the variables to be plotted.
<code>geom</code>	A way to specify ggplot geoms that are not aliased to gf functions.
<code>verbose</code>	If TRUE print the ggplot2 command in the console.
<code>add</code>	If TRUE then construct just the layer with no frame. The result can be added to an existing frame.
<code>...</code>	Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with <code>attribute = value</code> , (b) ggplot2 aesthetics to be mapped with <code>attribute = ~expression</code> , or (c) attributes of the layer as a whole, which are set with <code>attribute = value</code> . Available attributes include <code>label</code> , <code>alpha</code> , <code>angle</code> , <code>color</code> , <code>family</code> , <code>fontface</code> , <code>group</code> , <code>hjust</code> , <code>lineheight</code> , <code>size</code> , <code>vjust</code> , <code>parse</code> , <code>nudge_x</code> , <code>nudge_y</code> , <code>lparse</code> , <code>nudge_x</code> , <code>nudge_y</code> , <code>label.padding</code> , <code>label.r</code> , <code>label.size</code> , <code>check_overlap</code>
<code>position</code>	Position adjustment, either as a string, or the result of a call to a position adjustment function.
<code>show.help</code>	If TRUE, display some minimal help.

## Details

Positional aesthetics are typically specified using a formula (see the `gformula` argument). Additional formula terms of the form `+ attribute::value` map `attribute` to `value`. Additional terms of the form `+ attribute:value` will map `attribute` to `value` if `value` is the name of a variable in `data`, else `attribute` will be set to the constant value. Alternatively (and preferably) attributes can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`. In formulas of the form `A | B`, `B` will be used to form facets using `facet_wrap()` or `facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of **lattice**.

Evaluation 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.

## Value

a gg object

## See Also

[geom\\_label\(\)](#)

## Examples

```
if (require(dplyr)) {
  iris_means <-
    iris %>%
    group_by(Species) %>%
    summarise(Sepal.Length = mean(Sepal.Length), Sepal.Width = mean(Sepal.Width))
  gf_point(Sepal.Length ~ Sepal.Width, data = iris, color = ~ Species) %>%
  gf_label(Sepal.Length ~ Sepal.Width, data = iris_means,
    label = ~Species, color = ~Species, size = 2, alpha = 0.7)
}
```

---

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_facet_wrap(object, ...)
```

```
gf_facet_grid(object, ...)
```

```
gf_refine(object, ...)
```

### Arguments

object            a gg object

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

### Details

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

### Value

a modified gg object

---

gf\_line

Formula interface to geom\_line()

---

### Description

**ggformula** functions provide a formula interface to ggplot2 layer functions. For plots with just one layer, the formula interface is more compact and is consistent with modeling and **mosaic** notation. The functions generate a ggplot command string which can be displayed by setting verbose = TRUE as an argument.

**Usage**

```
gf_line(object = NULL, gformula = NULL, data = NULL, geom = type,
        verbose = FALSE, add = inherits(object, c("gg", "ggplot")), ...,
        position = NULL, show.help = NULL)
```

**Arguments**

object	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
gformula	A formula with shape $y \sim x$ . Faceting can be achieved by including <code> </code> in the formula.
data	A data frame with the variables to be plotted.
geom	A way to specify ggplot geoms that are not aliased to gf functions.
verbose	If TRUE print the ggplot2 command in the console.
add	If TRUE then construct just the layer with no frame. The result can be added to an existing frame.
...	Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with <code>attribute = value</code> , (b) ggplot2 aesthetics to be mapped with <code>attribute = ~expression</code> , or (c) attributes of the layer as a whole, which are set with <code>attribute = value</code> . Available attributes include <code>alpha</code> , <code>color</code> , <code>fill</code> , <code>group</code> , <code>linetype</code> , <code>size</code> , <code>lineend</code> , <code>linejoin</code> , <code>linemitre</code> , <code>arrow</code>
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
show.help	If TRUE, display some minimal help.

**Details**

Positional aesthetics are typically specified using a formula (see the `gformula` argument). Additional formula terms of the form `+ attribute::value` map `attribute` to `value`. Additional terms of the form `+ attribute:value` will map `attribute` to `value` if `value` is the name of a variable in `data`, else `attribute` will be set to the constant `value`. Alternatively (and preferably) attributes can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`. In formulas of the form `A | B`, `B` will be used to form facets using `facet_wrap()` or `facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of **lattice**.

Evaluation 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.

**Value**

a gg object

**See Also**

[geom\\_line\(\)](#)

## Examples

```
if (require(mosaicData)) {
  gf_point(age ~ sex, alpha = 0.25, data = HELPrct)
  gf_point(births ~ date, color = ~wday, data = Births78)
  # lines make the exceptions stand out more prominently
  gf_line(births ~ date, color = ~wday, data = Births78)
}
```

---

gf\_linerange

*Formula interface to geom\_linerange()*


---

## Description

**ggformula** functions provide a formula interface to ggplot2 layer functions. For plots with just one layer, the formula interface is more compact and is consistent with modeling and **mosaic** notation. The functions generate a ggplot command string which can be displayed by setting `verbose = TRUE` as an argument.

## Usage

```
gf_linerange(object = NULL, gformula = NULL, data = NULL, geom = type,
  verbose = FALSE, add = inherits(object, c("gg", "ggplot")), ...,
  position = NULL, show.help = NULL)
```

## Arguments

object	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
gformula	A formula with shape <code>ymin + ymax ~ x</code> . Faceting can be achieved by including <code> </code> in the formula.
data	A data frame with the variables to be plotted.
geom	A way to specify ggplot geoms that are not aliased to gf functions.
verbose	If TRUE print the ggplot2 command in the console.
add	If TRUE then construct just the layer with no frame. The result can be added to an existing frame.
...	Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with <code>attribute = value</code> , (b) ggplot2 aesthetics to be mapped with <code>attribute = ~expression</code> , or (c) attributes of the layer as a whole, which are set with <code>attribute = value</code> . Available attributes include <code>alpha</code> , <code>color</code> , <code>group</code> , <code>linetype</code> , <code>size</code>
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
show.help	If TRUE, display some minimal help.



## Details

Positional aesthetics are typically specified using a formula (see the `gformula` argument). Additional formula terms of the form `+ attribute::value` map `attribute` to `value`. Additional terms of the form `+ attribute:value` will map `attribute` to `value` if `value` is the name of a variable in data, else `attribute` will be set to the constant value. Alternatively (and preferably) attributes can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`. In formulas of the form `A | B`, `B` will be used to form facets using `facet_wrap()` or `facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of **lattice**.

Evaluation 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.

## Value

a gg object

## See Also

[geom\\_linerange\(\)](#)

## Examples

```
gf_linerange()
if (require(weatherData) & require(dplyr)) {
  Temps <- NewYork2013 %>% mutate(city = "NYC") %>%
  bind_rows(Mumbai2013 %>% mutate(city = "Mumbai")) %>%
  bind_rows(London2013 %>% mutate(city = "London")) %>%
    mutate(date = lubridate::date(Time),
           month = lubridate::month(Time)) %>%
  group_by(city, date) %>%
  summarise(
    hi = max(Temperature, na.rm = TRUE),
    lo = min(Temperature, na.rm = TRUE),
    mid = (hi + lo)/2
  )

  gf_ribbon(lo + hi ~ date, data = Temps, fill = ~city, alpha = 0.4) %>%
    gf_theme(theme = theme_minimal())
  gf_linerange(lo + hi ~ date | city ~ ., color = ~mid, data = Temps) %>%
    gf_refine(scale_colour_gradientn(colors = rev(rainbow(5))))
  gf_ribbon(lo + hi ~ date | city ~ ., data = Temps)
  # Chaining in the data
  Temps %>% gf_ribbon(lo + hi ~ date, alpha = 0.4) %>%
    gf_facet_grid(city ~ .)
}
```

---

gf_path	<i>Formula interface to geom_path()</i>
---------	-----------------------------------------

---

## Description

**ggformula** functions provide a formula interface to ggplot2 layer functions. For plots with just one layer, the formula interface is more compact and is consistent with modeling and **mosaic** notation. The functions generate a ggplot command string which can be displayed by setting `verbose = TRUE` as an argument.

## Usage

```
gf_path(object = NULL, gformula = NULL, data = NULL, geom = type,
        verbose = FALSE, add = inherits(object, c("gg", "ggplot")), ...,
        position = NULL, show.help = NULL)
```

## Arguments

object	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
gformula	A formula with shape <code>y ~ x</code> . Faceting can be achieved by including <code> </code> in the formula.
data	A data frame with the variables to be plotted.
geom	A way to specify ggplot geoms that are not aliased to gf functions.
verbose	If TRUE print the ggplot2 command in the console.
add	If TRUE then construct just the layer with no frame. The result can be added to an existing frame.
...	Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with <code>attribute = value</code> , (b) ggplot2 aesthetics to be mapped with <code>attribute = ~expression</code> , or (c) attributes of the layer as a whole, which are set with <code>attribute = value</code> . Available attributes include <code>alpha</code> , <code>color</code> , <code>group</code> , <code>linetype</code> , <code>size</code> , <code>lineend</code> , <code>linejoin</code> , <code>linemitre</code> , <code>arrow</code>
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
show.help	If TRUE, display some minimal help.

## Details

Positional aesthetics are typically specified using a formula (see the `gformula` argument). Additional formula terms of the form `+ attribute::value` map `attribute` to `value`. Additional terms of the form `+ attribute:value` will map `attribute` to `value` if `value` is the name of a variable in `data`, else `attribute` will be set to the constant `value`. Alternatively (and preferably) attributes can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`. In formulas of the form `A | B`, `B` will be used to form

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

Evaluation 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.

## Value

a gg object

## See Also

`geom_path()`

## Examples

```
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\_point

*Formula interface to geom\_point()*

---

## Description

**ggformula** functions provide a formula interface to ggplot2 layer functions. For plots with just one layer, the formula interface is more compact and is consistent with modeling and **mosaic** notation. The functions generate a ggplot command string which can be displayed by setting `verbose = TRUE` as an argument.

## Usage

```
gf_point(object = NULL, gformula = NULL, data = NULL, geom = type,
  verbose = FALSE, add = inherits(object, c("gg", "ggplot")), ...,
  position = NULL, show.help = NULL)
```

## Arguments

<code>object</code>	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.
<code>gformula</code>	A formula with shape <code>y ~ x</code> . Faceting can be achieved by including <code> </code> in the formula.
<code>data</code>	A data frame with the variables to be plotted.
<code>geom</code>	A way to specify ggplot geoms that are not aliased to gf functions.
<code>verbose</code>	If TRUE print the ggplot2 command in the console.

add	If TRUE then construct just the layer with no frame. The result can be added to an existing frame.
...	Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with <code>attribute = value</code> , (b) ggplot2 aesthetics to be mapped with <code>attribute = ~expression</code> , or (c) attributes of the layer as a whole, which are set with <code>attribute = value</code> . Available attributes include <code>alpha</code> , <code>color</code> , <code>size</code> , <code>shape</code> , <code>fill</code> , <code>group</code> , <code>stroke</code>
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
show.help	If TRUE, display some minimal help.

## Details

Positional aesthetics are typically specified using a formula (see the `gformula` argument). Additional formula terms of the form `+ attribute::value` map `attribute` to `value`. Additional terms of the form `+ attribute:value` will map `attribute` to `value` if `value` is the name of a variable in data, else `attribute` will be set to the constant value. Alternatively (and preferably) attributes can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`. In formulas of the form `A | B`, `B` will be used to form facets using `facet_wrap()` or `facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of **lattice**.

Evaluation 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.

## Value

a gg object

## See Also

[geom\\_point\(\)](#)

## Examples

```
gf_point()
gf_point(mpg ~ hp + color:cyl + size:wt, data = mtcars, verbose = TRUE)
# faceting -- two ways
gf_point(mpg ~ hp, data = mtcars) %>%
  gf_facet_wrap(~ am)
gf_point(mpg ~ hp + group:cyl | am, data = mtcars)
gf_point(mpg ~ hp + group:cyl | ~ am, data = mtcars)
gf_point(mpg ~ hp + group:cyl | am ~ ., data = mtcars)

# Chaining in the data
mtcars %>% gf_point(mpg ~ wt)
```

---

gf_pointrange	<i>Formula interface to geom_pointrange()</i>
---------------	-----------------------------------------------

---

## Description

**ggformula** functions provide a formula interface to ggplot2 layer functions. For plots with just one layer, the formula interface is more compact and is consistent with modeling and **mosaic** notation. The functions generate a ggplot command string which can be displayed by setting `verbose = TRUE` as an argument.

## Usage

```
gf_pointrange(object = NULL, gformula = NULL, data = NULL, geom = type,
  verbose = FALSE, add = inherits(object, c("gg", "ggplot")), ...,
  position = NULL, show.help = NULL)
```

## Arguments

<code>object</code>	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.
<code>gformula</code>	A formula with shape <code>y + ymin + ymax ~ x</code> . Faceting can be achieved by including <code> </code> in the formula.
<code>data</code>	A data frame with the variables to be plotted.
<code>geom</code>	A way to specify ggplot geoms that are not aliased to gf functions.
<code>verbose</code>	If TRUE print the ggplot2 command in the console.
<code>add</code>	If TRUE then construct just the layer with no frame. The result can be added to an existing frame.
<code>...</code>	Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with <code>attribute = value</code> , (b) ggplot2 aesthetics to be mapped with <code>attribute = ~expression</code> , or (c) attributes of the layer as a whole, which are set with <code>attribute = value</code> . Available attributes include <code>alpha</code> , <code>color</code> , <code>group</code> , <code>linetype</code> , <code>size</code> , <code>fatten</code>
<code>position</code>	Position adjustment, either as a string, or the result of a call to a position adjustment function.
<code>show.help</code>	If TRUE, display some minimal help.

## Details

Positional aesthetics are typically specified using a formula (see the `gformula` argument). Additional formula terms of the form `+ attribute::value` map `attribute` to `value`. Additional terms of the form `+ attribute:value` will map `attribute` to `value` if `value` is the name of a variable in `data`, else `attribute` will be set to the constant `value`. Alternatively (and preferably) attributes can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`. In formulas of the form `A | B`, `B` will be used to form facets using `facet_wrap()` or `facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of **lattice**.

Evaluation 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.

### Value

a gg object

### See Also

[geom\\_pointrange\(\)](#)

### 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) %>%
    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) %>%
    gf_facet_grid( ~ sex)
}
```

---

gf\_qq

*Formula interface to geom\_qq()*

---

### Description

**ggformula** functions provide a formula interface to ggplot2 layer functions. For plots with just one layer, the formula interface is more compact and is consistent with modeling and **mosaic** notation. The functions generate a ggplot command string which can be displayed by setting `verbose = TRUE` as an argument.

**Usage**

```
gf_qq(object = NULL, gformula = NULL, data = NULL, geom = type,
       verbose = FALSE, add = inherits(object, c("gg", "ggplot")), ...,
       position = NULL, show.help = NULL)
```

**Arguments**

<code>object</code>	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.
<code>gformula</code>	A formula with shape <code>~sample</code> . Faceting can be achieved by including <code> </code> in the formula.
<code>data</code>	A data frame with the variables to be plotted.
<code>geom</code>	A way to specify ggplot geoms that are not aliased to gf functions.
<code>verbose</code>	If TRUE print the ggplot2 command in the console.
<code>add</code>	If TRUE then construct just the layer with no frame. The result can be added to an existing frame.
<code>...</code>	Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with <code>attribute = value</code> , (b) ggplot2 aesthetics to be mapped with <code>attribute = ~expression</code> , or (c) attributes of the layer as a whole, which are set with <code>attribute = value</code> . Available attributes include <code>group</code> , <code>x</code> , <code>y</code> , <code>distribution</code> , <code>dparams</code>
<code>position</code>	Position adjustment, either as a string, or the result of a call to a position adjustment function.
<code>show.help</code>	If TRUE, display some minimal help.

**Details**

Positional aesthetics are typically specified using a formula (see the `gformula` argument). Additional formula terms of the form `+ attribute::value` map `attribute` to `value`. Additional terms of the form `+ attribute:value` will map `attribute` to `value` if `value` is the name of a variable in `data`, else `attribute` will be set to the constant value. Alternatively (and preferably) attributes can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`. In formulas of the form `A | B`, `B` will be used to form facets using `facet_wrap()` or `facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of **lattice**.

Evaluation 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.

**Value**

a gg object

**See Also**

[geom\\_qq\(\)](#)

## Examples

```
gf_qq(~rnorm(100))
gf_qq(~Sepal.Length | Species, data = iris)
gf_qq(~Sepal.Length, color = ~Species, data = iris)
```

---

gf\_quantile

*Formula interface to geom\_quantile()*


---

## Description

**ggformula** functions provide a formula interface to ggplot2 layer functions. For plots with just one layer, the formula interface is more compact and is consistent with modeling and **mosaic** notation. The functions generate a ggplot command string which can be displayed by setting `verbose = TRUE` as an argument.

## Usage

```
gf_quantile(object = NULL, gformula = NULL, data = NULL, geom = type,
  verbose = FALSE, add = inherits(object, c("gg", "ggplot")), ...,
  position = NULL, show.help = NULL)
```

## Arguments

object	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
gformula	A formula with shape <code>y ~ x</code> . Faceting can be achieved by including <code> </code> in the formula.
data	A data frame with the variables to be plotted.
geom	A way to specify ggplot geoms that are not aliased to gf functions.
verbose	If TRUE print the ggplot2 command in the console.
add	If TRUE then construct just the layer with no frame. The result can be added to an existing frame.
...	Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with <code>attribute = value</code> , (b) ggplot2 aesthetics to be mapped with <code>attribute = ~expression</code> , or (c) attributes of the layer as a whole, which are set with <code>attribute = value</code> . Available attributes include <code>alpha</code> , <code>color</code> , <code>group</code> , <code>linetype</code> , <code>size</code> , <code>weight</code> , <code>lineend</code> , <code>linejoin</code> , <code>linemitre</code> , <code>quantiles</code> , <code>formula</code> , <code>method</code> , <code>method.args</code>
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
show.help	If TRUE, display some minimal help.



## Details

Positional aesthetics are typically specified using a formula (see the `gformula` argument). Additional formula terms of the form `+ attribute::value` map `attribute` to `value`. Additional terms of the form `+ attribute:value` will map `attribute` to `value` if `value` is the name of a variable in data, else `attribute` will be set to the constant value. Alternatively (and preferably) attributes can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`. In formulas of the form `A | B`, `B` will be used to form facets using `facet_wrap()` or `facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of **lattice**.

Evaluation 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.

## Value

a gg object

## See Also

`geom_quantile()`

## Examples

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

---

gf\_raster

*Formula interface to geom\_raster()*

---

## Description

**ggformula** functions provide a formula interface to **ggplot2** layer functions. For plots with just one layer, the formula interface is more compact and is consistent with modeling and **mosaic** notation. The functions generate a **ggplot** command string which can be displayed by setting `verbose = TRUE` as an argument.

## Usage

```
gf_raster(object = NULL, gformula = NULL, data = NULL, geom = type,
  verbose = FALSE, add = inherits(object, c("gg", "ggplot")), ...,
  position = NULL, show.help = NULL)
```

## Arguments

object	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
gformula	A formula with shape $y \sim x$ . Faceting can be achieved by including <code> </code> in the formula.
data	A data frame with the variables to be plotted.
geom	A way to specify ggplot geoms that are not aliased to gf functions.
verbose	If TRUE print the ggplot2 command in the console.
add	If TRUE then construct just the layer with no frame. The result can be added to an existing frame.
...	Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with <code>attribute = value</code> , (b) ggplot2 aesthetics to be mapped with <code>attribute = ~expression</code> , or (c) attributes of the layer as a whole, which are set with <code>attribute = value</code> . Available attributes include <code>alpha</code> , <code>color</code> , <code>fill</code> , <code>group</code> , <code>linetype</code> , <code>size</code> , <code>hjust</code> , <code>vjust</code> , <code>interpolate</code>
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
show.help	If TRUE, display some minimal help.

## Details

Positional aesthetics are typically specified using a formula (see the `gformula` argument). Additional formula terms of the form `+ attribute::value` map `attribute` to `value`. Additional terms of the form `+ attribute:value` will map `attribute` to `value` if `value` is the name of a variable in `data`, else `attribute` will be set to the constant `value`. Alternatively (and preferably) attributes can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`. In formulas of the form `A | B`, `B` will be used to form facets using `facet_wrap()` or `facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of **lattice**.

Evaluation 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.

## Value

a gg object

## See Also

[geom\\_raster\(\)](#)

## Examples

```
# Justification controls where the cells are anchored
D <- expand.grid(x = 0:5, y = 0:5)
D$z <- runif(nrow(D))
```

```
# centered squares
gf_raster(z ~ x + y, data = D)
gf_raster(y ~ x, fill = ~ z, data = D)
# zero padding
gf_raster(z ~ x + y, data = D, hjust = 0, vjust = 0)
```

---

gf_rect	<i>Formula interface to geom_rect()</i>
---------	-----------------------------------------

---

## Description

**ggformula** functions provide a formula interface to ggplot2 layer functions. For plots with just one layer, the formula interface is more compact and is consistent with modeling and **mosaic** notation. The functions generate a ggplot command string which can be displayed by setting `verbose = TRUE` as an argument.

## Usage

```
gf_rect(object = NULL, gformula = NULL, data = NULL, geom = type,
        verbose = FALSE, add = inherits(object, c("gg", "ggplot")), ...,
        position = NULL, show.help = NULL)
```

## Arguments

object	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
gformula	A formula with shape <code>ymin + ymax ~ xmin + xmax</code> . Faceting can be achieved by including <code> </code> in the formula.
data	A data frame with the variables to be plotted.
geom	A way to specify ggplot geoms that are not aliased to gf functions.
verbose	If TRUE print the ggplot2 command in the console.
add	If TRUE then construct just the layer with no frame. The result can be added to an existing frame.
...	Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with <code>attribute = value</code> , (b) ggplot2 aesthetics to be mapped with <code>attribute = ~expression</code> , or (c) attributes of the layer as a whole, which are set with <code>attribute = value</code> . Available attributes include <code>alpha</code> , <code>color</code> , <code>fill</code> , <code>group</code> , <code>linetype</code> , <code>size</code>
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
show.help	If TRUE, display some minimal help.

## Details

Positional aesthetics are typically specified using a formula (see the `gformula` argument). Additional formula terms of the form `+ attribute::value` map `attribute` to `value`. Additional terms of the form `+ attribute:value` will map `attribute` to `value` if `value` is the name of a variable in data, else `attribute` will be set to the constant value. Alternatively (and preferably) attributes can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`. In formulas of the form `A | B`, `B` will be used to form facets using `facet_wrap()` or `facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of **lattice**.

Evaluation 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\\_rect\(\)](#)

## Examples

```
gf_rect( 1 + 2 ~ 3 + 4, alpha = 0.3, color = "red")
```

---

gf\_ribbon

*Formula interface to geom\_ribbon()*

---

## Description

**ggformula** functions provide a formula interface to `ggplot2` layer functions. For plots with just one layer, the formula interface is more compact and is consistent with modeling and **mosaic** notation. The functions generate a `ggplot` command string which can be displayed by setting `verbose = TRUE` as an argument.

## Usage

```
gf_ribbon(object = NULL, gformula = NULL, data = NULL, geom = type,
  verbose = FALSE, add = inherits(object, c("gg", "ggplot")), ...,
  position = NULL, show.help = NULL)
```

## Arguments

<code>object</code>	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.
<code>gformula</code>	A formula with shape <code>ymin + ymax ~ x</code> . Faceting can be achieved by including <code> </code> in the formula.
<code>data</code>	A data frame with the variables to be plotted.
<code>geom</code>	A way to specify <code>ggplot</code> geoms that are not aliased to <code>gf</code> functions.

verbose	If TRUE print the ggplot2 command in the console.
add	If TRUE then construct just the layer with no frame. The result can be added to an existing frame.
...	Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with <code>attribute = value</code> , (b) ggplot2 aesthetics to be mapped with <code>attribute = ~expression</code> , or (c) attributes of the layer as a whole, which are set with <code>attribute = value</code> . Available attributes include <code>alpha</code>
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
show.help	If TRUE, display some minimal help.

## Details

Positional aesthetics are typically specified using a formula (see the `gformula` argument). Additional formula terms of the form `+ attribute::value` map `attribute` to `value`. Additional terms of the form `+ attribute:value` will map `attribute` to `value` if `value` is the name of a variable in data, else `attribute` will be set to the constant value. Alternatively (and preferably) attributes can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`. In formulas of the form `A | B`, `B` will be used to form facets using `facet_wrap()` or `facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of **lattice**.

Evaluation 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.

## Value

a gg object

## See Also

[geom\\_ribbon\(\)](#)

## Examples

```
gf_ribbon()
if (require(weatherData) & require(dplyr)) {
  Temps <- NewYork2013 %>% mutate(city = "NYC") %>%
  bind_rows(Mumbai2013 %>% mutate(city = "Mumbai")) %>%
  bind_rows(London2013 %>% mutate(city = "London")) %>%
    mutate(date = lubridate::date(Time),
           month = lubridate::month(Time)) %>%
  group_by(city, date) %>%
  summarise(
    hi = max(Temperature, na.rm = TRUE),
    lo = min(Temperature, na.rm = TRUE),
    mid = (hi + lo)/2
  )
}
```

```

gf_ribbon(lo + hi ~ date, data = Temps, fill = ~city, alpha = 0.4) %>%
  gf_theme(theme = theme_minimal())
gf_linerange(lo + hi ~ date | city ~ ., color = ~mid, data = Temps) %>%
  gf_refine(scale_colour_gradientn(colors = rev(rainbow(5))))
gf_ribbon(lo + hi ~ date | city ~ ., data = Temps)
# Chaining in the data
Temps %>% gf_ribbon(lo + hi ~ date, alpha = 0.4) %>%
  gf_facet_grid(city ~ .)
}

```

gf\_rug

*Formula interface to geom\_rug()*

## Description

**ggformula** functions provide a formula interface to ggplot2 layer functions. For plots with just one layer, the formula interface is more compact and is consistent with modeling and **mosaic** notation. The functions generate a ggplot command string which can be displayed by setting `verbose = TRUE` as an argument.

## Usage

```

gf_rug(object = NULL, gformula = NULL, data = NULL, geom = type,
  verbose = FALSE, add = inherits(object, c("gg", "ggplot")), ...,
  position = NULL, show.help = NULL)

```

## Arguments

<code>object</code>	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.
<code>gformula</code>	A formula with shape <code>~x</code> or <code>y ~ x</code> . Faceting can be achieved by including <code> </code> in the formula.
<code>data</code>	A data frame with the variables to be plotted.
<code>geom</code>	A way to specify ggplot geoms that are not aliased to gf functions.
<code>verbose</code>	If TRUE print the ggplot2 command in the console.
<code>add</code>	If TRUE then construct just the layer with no frame. The result can be added to an existing frame.
<code>...</code>	Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with <code>attribute = value</code> , (b) ggplot2 aesthetics to be mapped with <code>attribute = ~expression</code> , or (c) attributes of the layer as a whole, which are set with <code>attribute = value</code> . Available attributes include <code>sides</code> , <code>alpha</code> , <code>color</code> , <code>group</code> , <code>linetype</code> , <code>size</code>
<code>position</code>	Position adjustment, either as a string, or the result of a call to a position adjustment function.
<code>show.help</code>	If TRUE, display some minimal help.

## Details

Positional aesthetics are typically specified using a formula (see the `gformula` argument). Additional formula terms of the form `+ attribute::value` map `attribute` to `value`. Additional terms of the form `+ attribute:value` will map `attribute` to `value` if `value` is the name of a variable in data, else `attribute` will be set to the constant value. Alternatively (and preferably) attributes can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`. In formulas of the form `A | B`, `B` will be used to form facets using `facet_wrap()` or `facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of **lattice**.

Evaluation 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.

## Value

a gg object

## See Also

[geom\\_rug\(\)](#)

## Examples

```
gf_histogram(~eruptions, data = faithful) %>%
gf_rug(~eruptions, data = faithful, color = "red", sides = "bl") %>%
gf_rug(~eruptions, data = faithful, color = "navy", sides = "tr")
gf_point(Sepal.Length ~ Sepal.Width, data = iris) %>%
gf_rug(Sepal.Length ~ Sepal.Width)
gf_point(Sepal.Length ~ Sepal.Width, data = iris) %>%
gf_rug( x = ~ Sepal.Width, data = iris, color = "navy") %>%
gf_rug( y = ~ Sepal.Length, data = iris, color = "red")
```

---

gf\_segment

---

*Formula interface to geom\_segment()*


---

## Description

**ggformula** functions provide a formula interface to **ggplot2** layer functions. For plots with just one layer, the formula interface is more compact and is consistent with modeling and **mosaic** notation. The functions generate a **ggplot** command string which can be displayed by setting `verbose = TRUE` as an argument.

## Usage

```
gf_segment(object = NULL, gformula = NULL, data = NULL, geom = type,
  verbose = FALSE, add = inherits(object, c("gg", "ggplot")), ...,
  position = NULL, show.help = NULL)
```

## Arguments

object	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
gformula	A formula with shape <code>y + yend ~ x + xend</code> . Faceting can be achieved by including <code> </code> in the formula.
data	A data frame with the variables to be plotted.
geom	A way to specify ggplot2 geoms that are not aliased to gf functions.
verbose	If TRUE print the ggplot2 command in the console.
add	If TRUE then construct just the layer with no frame. The result can be added to an existing frame.
...	Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with <code>attribute = value</code> , (b) ggplot2 aesthetics to be mapped with <code>attribute = ~expression</code> , or (c) attributes of the layer as a whole, which are set with <code>attribute = value</code> . Available attributes include <code>alpha</code> , <code>color</code> , <code>group</code> , <code>linetype</code> , <code>size</code> , <code>arrow</code> , <code>lineend</code>
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
show.help	If TRUE, display some minimal help.

## Details

Positional aesthetics are typically specified using a formula (see the `gformula` argument). Additional formula terms of the form `+ attribute::value` map `attribute` to `value`. Additional terms of the form `+ attribute:value` will map `attribute` to `value` if `value` is the name of a variable in `data`, else `attribute` will be set to the constant `value`. Alternatively (and preferably) attributes can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`. In formulas of the form `A | B`, `B` will be used to form facets using `facet_wrap()` or `facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of **lattice**.

Evaluation 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.

## Value

a gg object

## See Also

[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_smooth	<i>Formula interface to geom_smooth()</i>
-----------	-------------------------------------------

---

## Description

**ggformula** functions provide a formula interface to ggplot2 layer functions. For plots with just one layer, the formula interface is more compact and is consistent with modeling and **mosaic** notation. The functions generate a ggplot command string which can be displayed by setting `verbose = TRUE` as an argument.

## Usage

```
gf_smooth(object = NULL, gformula = NULL, data = NULL, geom = type,
  verbose = FALSE, add = inherits(object, c("gg", "ggplot")), ...,
  position = NULL, show.help = NULL)
```

```
gf_lm(object = NULL, gformula = NULL, data = NULL, geom = type,
  verbose = FALSE, add = inherits(object, c("gg", "ggplot")), ...,
  position = NULL, show.help = NULL)
```

## Arguments

object	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
gformula	A formula with shape $y \sim x$ . Faceting can be achieved by including <code> </code> in the formula.
data	A data frame with the variables to be plotted.
geom	A way to specify ggplot geoms that are not aliased to gf functions.
verbose	If TRUE print the ggplot2 command in the console.
add	If TRUE then construct just the layer with no frame. The result can be added to an existing frame.
...	Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with <code>attribute = value</code> , (b) ggplot2 aesthetics to be mapped with <code>attribute = ~expression</code> , or (c) attributes of the layer as a whole, which are set with <code>attribute = value</code> . Available attributes include <code>method</code> , <code>formula</code> , <code>se</code> , <code>method.args</code> , <code>n</code> , <code>span</code> , <code>fullrange</code> , <code>level</code>
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
show.help	If TRUE, display some minimal help.

## Details

Positional aesthetics are typically specified using a formula (see the `gformula` argument). Additional formula terms of the form `+ attribute::value` map `attribute` to `value`. Additional terms of the form `+ attribute:value` will map `attribute` to `value` if `value` is the name of a variable in data, else `attribute` will be set to the constant value. Alternatively (and preferably) attributes can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`. In formulas of the form `A | B`, `B` will be used to form facets using `facet_wrap()` or `facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of **lattice**.

Evaluation 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.

## Value

a gg object

## See Also

[geom\\_smooth\(\)](#)

## Examples

```
if (require(mosaicData)) {
  gf_smooth(births ~ date, color = ~wday, data = Births78)
}
```

---

gf\_spline

*Formula interface to geom\_spline()*


---

## Description

**ggformula** functions provide a formula interface to **ggplot2** layer functions. For plots with just one layer, the formula interface is more compact and is consistent with modeling and **mosaic** notation. The functions generate a **ggplot** command string which can be displayed by setting `verbose = TRUE` as an argument.

## Usage

```
gf_spline(object = NULL, gformula = NULL, data = NULL, geom = type,
  verbose = FALSE, add = inherits(object, c("gg", "ggplot")), ...,
  position = NULL, show.help = NULL)
```

**Arguments**

object	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
gformula	A formula with shape $y \sim x$ . Faceting can be achieved by including <code> </code> in the formula.
data	A data frame with the variables to be plotted.
geom	A way to specify ggplot geoms that are not aliased to gf functions.
verbose	If TRUE print the ggplot2 command in the console.
add	If TRUE then construct just the layer with no frame. The result can be added to an existing frame.
...	Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with <code>attribute = value</code> , (b) ggplot2 aesthetics to be mapped with <code>attribute = ~expression</code> , or (c) attributes of the layer as a whole, which are set with <code>attribute = value</code> . Available attributes include <code>alpha</code> , <code>color</code> , <code>group</code> , <code>linetype</code> , <code>size</code> , <code>weight</code> , <code>df</code> , <code>spar</code> , <code>tol</code>
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
show.help	If TRUE, display some minimal help.

**Details**

Positional aesthetics are typically specified using a formula (see the `gformula` argument). Additional formula terms of the form `+ attribute::value` map `attribute` to `value`. Additional terms of the form `+ attribute:value` will map `attribute` to `value` if `value` is the name of a variable in `data`, else `attribute` will be set to the constant `value`. Alternatively (and preferably) attributes can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`. In formulas of the form `A | B`, `B` will be used to form facets using `facet_wrap()` or `facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of **lattice**.

Evaluation 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.

**Value**

a gg object

**See Also**

[geom\\_spline\(\)](#)

**Examples**

```
if (require(mosaic)) {
  gf_spline(births ~ date, color = ~wday, data = Births78)
  gf_spline(births ~ date, color = ~wday, data = Births78, df = 20)
```

```
gf_spline(births ~ date, color = ~wday, data = Births78, df = 4)
}
```

---

gf\_spoke

*Formula interface to geom\_spoke()*


---

## Description

**ggformula** functions provide a formula interface to ggplot2 layer functions. For plots with just one layer, the formula interface is more compact and is consistent with modeling and **mosaic** notation. The functions generate a ggplot command string which can be displayed by setting `verbose = TRUE` as an argument.

## Usage

```
gf_spoke(object = NULL, gformula = NULL, data = NULL, geom = type,
         verbose = FALSE, add = inherits(object, c("gg", "ggplot")), ...,
         position = NULL, show.help = NULL)
```

## Arguments

object	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
gformula	A formula with shape <code>y ~ x</code> . Faceting can be achieved by including <code> </code> in the formula.
data	A data frame with the variables to be plotted.
geom	A way to specify ggplot geoms that are not aliased to gf functions.
verbose	If TRUE print the ggplot2 command in the console.
add	If TRUE then construct just the layer with no frame. The result can be added to an existing frame.
...	Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with <code>attribute = value</code> , (b) ggplot2 aesthetics to be mapped with <code>attribute = ~expression</code> , or (c) attributes of the layer as a whole, which are set with <code>attribute = value</code> . Available attributes include <code>angle</code> , <code>radius</code> , <code>alpha</code> , <code>color</code> , <code>group</code> , <code>linetype</code> , <code>size</code>
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
show.help	If TRUE, display some minimal help.

## Details

Positional aesthetics are typically specified using a formula (see the `gformula` argument). Additional formula terms of the form `+ attribute::value` map `attribute` to `value`. Additional terms of the form `+ attribute:value` will map `attribute` to `value` if `value` is the name of a variable in data, else `attribute` will be set to the constant value. Alternatively (and preferably) attributes can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`. In formulas of the form `A | B`, `B` will be used to form facets using `facet_wrap()` or `facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of **lattice**.

Evaluation 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.

## Value

a gg object

**Note** `angle` and `radius` must be set or mapped.

NA

## See Also

[geom\\_spoke\(\)](#)

## Examples

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

gf_point(y ~ x, data = D) %>%
  gf_spoke(y ~ x, angle = ~angle, radius = 0.5)

gf_point(y ~ x, data = D) %>%
  gf_spoke(y ~ x, angle = ~angle, radius = ~speed)
```

---

gf\_step

*Formula interface to geom\_step()*

---

## Description

**ggformula** functions provide a formula interface to `ggplot2` layer functions. For plots with just one layer, the formula interface is more compact and is consistent with modeling and **mosaic** notation. The functions generate a `ggplot` command string which can be displayed by setting `verbose = TRUE` as an argument.

**Usage**

```
gf_step(object = NULL, gformula = NULL, data = NULL, geom = type,
        verbose = FALSE, add = inherits(object, c("gg", "ggplot")), ...,
        position = NULL, show.help = NULL)
```

**Arguments**

object	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
gformula	A formula with shape $y \sim x$ . Faceting can be achieved by including <code> </code> in the formula.
data	A data frame with the variables to be plotted.
geom	A way to specify ggplot geoms that are not aliased to gf functions.
verbose	If TRUE print the ggplot2 command in the console.
add	If TRUE then construct just the layer with no frame. The result can be added to an existing frame.
...	Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with <code>attribute = value</code> , (b) ggplot2 aesthetics to be mapped with <code>attribute = ~expression</code> , or (c) attributes of the layer as a whole, which are set with <code>attribute = value</code> . Available attributes include <code>alpha</code> , <code>color</code> , <code>group</code> , <code>linetype</code> , <code>size</code> , <code>direction</code>
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
show.help	If TRUE, display some minimal help.

**Details**

Positional aesthetics are typically specified using a formula (see the `gformula` argument). Additional formula terms of the form `+ attribute::value` map `attribute` to `value`. Additional terms of the form `+ attribute:value` will map `attribute` to `value` if `value` is the name of a variable in `data`, else `attribute` will be set to the constant value. Alternatively (and preferably) attributes can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`. In formulas of the form `A | B`, `B` will be used to form facets using `facet_wrap()` or `facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of **lattice**.

Evaluation 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.

**Value**

a gg object

**See Also**

[geom\\_step\(\)](#)

## Examples

```
if (require(mosaicData)) {
  gf_step( births ~ date, data = Births78, color = ~wday)
}
```

---

gf\_text

*Formula interface to geom\_text()*


---

## Description

**ggformula** functions provide a formula interface to ggplot2 layer functions. For plots with just one layer, the formula interface is more compact and is consistent with modeling and **mosaic** notation. The functions generate a ggplot command string which can be displayed by setting `verbose = TRUE` as an argument.

## Usage

```
gf_text(object = NULL, gformula = NULL, data = NULL, geom = type,
  verbose = FALSE, add = inherits(object, c("gg", "ggplot")), ...,
  position = NULL, show.help = NULL)
```

## Arguments

object	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
gformula	A formula with shape <code>y ~ x</code> . Faceting can be achieved by including <code> </code> in the formula.
data	A data frame with the variables to be plotted.
geom	A way to specify ggplot geoms that are not aliased to gf functions.
verbose	If TRUE print the ggplot2 command in the console.
add	If TRUE then construct just the layer with no frame. The result can be added to an existing frame.
...	Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with <code>attribute = value</code> , (b) ggplot2 aesthetics to be mapped with <code>attribute = ~expression</code> , or (c) attributes of the layer as a whole, which are set with <code>attribute = value</code> . Available attributes include <code>label</code> , <code>alpha</code> , <code>angle</code> , <code>color</code> , <code>family</code> , <code>fontface</code> , <code>group</code> , <code>hjust</code> , <code>lineheight</code> , <code>size</code> , <code>vjust</code> , <code>parse</code> , <code>nudge_x</code> , <code>nudge_y</code> , <code>check_overlap</code>
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
show.help	If TRUE, display some minimal help.

## Details

Positional aesthetics are typically specified using a formula (see the `gformula` argument). Additional formula terms of the form `+ attribute::value` map `attribute` to `value`. Additional terms of the form `+ attribute:value` will map `attribute` to `value` if `value` is the name of a variable in data, else `attribute` will be set to the constant value. Alternatively (and preferably) attributes can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`. In formulas of the form `A | B`, `B` will be used to form facets using `facet_wrap()` or `facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of **lattice**.

Evaluation 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.

## Value

a gg object

## See Also

`geom_text()`

## Examples

```
gf_text(Sepal.Length ~ Sepal.Width, data = iris,
  label = ~Species, color = ~Species, size = 2, angle = 30)
```

---

gf\_theme

*Themes for ggformula*

---

## Description

Themes for ggformula

## Usage

```
gf_theme(object, theme, ...)
```

## Arguments

<code>object</code>	a gg object
<code>theme</code>	a ggplot2 theme function like <code>theme_minimal</code> .
<code>...</code>	If theme is missing, then these additional arguments are theme elements of the sort handled by <code>theme()</code> .

## Value

a modified gg object



---

gf_tile	<i>Formula interface to geom_tile()</i>
---------	-----------------------------------------

---

## Description

**ggformula** functions provide a formula interface to ggplot2 layer functions. For plots with just one layer, the formula interface is more compact and is consistent with modeling and **mosaic** notation. The functions generate a ggplot command string which can be displayed by setting `verbose = TRUE` as an argument.

## Usage

```
gf_tile(object = NULL, gformula = NULL, data = NULL, geom = type,
        verbose = FALSE, add = inherits(object, c("gg", "ggplot")), ...,
        position = NULL, show.help = NULL)
```

## Arguments

object	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
gformula	A formula with shape <code>y ~ x</code> . Faceting can be achieved by including <code> </code> in the formula.
data	A data frame with the variables to be plotted.
geom	A way to specify ggplot geoms that are not aliased to gf functions.
verbose	If TRUE print the ggplot2 command in the console.
add	If TRUE then construct just the layer with no frame. The result can be added to an existing frame.
...	Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with <code>attribute = value</code> , (b) ggplot2 aesthetics to be mapped with <code>attribute = ~expression</code> , or (c) attributes of the layer as a whole, which are set with <code>attribute = value</code> . Available attributes include <code>alpha</code> , <code>color</code> , <code>fill</code> , <code>group</code> , <code>linetype</code> , <code>size</code>
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
show.help	If TRUE, display some minimal help.

## Details

Positional aesthetics are typically specified using a formula (see the `gformula` argument). Additional formula terms of the form `+ attribute::value` map `attribute` to `value`. Additional terms of the form `+ attribute:value` will map `attribute` to `value` if `value` is the name of a variable in `data`, else `attribute` will be set to the constant `value`. Alternatively (and preferably) attributes can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`. In formulas of the form `A | B`, `B` will be used to form facets using `facet_wrap()` or `facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of **lattice**.

Evaluation 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.

### Value

a gg object

### See Also

[geom\\_tile\(\)](#)

### Examples

```
D <- expand.grid(x = 0:5, y = 0:5)
D$z <- runif(nrow(D))
gf_tile(y ~ x, fill = ~ z, data = D)
gf_tile(z ~ x + y, data = D)
```

---

gf\_violin

*Formula interface to geom\_violin()*

---

### Description

**ggformula** functions provide a formula interface to ggplot2 layer functions. For plots with just one layer, the formula interface is more compact and is consistent with modeling and **mosaic** notation. The functions generate a ggplot command string which can be displayed by setting `verbose = TRUE` as an argument.

### Usage

```
gf_violin(object = NULL, gformula = NULL, data = NULL, geom = type,
  verbose = FALSE, add = inherits(object, c("gg", "ggplot")), ...,
  position = NULL, show.help = NULL)
```

### Arguments

object	When chaining, this holds an object produced in the earlier portions of the chain. Most users can safely ignore this argument. See details and examples.
gformula	A formula with shape <code>y ~ x</code> . Faceting can be achieved by including <code> </code> in the formula.
data	A data frame with the variables to be plotted.
geom	A way to specify ggplot geoms that are not aliased to gf functions.
verbose	If TRUE print the ggplot2 command in the console.
add	If TRUE then construct just the layer with no frame. The result can be added to an existing frame.

...	Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with <code>attribute = value</code> , (b) ggplot2 aesthetics to be mapped with <code>attribute = ~expression</code> , or (c) attributes of the layer as a whole, which are set with <code>attribute = value</code> . Available attributes include <code>alpha</code> , <code>color</code> , <code>fill</code> , <code>group</code> , <code>linetype</code> , <code>size</code> , <code>draw_quatiles</code> , <code>trim</code> , <code>scale</code> , <code>bw</code> , <code>adjust</code> , <code>kernel</code>
<code>position</code>	Position adjustment, either as a string, or the result of a call to a position adjustment function.
<code>show.help</code>	If TRUE, display some minimal help.

## Details

Positional aesthetics are typically specified using a formula (see the `gformula` argument). Additional formula terms of the form `+ attribute::value` map `attribute` to `value`. Additional terms of the form `+ attribute:value` will map `attribute` to `value` if `value` is the name of a variable in data, else `attribute` will be set to the constant value. Alternatively (and preferably) attributes can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`. In formulas of the form `A | B`, `B` will be used to form facets using `facet_wrap()` or `facet_grid()`. This provides an alternative to `gf_facet_wrap()` and `gf_facet_grid()` that is terser and may feel more familiar to users of **lattice**.

Evaluation 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.

## Value

a gg object

## See Also

`geom_violin()`

## Examples

```
if (require(mosaicData)) {
  gf_violin(age ~ substance, data = HELPrct)
  gf_violin(age ~ substance, data = HELPrct, fill = ~sex)
}
```

## Description

The functions in **ggformula** provide a formula interface to **ggplot2** layer functions and a system for working with pipes to create multi-layer plots and to refine plots. For plots with just one layer, the formula interface is more compact than native **ggplot2** code and is consistent with modeling functions like **lm()** that use a formula interface and with the numerical summary functions in the **mosaic** package. The functions generate a **ggplot** command string which can be displayed by setting `verbose = TRUE` as an argument.

## Details

Positional aesthetics are typically specified using a formula (see the `gformula` argument). Setting and mapping of additional attributes can be done within the formula or through the use of additional arguments. The latter is considered preferable. Attributes can be set using arguments of the form `attribute = value` or mapped using arguments of the form `attribute = ~ expression`. Additional formula terms of the form `+ attribute::value` map `attribute` to `value`; terms of the form `+ attribute:value` will map `attribute` to `value` if `value` is the name of a variable in data, else `attribute` will be set to the constant value.

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

Evaluation 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.

## Examples

```
apropos("gf_")  
gf_point()
```

# Index

df\_stats, 3

facet\_grid, 6, 8, 10–12, 14, 15, 17, 19, 20, 22–24, 26, 27, 29, 31, 33, 34, 36, 37, 39, 41, 43–45, 47, 49, 50, 52, 53, 55, 56, 58, 59, 61, 62, 64, 65, 67, 68

facet\_wrap, 6, 8, 10–12, 14, 15, 17, 19, 20, 22–24, 26, 27, 29, 31, 33, 34, 36, 37, 39, 41, 43–45, 47, 49, 50, 52, 53, 55, 56, 58, 59, 61, 62, 64, 65, 67, 68

favstats, 3

geom\_abline, 5

geom\_area, 6

geom\_bar, 8

geom\_blank, 29

geom\_boxplot, 10

geom\_col, 11

geom\_contour, 13

geom\_count, 14

geom\_crossbar, 15

geom\_curve, 17

geom\_density, 20

geom\_density2d, 22

geom\_density\_2d, 23

geom\_dotplot, 25

geom\_errorbar, 26

geom\_errorbarh, 28

geom\_freqpoly, 31

geom\_hex, 33

geom\_histogram, 34

geom\_hline, 5

geom\_jitter, 36

geom\_label, 37

geom\_line, 19, 39

geom\_linerange, 41

geom\_path, 43

geom\_point, 44

geom\_pointrange, 46

geom\_qq, 47

geom\_quantile, 49

geom\_raster, 50

geom\_rect, 52

geom\_ribbon, 53

geom\_rug, 55

geom\_segment, 56

geom\_smooth, 58

geom\_spline, 59

geom\_spoke, 61

geom\_step, 62

geom\_text, 64

geom\_tile, 66

geom\_violin, 67

geom\_vline, 5

gf\_abline, 4

gf\_area, 5

gf\_bar, 7

gf\_boxplot, 9

gf\_coefline (gf\_abline), 4

gf\_col, 10

gf\_contour, 12

gf\_count, 13

gf\_counts (gf\_bar), 7

gf\_crossbar, 14

gf\_curve, 16

gf\_dens, 18

gf\_density, 19

gf\_density2d, 21

gf\_density\_2d, 22

gf\_dotplot, 23

gf\_errorbar, 25

gf\_errorbarh, 27

gf\_facet\_grid, 6, 8, 10–12, 14, 15, 17, 19, 20, 22–24, 26, 27, 29, 31, 33, 34, 36, 37, 39, 41, 43–45, 47, 49, 50, 52, 53, 55, 56, 58, 59, 61, 62, 64, 65, 67, 68

gf\_facet\_grid (gf\_labs), 38

gf\_facet\_wrap, 6, 8, 10–12, 14, 15, 17, 19, 20, 22–24, 26, 27, 29, 31, 33, 34, 36,

37, 39, 41, 43–45, 47, 49, 50, 52, 53,  
55, 56, 58, 59, 61, 62, 64, 65, 67, 68

`gf_facet_wrap` (`gf_labs`), 38

`gf_frame`, 28

`gf_freqpoly`, 30

`gf_fun` (`gf_function`), 31

`gf_function`, 31

`gf_hex`, 32

`gf_histogram`, 33

`gf_hline` (`gf_abline`), 4

`gf_jitter`, 35

`gf_label`, 36

`gf_labs`, 38

`gf_lims` (`gf_labs`), 38

`gf_line`, 38

`gf_linerange`, 40

`gf_lm` (`gf_smooth`), 57

`gf_path`, 42

`gf_point`, 43

`gf_pointrange`, 45

`gf_qq`, 46

`gf_quantile`, 48

`gf_raster`, 49

`gf_rect`, 51

`gf_refine` (`gf_labs`), 38

`gf_ribbon`, 52

`gf_rug`, 54

`gf_segment`, 55

`gf_smooth`, 57

`gf_spline`, 58

`gf_spoke`, 60

`gf_step`, 61

`gf_text`, 63

`gf_theme`, 64

`gf_tile`, 65

`gf_violin`, 66

`gf_vline` (`gf_abline`), 4

`ggformula`, 67

`lm`, 68

`make.names`, 3

`ntiles`, 3

`summarise`, 3

`theme`, 64

`theme_minimal`, 64