Package: ggdibbler (via r-universe)

May 13, 2026

Type Package

Title Add Uncertainty to Data Visualisations

Version 0.6.5.9000

Maintainer Harriet Mason <harriet.m.mason@gmail.com>

Description A 'ggplot2' extension for visualising uncertainty with the goal of signal suppression. Usually, uncertainty visualisation focuses on expressing uncertainty as a distribution or probability, whereas 'ggdibbler' differentiates itself by viewing an uncertainty visualisation as an adjustment to an existing graphic that incorporates the inherent uncertainty in the estimates. You provide the code for an existing plot, but replace any of the variables with a vector of distributions, and it will convert the visualisation into it's signal suppression counterpart.

License GPL-3

URL https://harriet-mason.github.io/ggdibbler/, https://github.com/harriet-mason/ggdibbler

Depends R (>= 4.1.0)

Imports distributional, dplyr, ggplot2, rlang, lifecycle, scales, tidyr, tibble, cli, sf

Suggests knitr, rmarkdown, testthat (>= 3.0.0), vdiffr, mgcv, fable, urca, gganimate, tidyverse, tidygraph, ggthemes, gifski, ggridges, quantreg, ggdist, ggraph, feasts, patchwork

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

BugReports https://github.com/harriet-mason/ggdibbler/issues

Config/pak/sysreqs libabsl-dev cmake libgdal-dev gdal-bin libgeos-dev libicu-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev

Repository https://harriet-mason.r-universe.dev

Date/Publication 2026-05-13 14:18:31 UTC

RemoteUrl https://github.com/harriet-mason/ggdibbler

RemoteRef HEAD

RemoteSha fe4939729bdead79ccd2c4edf372ab9b1e1fa2a5

Contents

`geom_abline_sample` *Reference lines with uncertainty: horizontal, vertical, and diagonal*

Description

Identical to geom_vline, geom_hline and geom_abline, except that it will accept a distribution in place of any of the usual aesthetics.

Usage

```
geom_abline_sample(
  mapping = NULL,
  data = NULL,
  stat = "identity_sample",
  times = 10,
  seed = NULL,
  ...,
  slope,
  intercept,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = FALSE
)

geom_hline_sample(
  mapping = NULL,
  data = NULL,
  stat = "identity_sample",
  position = "identity",
  ...,
```

```
  seed = NULL,
  times = 10,
  yintercept,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = FALSE
)

geom_vline_sample(
  mapping = NULL,
  data = NULL,
  stat = "identity_sample",
  position = "identity",
  ...,
  times = 10,
  seed = NULL,
  xintercept,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = FALSE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes().
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
stat	The statistical transformation to use on the data for this layer. When using a geom_*() function to construct a layer, the stat argument can be used to override the default coupling between geoms and stats. The stat argument accepts the following: • A Stat ggproto subclass, for example StatCount. • A string naming the stat. To give the stat as a string, strip the function name of the stat_ prefix. For example, to use stat_count(), give the stat as "count". • For more information and other ways to specify the stat, see the layer stat documentation.
times	A parameter used to control the number of values sampled from each distribution.

`seed` Set the seed for the layers random draw, allows you to plot the same draw across multiple layers.

`...` Other arguments passed on to `layer()`'s `params` argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the `position` argument, or aesthetics that are required can *not* be passed through `...`. Unknown arguments that are not part of the 4 categories below are ignored.

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

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

`show.legend` logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use `TRUE`. If `NA`, all levels are shown in legend, but unobserved levels are omitted.

`inherit.aes` If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. `annotation_borders()`.

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

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

`xintercept`, `yintercept`, `slope`, `intercept`

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

Value

A ggplot2 layer

Examples

```
# load libraries
library(ggplot2)
library(distributional)

# ggplot
p <- ggplot(mtcars, aes(wt, mpg)) + geom_point()
# ggdibbler
q <- ggplot(uncertain_mtcars, aes(wt, mpg)) + geom_point_sample(alpha=0.3)

# ggplot
p + geom_abline(intercept = 20) # ggplot
q + geom_abline_sample(intercept = dist_normal(20, 1), alpha=0.3) # ggdibbler
p + geom_vline(xintercept = 5) # ggplot
q + geom_vline_sample(xintercept = dist_normal(5, 0.1), alpha=0.3) # ggdibbler
p + geom_hline(yintercept = 20) # ggplot
q + geom_hline_sample(yintercept = dist_normal(20, 1), alpha=0.3) # ggdibbler

# Calculate slope and intercept of line of best fit
# get coef and standard error
summary(lm(mpg ~ wt, data = mtcars))
# ggplot for coef
p + geom_abline(intercept = 37, slope = -5) # ggplot
# ggdibbler for coef AND standard error
p + geom_abline_sample(intercept = dist_normal(37, 1.8),
  slope = dist_normal(-5, 0.56),
  times=30, alpha=0.3) # ggplot
```

---

geom_bar_sample *Uncertain Bar Charts*

Description

Identical to geom_bar, except that it will accept a distribution in place of any of the usual aesthetics.

Usage

```
geom_bar_sample(
  mapping = NULL,
  data = NULL,
  stat = "count_sample",
  position = "stack_dodge",
  ...,
  just = 0.5,
  times = 10,
```

```
  seed = NULL,
  lineend = "butt",
  linejoin = "mitre",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

geom_col_sample(
  mapping = NULL,
  data = NULL,
  stat = "identity_sample",
  position = "stack_dodge",
  ...,
  just = 0.5,
  times = 10,
  seed = NULL,
  lineend = "butt",
  linejoin = "mitre",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

stat_count_sample(
  mapping = NULL,
  data = NULL,
  geom = "bar",
  position = "stack_identity",
  ...,
  orientation = NA,
  times = 10,
  seed = NULL,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be

created.

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

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

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

`...` Other arguments passed on to `layer()`'s `params` argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the `position` argument, or aesthetics that are required can *not* be passed through `...`. Unknown arguments that are not part of the 4 categories below are ignored.

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

`just` Adjustment for column placement. Set to `0.5` by default, meaning that columns will be centered about axis breaks. Set to `0` or `1` to place columns to the left/right of axis breaks. Note that this argument may have unintended behaviour when used with alternative positions, e.g. `position_dodge()`.

`times` A parameter used to control the number of values sampled from each distribution.

`seed` Set the seed for the layers random draw, allows you to plot the same draw across multiple layers.

`lineend` Line end style (round, butt, square).

linejoin	Line join style (round, mitre, bevel).
na.rm	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. annotation_borders().
geom, stat	Override the default connection between geom_bar() and stat_count(). For more information about overriding these connections, see how the stat and geom arguments work.
orientation	The orientation of the layer. The default (NA) automatically determines the orientation from the aesthetic mapping. In the rare event that this fails it can be given explicitly by setting orientation to either "x" or "y". See the *Orientation* section for more detail.

Value

A ggplot2 layer

Examples

```
library(distributional)
library(ggplot2)

# Set up data
g <- ggplot(mpg, aes(class)) #ggplot
q <- ggplot(uncertain_mpg, aes(class)) #ggdibbler

# Number of cars in each class:
g + geom_bar() #ggplot
q + geom_bar_sample() #ggdibbler - a
q + geom_bar_sample(position = "identity_identity", alpha=0.1) #ggdibbler - b

# make dataframe
df <- data.frame(trt = c("a", "b", "c"), outcome = c(2.3, 1.9, 3.2))
uncertain_df <-  data.frame(trt = c("a", "b", "c"),
                            outcome = dist_normal(mean = c(2.3, 1.9, 3.2),
                                                  sd = c(0.5, 0.8, 0.7)))
# geom_col also has a sample counterpart
# ggplot
ggplot(df, aes(trt, outcome)) + geom_col()
# ggdibbler
ggplot(uncertain_df, aes(x=trt, y=outcome)) + geom_col_sample()

# ggplot
ggplot(mpg, aes(y = class)) +
```

```
  geom_bar(aes(fill = drv), position = position_stack(reverse = TRUE)) +
  theme(legend.position = "top")
# ggdibbler
ggplot(uncertain_mpg, aes(y = class)) +
  geom_bar_sample(aes(fill = drv), alpha=1,
                  position = position_stack_dodge(reverse = TRUE)) +
  theme(legend.position = "top")
```

`geom_bin_2d_sample` *Uncertain heatmap of 2d bin counts*

Description

Identical to geom_bin_2d, except that it will accept a distribution in place of any of the usual aesthetics.

Usage

```
geom_bin_2d_sample(
  mapping = NULL,
  data = NULL,
  stat = "bin2d_sample",
  position = "identity_dodge",
  ...,
  times = 10,
  seed = NULL,
  lineend = "butt",
  linejoin = "mitre",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

stat_bin_2d_sample(
  mapping = NULL,
  data = NULL,
  geom = "tile",
  position = "identity_dodge",
  ...,
  times = 10,
  seed = NULL,
  binwidth = NULL,
  bins = 30,
  breaks = NULL,
  drop = TRUE,
  boundary = NULL,
  closed = NULL,
```

```
  center = NULL,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.
...	Other arguments passed on to layer()'s params argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the position argument, or aesthetics that are required can *not* be passed through .... Unknown arguments that are not part of the 4 categories below are ignored. • Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, colour = "red" or linewidth = 3. The geom's documentation has an **Aesthetics** section that lists the available options. The 'required' aesthetics cannot be passed on to the params. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data. • When constructing a layer using a stat_*() function, the ... argument can be used to pass on parameters to the geom part of the layer. An example of this is stat_density(geom = "area", outline.type = "both"). The geom's documentation lists which parameters it can accept.

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

`times` A parameter used to control the number of values sampled from each distribution.

`seed` Set the seed for the layers random draw, allows you to plot the same draw across multiple layers.

`lineend` Line end style (round, butt, square).

`linejoin` Line join style (round, mitre, bevel).

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

`show.legend` logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use `TRUE`. If `NA`, all levels are shown in legend, but unobserved levels are omitted.

`inherit.aes` If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. `annotation_borders()`.

`geom, stat` Use to override the default connection between `geom_bin_2d()` and `stat_bin_2d()`. For more information about overriding these connections, see how the stat and geom arguments work.

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

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

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

`breaks` Alternatively, you can supply a numeric vector giving the bin boundaries. Overrides `binwidth`, `bins`, `center`, and `boundary`. Can also be a function that takes group-wise values as input and returns bin boundaries.

`drop` if `TRUE` removes all cells with 0 counts.

`closed` One of `"right"` or `"left"` indicating whether right or left edges of bins are included in the bin.

`center, boundary`

bin position specifiers. Only one, `center` or `boundary`, may be specified for a single plot. `center` specifies the center of one of the bins. `boundary` specifies

the boundary between two bins. Note that if either is above or below the range of the data, things will be shifted by the appropriate integer multiple of `binwidth`. For example, to center on integers use `binwidth = 1` and `center = 0`, even if `0` is outside the range of the data. Alternatively, this same alignment can be specified with `binwidth = 1` and `boundary = 0.5`, even if `0.5` is outside the range of the data.

Value

A ggplot2 layer

Examples

```
# ggplot
library(ggplot2)
d <- ggplot(smaller_diamonds, aes(x, y))
d + geom_bin_2d()
# ggdibbler
b <- ggplot(smaller_uncertain_diamonds, aes(x, y))
# the ggdibbler default position adjustment is dodging
b + geom_bin_2d_sample(times=100)
# but it can change it to be transparency
b + geom_bin_2d_sample(position="identity", alpha=0.1)
# Still have the same options
d + geom_bin_2d(bins = 10) #ggplot
b + geom_bin_2d_sample(bins = 10) #ggdibbler
```

---

geom_boxplot_sample *An uncertain box and whiskers plot (in the style of Tukey)*

Description

Identical to geom_boxplot, except that it will accept a distribution in place of any of the usual aesthetics.

Usage

```
geom_boxplot_sample(
  mapping = NULL,
  data = NULL,
  times = 10,
  seed = NULL,
  stat = "boxplot_sample",
  position = "identity",
  ...,
  outliers = TRUE,
  outlier.colour = NULL,
  outlier.color = NULL,
```

```
  outlier.fill = NULL,
  outlier.shape = NULL,
  outlier.size = NULL,
  outlier.stroke = 0.5,
  outlier.alpha = NULL,
  whisker.colour = NULL,
  whisker.color = NULL,
  whisker.linetype = NULL,
  whisker.linewidth = NULL,
  staple.colour = NULL,
  staple.color = NULL,
  staple.linetype = NULL,
  staple.linewidth = NULL,
  median.colour = NULL,
  median.color = NULL,
  median.linetype = NULL,
  median.linewidth = NULL,
  box.colour = NULL,
  box.color = NULL,
  box.linetype = NULL,
  box.linewidth = NULL,
  notch = FALSE,
  notchwidth = 0.5,
  staplewidth = 0,
  varwidth = FALSE,
  na.rm = FALSE,
  orientation = NA,
  show.legend = NA,
  inherit.aes = TRUE
)

stat_boxplot_sample(
  mapping = NULL,
  data = NULL,
  geom = "boxplot",
  position = "identity",
  ...,
  times = 10,
  orientation = NA,
  seed = NULL,
  coef = 1.5,
  quantile.type = 7,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
times	A parameter used to control the number of values sampled from each distribution.
seed	Set the seed for the layers random draw, allows you to plot the same draw across multiple layers.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.
...	Other arguments passed on to layer()'s params argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the position argument, or aesthetics that are required can *not* be passed through .... Unknown arguments that are not part of the 4 categories below are ignored. • Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, colour = "red" or linewidth = 3. The geom's documentation has an **Aesthetics** section that lists the available options. The 'required' aesthetics cannot be passed on to the params. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data. • When constructing a layer using a stat_*() function, the ... argument can be used to pass on parameters to the geom part of the layer. An example of this is stat_density(geom = "area", outline.type = "both"). The geom's documentation lists which parameters it can accept. • Inversely, when constructing a layer using a geom_*() function, the ... argument can be used to pass on parameters to the stat part of the layer. An example of this is geom_area(stat = "density", adjust = 0.5). The stat's documentation lists which parameters it can accept.

- The key_glyph argument of layer() may also be passed on through .... This can be one of the functions described as key glyphs, to change the display of the layer in the legend.

outliers
: Whether to display (TRUE) or discard (FALSE) outliers from the plot. Hiding or discarding outliers can be useful when, for example, raw data points need to be displayed on top of the boxplot. By discarding outliers, the axis limits will adapt to the box and whiskers only, not the full data range. If outliers need to be hidden and the axes needs to show the full data range, please use outlier.shape = NA instead.

outlier.colour, outlier.color, outlier.fill, outlier.shape, outlier.size, outlier.stroke, outlier.alpha
: Default aesthetics for outliers. Set to NULL to inherit from the data's aesthetics.

whisker.colour, whisker.color, whisker.linetype, whisker.linewidth
: Default aesthetics for the whiskers. Set to NULL to inherit from the data's aesthetics.

staple.colour, staple.color, staple.linetype, staple.linewidth
: Default aesthetics for the staples. Set to NULL to inherit from the data's aesthetics. Note that staples don't appear unless the staplewidth argument is set to a non-zero size.

median.colour, median.color, median.linetype, median.linewidth
: Default aesthetics for the median line. Set to NULL to inherit from the data's aesthetics.

box.colour, box.color, box.linetype, box.linewidth
: Default aesthetics for the boxes. Set to NULL to inherit from the data's aesthetics.

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

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

staplewidth
: The relative width of staples to the width of the box. Staples mark the ends of the whiskers with a line.

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

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

orientation
: The orientation of the layer. The default (NA) automatically determines the orientation from the aesthetic mapping. In the rare event that this fails it can be given explicitly by setting orientation to either "x" or "y". See the *Orientation* section for more detail.

show.legend
: logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.

inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. annotation_borders().
geom, stat	Use to override the default connection between geom_boxplot() and stat_boxplot(). For more information about overriding these connections, see how the stat and geom arguments work.
coef	Length of the whiskers as multiple of IQR. Defaults to 1.5.
quantile.type	An integer between 1 and 9 setting the quantile algorithm per stats::quantile(type). Defaults to 7

Value

A ggplot2 layer

Examples

```
library(ggplot2)
# ggplot
p <- ggplot(mpg, aes(class, hwy))
p + geom_boxplot(alpha=0.5)

# using alpha to manage overplotting
q <- ggplot(uncertain_mpg, aes(class, hwy))
q + geom_boxplot_sample(alpha=0.1)

# ggplot
p + geom_boxplot(varwidth = TRUE)
# ggdibbler
q + geom_boxplot_sample(alpha=0.1, varwidth = TRUE)

# ggplot
p + geom_boxplot(aes(colour = drv), position = position_dodge(preserve = "single"))
# ggdibbler
q + geom_boxplot_sample(aes(colour = drv), alpha=0.05, position = "dodge_identity")
```

---

geom_contour_sample *Uncertain 2D contours of a 3D surface*

Description

Identical to geom_contour and geom_contour_filled, except that it will accept a distribution in place of any of the usual aesthetics.

Usage

```
geom_contour_sample(
  mapping = NULL,
  data = NULL,
  stat = "contour_sample",
  position = "identity",
  ...,
  times = 10,
  seed = NULL,
  bins = NULL,
  binwidth = NULL,
  breaks = NULL,
  arrow = NULL,
  arrow.fill = NULL,
  lineend = "butt",
  linejoin = "round",
  linemitre = 10,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

geom_contour_filled_sample(
  mapping = NULL,
  data = NULL,
  stat = "contour_filled_sample",
  position = "identity",
  ...,
  times = 10,
  seed = NULL,
  bins = NULL,
  binwidth = NULL,
  breaks = NULL,
  rule = "evenodd",
  lineend = "butt",
  linejoin = "round",
  linemitre = 10,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

stat_contour_sample(
  mapping = NULL,
  data = NULL,
  geom = "contour",
  position = "identity",
  ...,
```

```
  times = 10,
  seed = NULL,
  bins = NULL,
  binwidth = NULL,
  breaks = NULL,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

stat_contour_filled_sample(
  mapping = NULL,
  data = NULL,
  geom = "contour_filled",
  position = "identity",
  ...,
  times = 10,
  seed = NULL,
  bins = NULL,
  binwidth = NULL,
  breaks = NULL,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
stat	The statistical transformation to use on the data for this layer. When using a geom_*() function to construct a layer, the stat argument can be used to override the default coupling between geoms and stats. The stat argument accepts the following: • A Stat ggproto subclass, for example StatCount. • A string naming the stat. To give the stat as a string, strip the function name of the stat_ prefix. For example, to use stat_count(), give the stat as "count".

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

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

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

`...` Other arguments passed on to `layer()`'s `params` argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the `position` argument, or aesthetics that are required can *not* be passed through `...`. Unknown arguments that are not part of the 4 categories below are ignored.

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

`times` A parameter used to control the number of values sampled from each distribution.

`seed` Set the seed for the layers random draw, allows you to plot the same draw across multiple layers.

`bins` Number of contour bins. Overridden by `breaks`.

`binwidth` The width of the contour bins. Overridden by `bins`.

`breaks` One of:

- Numeric vector to set the contour breaks
- A function that takes the range of the data and binwidth as input and returns breaks as output. A function can be created from a formula (e.g. ~ fullseq(.x, .y)).

	Overrides `binwidth` and `bins`. By default, this is a vector of length ten with `pretty()` breaks.
`arrow`	Arrow specification, as created by `grid::arrow()`.
`arrow.fill`	fill colour to use for the arrow head (if closed). `NULL` means use `colour` aesthetic.
`lineend`	Line end style (round, butt, square).
`linejoin`	Line join style (round, mitre, bevel).
`linemitre`	Line mitre limit (number greater than 1).
`na.rm`	If `FALSE`, the default, missing values are removed with a warning. If `TRUE`, missing values are silently removed.
`show.legend`	logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use `TRUE`. If `NA`, all levels are shown in legend, but unobserved levels are omitted.
`inherit.aes`	If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. `annotation_borders()`.
`rule`	Either `"evenodd"` or `"winding"`. If polygons with holes are being drawn (using the `subgroup` aesthetic) this argument defines how the hole coordinates are interpreted. See the examples in `grid::pathGrob()` for an explanation.
`geom`	The geometric object to use to display the data for this layer. When using a `stat_*()` function to construct a layer, the `geom` argument can be used to override the default coupling between stats and geoms. The `geom` argument accepts the following: • A `Geom` ggproto subclass, for example `GeomPoint`. • A string naming the geom. To give the geom as a string, strip the function name of the `geom_` prefix. For example, to use `geom_point()`, give the geom as `"point"`. • For more information and other ways to specify the geom, see the layer geom documentation.

Value

A ggplot2 layer

Examples

```
library(ggplot2)
library(dplyr)
faithfuld
# ggplot2
v <- ggplot(faithfuld |>
  filter(waiting>80) |>
  filter(eruptions >3),
  aes(waiting, eruptions, z = density))
```

```
v + geom_contour()
# ggdibbler
u <- ggplot(uncertain_faithfuld |>
  filter(waiting>80) |>
  filter(eruptions >3),
  aes(waiting, eruptions, z = density))
u + geom_contour_sample()

# use geom_contour_filled() for filled contours
# ggplot2
v + geom_contour_filled() # no error (point prediction)
# ggdibbler
u + geom_contour_filled_sample()
```

`geom_count_sample` *Uncertain Count overlapping points*

Description

Identical to geom_count, except that it will accept a distribution in place of any of the usual aesthetics.

Usage

```
geom_count_sample(
  mapping = NULL,
  data = NULL,
  stat = "sum_sample",
  position = "identity",
  ...,
  times = 10,
  seed = NULL,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

stat_sum_sample(
  mapping = NULL,
  data = NULL,
  geom = "point",
  position = "identity",
  ...,
  times = 10,
  seed = NULL,
  na.rm = FALSE,
  show.legend = NA,
```

```
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.
...	Other arguments passed on to layer()'s params argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the position argument, or aesthetics that are required can *not* be passed through .... Unknown arguments that are not part of the 4 categories below are ignored. • Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, colour = "red" or linewidth = 3. The geom's documentation has an **Aesthetics** section that lists the available options. The 'required' aesthetics cannot be passed on to the params. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data. • When constructing a layer using a stat_*() function, the ... argument can be used to pass on parameters to the geom part of the layer. An example of this is stat_density(geom = "area", outline.type = "both"). The geom's documentation lists which parameters it can accept. • Inversely, when constructing a layer using a geom_*() function, the ... argument can be used to pass on parameters to the stat part of the layer. An example of this is geom_area(stat = "density", adjust = 0.5). The stat's documentation lists which parameters it can accept.

- The key_glyph argument of layer() may also be passed on through .... This can be one of the functions described as key glyphs, to change the display of the layer in the legend.

times	A parameter used to control the number of values sampled from each distribution.
seed	Set the seed for the layers random draw, allows you to plot the same draw across multiple layers.
na.rm	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. annotation_borders().
geom, stat	Use to override the default connection between geom_count() and stat_sum(). For more information about overriding these connections, see how the stat and geom arguments work.

Value

A ggplot2 layer

Examples

```
library(ggplot2)

# Discrete values have overplotting
# ggplot
ggplot(mpg, aes(cty, hwy)) +
  geom_point()
# ggdibbler
ggplot(uncertain_mpg, aes(cty, hwy)) +
  geom_point_sample()

# Can use geom_count to fix it
# ggplot
ggplot(mpg, aes(cty, hwy)) +
  geom_count()
# ggdibbler (alpha for resample overlap)
ggplot(uncertain_mpg, aes(cty, hwy)) +
  geom_count_sample(alpha=0.15)

# Best used in conjunction with scale_size_area
# ggplot
ggplot(mpg, aes(cty, hwy)) +
  geom_count() +
```

```
  scale_size_area()
# ggdibbler
ggplot(uncertain_mpg, aes(cty, hwy)) +
  geom_count_sample(alpha=0.15) +
  scale_size_area()
```

`geom_crossbar_sample` *Vertical intervals: lines, crossbars & errorbars with uncertainty*

Description

Identical to geom_linerange, geom_errorbar, geom_crossbar, and geom_pointrange except that they will accept a distribution in place of any of the usual aesthetics.

Usage

```
geom_crossbar_sample(
  mapping = NULL,
  data = NULL,
  times = 10,
  seed = NULL,
  stat = "identity_sample",
  position = "identity",
  ...,
  middle.colour = NULL,
  middle.color = NULL,
  middle.linetype = NULL,
  middle.linewidth = NULL,
  box.colour = NULL,
  box.color = NULL,
  box.linetype = NULL,
  box.linewidth = NULL,
  fatten = deprecated(),
  na.rm = FALSE,
  orientation = NA,
  show.legend = NA,
  inherit.aes = TRUE
)

geom_errorbar_sample(
  mapping = NULL,
  data = NULL,
  stat = "identity_sample",
  position = "identity",
  ...,
  times = 10,
  orientation = NA,
```

```
  seed = NULL,
  lineend = "butt",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

geom_linerange_sample(
  mapping = NULL,
  data = NULL,
  stat = "identity_sample",
  position = "identity",
  ...,
  times = 10,
  orientation = NA,
  seed = NULL,
  lineend = "butt",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

geom_pointrange_sample(
  mapping = NULL,
  data = NULL,
  stat = "identity_sample",
  position = "identity",
  ...,
  times = 10,
  orientation = NA,
  seed = NULL,
  lineend = "butt",
  fatten = 4,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be

created.

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

times — A parameter used to control the number of values sampled from each distribution.

seed — Set the seed for the layers random draw, allows you to plot the same draw across multiple layers.

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

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

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

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

... — Other arguments passed on to layer()'s params argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the position argument, or aesthetics that are required can *not* be passed through .... Unknown arguments that are not part of the 4 categories below are ignored.

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

An example of this is `geom_area(stat = "density", adjust = 0.5)`. The stat's documentation lists which parameters it can accept.

- The `key_glyph` argument of `layer()` may also be passed on through `...`. This can be one of the functions described as key glyphs, to change the display of the layer in the legend.

`middle.colour`, `middle.color`, `middle.linetype`, `middle.linewidth`
: Default aesthetics for the middle line. Set to `NULL` to inherit from the data's aesthetics.

`box.colour`, `box.color`, `box.linetype`, `box.linewidth`
: Default aesthetics for the boxes. Set to `NULL` to inherit from the data's aesthetics.

`fatten`
: **[Deprecated]** A multiplicative factor used to increase the size of the middle bar in `geom_crossbar()` and the middle point in `geom_pointrange()`.

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

`orientation`
: The orientation of the layer. The default (`NA`) automatically determines the orientation from the aesthetic mapping. In the rare event that this fails it can be given explicitly by setting `orientation` to either `"x"` or `"y"`. See the *Orientation* section for more detail.

`show.legend`
: logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use `TRUE`. If `NA`, all levels are shown in legend, but unobserved levels are omitted.

`inherit.aes`
: If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. `annotation_borders()`.

`lineend`
: Line end style (round, butt, square).

Value

A ggplot2 layer

Examples

```
library(ggplot2)
library(dplyr)
library(distributional)
# Create a simple example dataset

df <- data.frame(
  trt = factor(c(1, 1, 2, 2)),
  resp = c(1, 5, 3, 4),
  group = factor(c(1, 2, 1, 2)),
  upper = c(1.1, 5.3, 3.3, 4.2),
  lower = c(0.8, 4.6, 2.4, 3.6)
)

uncertain_df <- df |>
```

```
  group_by(trt, group) |>
  mutate(resp = dist_normal(resp, runif(1,0,0.2)),
         upper = dist_normal(upper, runif(1,0,0.2)),
         lower = dist_normal(lower, runif(1,0,0.2))
  )

p <- ggplot(df, aes(trt, resp, colour = group))
q <- ggplot(uncertain_df, aes(trt, resp, colour = group))

# ggplot
p + geom_linerange(aes(ymin = lower, ymax = upper), linewidth=4)
#ggdibbler
q + geom_linerange_sample(aes(ymin = lower, ymax = upper), linewidth=4)

# ggplot
p + geom_pointrange(aes(ymin = lower, ymax = upper))
# ggdibbler
q + geom_pointrange_sample(aes(ymin = lower, ymax = upper))

# ggplot
p + geom_crossbar(aes(ymin = lower, ymax = upper), width = 0.2)
# ggdibbler
q + geom_crossbar_sample(aes(ymin = lower, ymax = upper), width = 0.2)

# ggplot
p + geom_errorbar(aes(ymin = lower, ymax = upper), width = 0.2)
# ggdibbler
q + geom_errorbar_sample(aes(ymin = lower, ymax = upper), width = 0.2)
```

`geom_curve_sample` *Line segments and curves with uncertainty*

Description

Identical to geom_segment, except that it will accept a distribution in place of any of the usual aesthetics.

Usage

```
geom_curve_sample(
  mapping = NULL,
  data = NULL,
  stat = "identity_sample",
  position = "identity",
  ...,
  times = 10,
  seed = NULL,
  curvature = 0.5,
  angle = 90,
```

```
  ncp = 5,
  arrow = NULL,
  arrow.fill = NULL,
  lineend = "butt",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

geom_segment_sample(
  mapping = NULL,
  data = NULL,
  stat = "identity_sample",
  position = "identity",
  ...,
  times = 10,
  seed = NULL,
  arrow = NULL,
  arrow.fill = NULL,
  lineend = "butt",
  linejoin = "round",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
stat	The statistical transformation to use on the data for this layer. When using a geom_*() function to construct a layer, the stat argument can be used to override the default coupling between geoms and stats. The stat argument accepts the following: • A Stat ggproto subclass, for example StatCount. • A string naming the stat. To give the stat as a string, strip the function name of the stat_ prefix. For example, to use stat_count(), give the stat as "count".

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

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

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

`...` Other arguments passed on to `layer()`'s `params` argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the `position` argument, or aesthetics that are required can *not* be passed through `...`. Unknown arguments that are not part of the 4 categories below are ignored.

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

`times` A parameter used to control the number of values sampled from each distribution.

`seed` Set the seed for the layers random draw, allows you to plot the same draw across multiple layers.

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

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

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

arrow	specification for arrow heads, as created by grid::arrow().
arrow.fill	fill colour to use for the arrow head (if closed). NULL means use colour aesthetic.
lineend	Line end style (round, butt, square).
na.rm	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. annotation_borders().
linejoin	Line join style (round, mitre, bevel).

Value

A ggplot2 layer

Examples

```
library(ggplot2)
library(distributional)
# ggplot
b <- ggplot(mtcars, aes(wt, mpg)) +
  geom_point()
# ggdibbler
a <- ggplot(uncertain_mtcars, aes(wt, mpg)) +
  geom_point_sample(seed=77, alpha=0.5)


df <- data.frame(x1 = 2.62, x2 = 3.57,
                 y1 = 21.0, y2 = 15.0)
uncertain_df <- data.frame(x1 = dist_normal(2.62, 0.1),
                           x2 = dist_normal(3.57,0.1),
                           y1 = dist_normal(21.0, 0.1),
                           y2 = dist_normal(15.0,0.1))
# ggplot
b +
  geom_curve(aes(x = x1, y = y1, xend = x2, yend = y2, colour = "curve"), data = df) +
  geom_segment(aes(x = x1, y = y1, xend = x2, yend = y2, colour = "segment"), data = df)

# ggdibbler
a +
  geom_curve_sample(aes(x = x1, y = y1, xend = x2, yend = y2, colour = "curve"),
                    data = uncertain_df, seed=77, alpha=0.5) +
  geom_segment_sample(aes(x = x1, y = y1, xend = x2, yend = y2, colour = "segment"),
                      data = uncertain_df, seed=77, alpha=0.5)
```

geom_density_2d_sample *Uncertain contours of a 2D density estimate*

Description

Identical to geom_density_2d() and geom_density_2d_filled, except that it will accept a distribution in place of any of the usual aesthetics.

Usage

```
geom_density_2d_sample(
  mapping = NULL,
  data = NULL,
  stat = "density_2d_sample",
  position = "identity",
  ...,
  times = 10,
  seed = NULL,
  arrow = NULL,
  arrow.fill = NULL,
  lineend = "butt",
  linejoin = "round",
  linemitre = 10,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

geom_density_2d_filled_sample(
  mapping = NULL,
  data = NULL,
  stat = "density_2d_filled_sample",
  position = "identity",
  ...,
  times = 10,
  seed = NULL,
  rule = "evenodd",
  lineend = "butt",
  linejoin = "round",
  linemitre = 10,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

stat_density_2d_sample(
```

```
  mapping = NULL,
  data = NULL,
  geom = "density_2d",
  position = "identity",
  ...,
  contour = TRUE,
  contour_var = "density",
  times = 10,
  seed = NULL,
  h = NULL,
  adjust = c(1, 1),
  n = 100,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

stat_density_2d_filled_sample(
  mapping = NULL,
  data = NULL,
  geom = "density_2d_filled",
  position = "identity",
  ...,
  contour = TRUE,
  contour_var = "density",
  times = 10,
  seed = NULL,
  h = NULL,
  adjust = c(1, 1),
  n = 100,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function

can be created from a formula (e.g. ~ head(.x, 10)).

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

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

... Arguments passed on to geom_contour

binwidth The width of the contour bins. Overridden by bins.

bins Number of contour bins. Overridden by breaks.

breaks One of:

- Numeric vector to set the contour breaks
- A function that takes the range of the data and binwidth as input and returns breaks as output. A function can be created from a formula (e.g. ~ fullseq(.x, .y)).

Overrides binwidth and bins. By default, this is a vector of length ten with pretty() breaks.

times A parameter used to control the number of values sampled from each distribution.

seed Set the seed for the layers random draw, allows you to plot the same draw across multiple layers.

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

arrow.fill fill colour to use for the arrow head (if closed). NULL means use colour aesthetic.

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

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

linemitre Line mitre limit (number greater than 1).

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

show.legend logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.

inherit.aes If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. annotation_borders().

rule Either "evenodd" or "winding". If polygons with holes are being drawn (using the subgroup aesthetic) this argument defines how the hole coordinates are interpreted. See the examples in grid::pathGrob() for an explanation.

geom, stat	Use to override the default connection between geom_density_2d() and stat_density_2d(). For more information at overriding these connections, see how the stat and geom arguments work.
contour	If TRUE, contour the results of the 2d density estimation.
contour_var	Character string identifying the variable to contour by. Can be one of "density", "ndensity", or "count". See the section on computed variables for details.
h	Bandwidth (vector of length two). If NULL, estimated using MASS::bandwidth.nrd().
adjust	A multiplicative bandwidth adjustment to be used if 'h' is 'NULL'. This makes it possible to adjust the bandwidth while still using the a bandwidth estimator. For example, adjust = 1/2 means use half of the default bandwidth.
n	Number of grid points in each direction.

Value

A ggplot2 layer

Examples

```
library(ggplot2)
# ggplot
m <- ggplot(faithful, aes(x = eruptions, y = waiting)) +
  geom_point() +
  xlim(0.5, 6) +
  ylim(40, 110)
# contour lines
m + geom_density_2d()

# ggdibbler
n <- ggplot(uncertain_faithful, aes(x = eruptions, y = waiting)) +
  geom_point_sample(size=2/10) +
  scale_x_continuous_distribution(limits = c(0.5, 6)) +
  scale_y_continuous_distribution(limits = c(40, 110))
n + geom_density_2d_sample(linewidth=2/10, alpha=0.5)

# contour bands
# ggplot
m + geom_density_2d_filled(alpha = 0.5)
# ggdibbler
n + geom_density_2d_filled_sample(alpha = 0.1)
```

---

geom_density_sample *Visualise densities with Uncertainty*

Description

Identical to geom_density, except that the fill for each density will be represented by a sample from each distribution.

Usage

```
geom_density_sample(
  mapping = NULL,
  data = NULL,
  stat = "density_sample",
  position = "identity",
  ...,
  outline.type = "upper",
  seed = NULL,
  times = 10,
  lineend = "butt",
  linejoin = "round",
  linemitre = 10,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

stat_density_sample(
  mapping = NULL,
  data = NULL,
  geom = "area",
  position = "stack_identity",
  ...,
  orientation = NA,
  seed = NULL,
  times = 10,
  bw = "nrd0",
  adjust = 1,
  kernel = "gaussian",
  n = 512,
  trim = FALSE,
  bounds = c(-Inf, Inf),
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be

created.

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

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

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

... — Other arguments passed on to layer()'s params argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the position argument, or aesthetics that are required can *not* be passed through .... Unknown arguments that are not part of the 4 categories below are ignored.

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

outline.type — Type of the outline of the area; "both" draws both the upper and lower lines, "upper"/"lower" draws the respective lines only. "full" draws a closed polygon around the area.

seed — Set the seed for the layers random draw, allows you to plot the same draw across multiple layers.

times — A parameter used to control the number of values sampled from each distribution.

lineend — Line end style (round, butt, square).

linejoin — Line join style (round, mitre, bevel).

linemitre	Line mitre limit (number greater than 1).
na.rm	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. annotation_borders().
geom, stat	Use to override the default connection between geom_density() and stat_density(). For more information about overriding these connections, see how the stat and geom arguments work.
orientation	The orientation of the layer. The default (NA) automatically determines the orientation from the aesthetic mapping. In the rare event that this fails it can be given explicitly by setting orientation to either "x" or "y". See the *Orientation* section for more detail.
bw	The smoothing bandwidth to be used. If numeric, the standard deviation of the smoothing kernel. If character, a rule to choose the bandwidth, as listed in stats::bw.nrd(). Note that automatic calculation of the bandwidth does not take weights into account.
adjust	A multiplicate bandwidth adjustment. This makes it possible to adjust the bandwidth while still using the a bandwidth estimator. For example, adjust = 1/2 means use half of the default bandwidth.
kernel	Kernel. See list of available kernels in density().
n	number of equally spaced points at which the density is to be estimated, should be a power of two, see density() for details
trim	If FALSE, the default, each density is computed on the full range of the data. If TRUE, each density is computed over the range of that group: this typically means the estimated x values will not line-up, and hence you won't be able to stack density values. This parameter only matters if you are displaying multiple densities in one plot or if you are manually adjusting the scale limits.
bounds	Known lower and upper bounds for estimated data. Default c(-Inf, Inf) means that there are no (finite) bounds. If any bound is finite, boundary effect of default density estimation will be corrected by reflecting tails outside bounds around their closest edge. Data points outside of bounds are removed with a warning.

Value

A ggplot2 layer

Examples

```
library(ggplot2)

# Basic density plot
# GGPLOT
ggplot(smaller_diamonds, aes(carat)) +
  geom_density()
# GGDIBBLER
ggplot(smaller_uncertain_diamonds, aes(carat)) +
  geom_density_sample(alpha=0.5)

# ggplot
ggplot(smaller_diamonds, aes(depth, fill = cut, colour = cut)) +
  geom_density(alpha = 0.7) +
  xlim(55, 70)
# ggdibbler
ggplot(smaller_uncertain_diamonds, aes(depth, fill = cut)) +
  geom_density_sample(aes(colour = after_stat(fill)), alpha = 0.1) +
 scale_x_continuous_distribution(limits=c(55, 70)) + #' ggdibbler does not have an xlim (yet)
  theme(palette.colour.discrete = "viridis",
        palette.fill.discrete = "viridis") #' bug: random variables have different colour
```

---

geom_dotplot_sample *Dot plot with uncertainty*

Description

Identical to geom_dotplot, except that it will accept a distribution in place of any of the usual aesthetics.

Usage

```
geom_dotplot_sample(
  mapping = NULL,
  data = NULL,
  position = "identity",
  seed = NULL,
  ...,
  times = 10,
  binwidth = NULL,
  binaxis = "x",
  method = "dotdensity",
  binpositions = "bygroup",
  stackdir = "up",
  stackratio = 1,
  dotsize = 1,
```

```
  stackgroups = FALSE,
  origin = NULL,
  right = TRUE,
  width = 0.9,
  drop = FALSE,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.
seed	Set the seed for the layers random draw, allows you to plot the same draw across multiple layers.
...	Other arguments passed on to layer()'s params argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the position argument, or aesthetics that are required can *not* be passed through .... Unknown arguments that are not part of the 4 categories below are ignored. • Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, colour = "red" or linewidth = 3. The geom's documentation has an **Aesthetics** section that lists the available options. The 'required' aesthetics cannot be passed on to the params. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data.

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

`times` A parameter used to control the number of values sampled from each distribution.

`binwidth` When `method` is "dotdensity", this specifies maximum bin width. When `method` is "histodot", this specifies bin width. Defaults to 1/30 of the range of the data

`binaxis` The axis to bin along, "x" (default) or "y"

`method` "dotdensity" (default) for dot-density binning, or "histodot" for fixed bin widths (like stat_bin)

`binpositions` When `method` is "dotdensity", "bygroup" (default) determines positions of the bins for each group separately. "all" determines positions of the bins with all the data taken together; this is used for aligning dot stacks across multiple groups.

`stackdir` which direction to stack the dots. "up" (default), "down", "center", "centerwhole" (centered, but with dots aligned)

`stackratio` how close to stack the dots. Default is 1, where dots just touch. Use smaller values for closer, overlapping dots.

`dotsize` The diameter of the dots relative to `binwidth`, default 1.

`stackgroups` should dots be stacked across groups? This has the effect that `position = "stack"` should have, but can't (because this geom has some odd properties).

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

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

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

`drop` If TRUE, remove all bins with zero counts

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

`show.legend` logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use `TRUE`. If `NA`, all levels are shown in legend, but unobserved levels are omitted.

`inherit.aes` If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. `annotation_borders()`.

Value

A ggplot2 layer

Examples

```
library(ggplot2)
# ggplot
ggplot(mtcars, aes(x = mpg)) +
  geom_dotplot()
# ggdibbler
ggplot(uncertain_mtcars, aes(x = mpg)) +
  geom_dotplot_sample(alpha=0.2)

# ggplot
ggplot(mtcars, aes(x = mpg)) +
  geom_dotplot(binwidth = 1.5)
# ggdibbler
ggplot(uncertain_mtcars, aes(x = mpg)) +
  geom_dotplot_sample(binwidth = 1.5, alpha=0.2)

# Use fixed-width bins
#ggplot
ggplot(mtcars, aes(x = mpg)) +
  geom_dotplot(method="histodot", binwidth = 1.5)
# ggdibbler
ggplot(uncertain_mtcars, aes(x = mpg)) +
  geom_dotplot_sample(method="histodot", binwidth = 1.5,
                      alpha=0.2)
```

---

geom_freqpoly_sample *Histograms and frequency polygons with uncertainty*

Description

Identical to geom_histogram, geom_freqpoly, and stat-bin except that it will accept a distribution in place of any of the usual aesthetics.

Usage

```
geom_freqpoly_sample(
  mapping = NULL,
  data = NULL,
  stat = "bin_sample",
  position = "identity",
  ...,
  na.rm = FALSE,
  times = 10,
  seed = NULL,
```

```
  show.legend = NA,
  inherit.aes = TRUE
)

geom_histogram_sample(
  mapping = NULL,
  data = NULL,
  stat = "bin_sample",
  position = "stack_dodge",
  ...,
  times = 10,
  seed = NULL,
  binwidth = NULL,
  bins = NULL,
  orientation = NA,
  lineend = "butt",
  linejoin = "mitre",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

stat_bin_sample(
  mapping = NULL,
  data = NULL,
  geom = "bar",
  position = "stack_dodge",
  ...,
  times = 10,
  orientation = NA,
  seed = NULL,
  binwidth = NULL,
  bins = NULL,
  center = NULL,
  boundary = NULL,
  closed = c("right", "left"),
  pad = FALSE,
  breaks = NULL,
  drop = "none",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.

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

If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`.

A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created.

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

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

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

`...` Other arguments passed on to `layer()`'s `params` argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the `position` argument, or aesthetics that are required can *not* be passed through `...`. Unknown arguments that are not part of the 4 categories below are ignored.

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

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

`times` A parameter used to control the number of values sampled from each distribution.

seed	Set the seed for the layers random draw, allows you to plot the same draw across multiple layers.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. annotation_borders().
binwidth	The width of the bins. Can be specified as a numeric value or as a function that takes x after scale transformation as input and returns a single numeric value. When specifying a function along with a grouping structure, the function will be called once per group. The default is to use the number of bins in bins, covering the range of the data. You should always override this value, exploring multiple widths to find the best to illustrate the stories in your data. The bin width of a date variable is the number of days in each time; the bin width of a time variable is the number of seconds.
bins	Number of bins. Overridden by binwidth. Defaults to 30.
orientation	The orientation of the layer. The default (NA) automatically determines the orientation from the aesthetic mapping. In the rare event that this fails it can be given explicitly by setting orientation to either "x" or "y". See the *Orientation* section for more detail.
lineend	Line end style (round, butt, square).
linejoin	Line join style (round, mitre, bevel).
geom, stat	Use to override the default connection between geom_histogram()/geom_freqpoly() and stat_bin(). For more information at overriding these connections, see how the stat and geom arguments work.
center, boundary	bin position specifiers. Only one, center or boundary, may be specified for a single plot. center specifies the center of one of the bins. boundary specifies the boundary between two bins. Note that if either is above or below the range of the data, things will be shifted by the appropriate integer multiple of binwidth. For example, to center on integers use binwidth = 1 and center = 0, even if 0 is outside the range of the data. Alternatively, this same alignment can be specified with binwidth = 1 and boundary = 0.5, even if 0.5 is outside the range of the data.
closed	One of "right" or "left" indicating whether right or left edges of bins are included in the bin.
pad	If TRUE, adds empty bins at either end of x. This ensures frequency polygons touch 0. Defaults to FALSE.
breaks	Alternatively, you can supply a numeric vector giving the bin boundaries. Overrides binwidth, bins, center, and boundary. Can also be a function that takes group-wise values as input and returns bin boundaries.

drop — Treatment of zero count bins. If "none" (default), such bins are kept as-is. If "all", all zero count bins are filtered out. If "extremes" only zero count bins at the flanks are filtered out, but not in the middle. TRUE is shorthand for "all" and FALSE is shorthand for "none".

Value

A ggplot2 layer

Examples

```
# load ggplot
library(ggplot2)
# ggplot
ggplot(smaller_diamonds, aes(carat)) +
  geom_histogram()
# ggdibbler
ggplot(smaller_uncertain_diamonds, aes(carat)) +
  geom_histogram_sample() #' alpha
ggplot(smaller_uncertain_diamonds, aes(carat)) +
  geom_histogram_sample(position="identity_identity", alpha=0.15)

# ggplot
ggplot(smaller_diamonds, aes(price, colour = cut)) +
  geom_freqpoly(binwidth = 500)
# ggdibbler
ggplot(smaller_uncertain_diamonds, aes(price, colour = cut)) +
  geom_freqpoly_sample(binwidth = 500)

# ggplot2
ggplot(smaller_diamonds, aes(price, fill = cut)) +
  geom_histogram(binwidth = 500)
# ggdibbler
ggplot(smaller_uncertain_diamonds, aes(price, fill = cut)) +
 geom_histogram_sample(binwidth = 500)
```

---

geom_hex_sample *Uncertain hexagonal heatmap of 2d bin counts*

Description

Identical to geom_hex, except that it will accept a distribution in place of any of the usual aesthetics.

Usage

```
geom_hex_sample(
  mapping = NULL,
```

```
  data = NULL,
  stat = "bin_hex_sample",
  position = "identity",
  ...,
  times = 10,
  seed = NULL,
  lineend = "butt",
  linejoin = "mitre",
  linemitre = 10,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

stat_bin_hex_sample(
  mapping = NULL,
  data = NULL,
  geom = "hex",
  position = "identity",
  ...,
  times = 10,
  seed = NULL,
  binwidth = NULL,
  bins = 30,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position.

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

`...` Other arguments passed on to `layer()`'s `params` argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the `position` argument, or aesthetics that are required can *not* be passed through `....` Unknown arguments that are not part of the 4 categories below are ignored.

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

`times` A parameter used to control the number of values sampled from each distribution.

`seed` Set the seed for the layers random draw, allows you to plot the same draw across multiple layers.

`lineend` Line end style (round, butt, square).

`linejoin` Line join style (round, mitre, bevel).

`linemitre` Line mitre limit (number greater than 1).

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

`show.legend` logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use `TRUE`. If `NA`, all levels are shown in legend, but unobserved levels are omitted.

`inherit.aes` If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. `annotation_borders()`.

geom, stat	Override the default connection between geom_hex() and stat_bin_hex(). For more information about overriding these connections, see how the stat and geom arguments work.
binwidth	The width of the bins. Can be specified as a numeric value or as a function that takes x after scale transformation as input and returns a single numeric value. When specifying a function along with a grouping structure, the function will be called once per group. The default is to use the number of bins in bins, covering the range of the data. You should always override this value, exploring multiple widths to find the best to illustrate the stories in your data. The bin width of a date variable is the number of days in each time; the bin width of a time variable is the number of seconds.
bins	Number of bins. Overridden by binwidth. Defaults to 30.

Value

A ggplot2 layer

Examples

```
library(ggplot2)
d <- ggplot(smaller_diamonds, aes(carat, price))
d + geom_hex()

b <- ggplot(smaller_uncertain_diamonds, aes(carat, price))
b + geom_hex_sample(alpha=0.15)

# You still have access to all the same parameters
d + geom_hex(bins = 10)
b + geom_hex_sample(bins = 10, alpha=0.15)
```

---

geom_jitter_sample *Uncertain Jittered Points*

Description

Identical to geom_jitter, except that it will accept a distribution in place of any of the usual aesthetics.

Usage

```
geom_jitter_sample(
  mapping = NULL,
  data = NULL,
  stat = "identity_sample",
  position = "jitter",
  ...,
  width = NULL,
```

```
  height = NULL,
  na.rm = FALSE,
  times = 10,
  seed = NULL,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
stat	The statistical transformation to use on the data for this layer. When using a geom_*() function to construct a layer, the stat argument can be used to override the default coupling between geoms and stats. The stat argument accepts the following: • A Stat ggproto subclass, for example StatCount. • A string naming the stat. To give the stat as a string, strip the function name of the stat_ prefix. For example, to use stat_count(), give the stat as "count". • For more information and other ways to specify the stat, see the layer stat documentation.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.
...	Other arguments passed on to layer()'s params argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the position argument, or aesthetics that are required can *not* be passed through .... Unknown arguments that are not part of the 4 categories below are ignored.

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

`width, height` Amount of vertical and horizontal jitter. The jitter is added in both positive and negative directions, so the total spread is twice the value specified here.

If omitted, defaults to 40% of the resolution of the data: this means the jitter values will occupy 80% of the implied bins. Categorical data is aligned on the integers, so a width or height of 0.5 will spread the data so it's not possible to see the distinction between the categories.

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

`times` A parameter used to control the number of values sampled from each distribution.

`seed` Set the seed for the layers random draw, allows you to plot the same draw across multiple layers.

`show.legend` logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use `TRUE`. If `NA`, all levels are shown in legend, but unobserved levels are omitted.

`inherit.aes` If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. `annotation_borders()`.

Value

A ggplot2 layer

Examples

```
library(ggplot2)
```

```
# ggplot
p <- ggplot(mpg, aes(cyl, hwy)) #ggplot
q <- ggplot(uncertain_mpg, aes(cyl, hwy)) #ggdibbler
p + geom_point()
q + geom_point_sample(times=10)

# ggplot
p + geom_jitter()
# ggdibbler
q + geom_jitter_sample(times=10)

# Add aesthetic mappings
p + geom_jitter(aes(colour = class)) #ggplot
p + geom_jitter_sample(aes(colour = class)) #ggdibler
```

`geom_label_sample` *Uncertain Text*

Description

Identical to geom_text and geom_label except that it will accept a distribution in place of any of the usual aesthetics.

Usage

```
geom_label_sample(
  mapping = NULL,
  data = NULL,
  times = 10,
  seed = NULL,
  stat = "identity_sample",
  position = "nudge",
  ...,
  parse = FALSE,
  label.padding = unit(0.25, "lines"),
  label.r = unit(0.15, "lines"),
  label.size = deprecated(),
  border.colour = NULL,
  border.color = NULL,
  text.colour = NULL,
  text.color = NULL,
  size.unit = "mm",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

```
geom_text_sample(
  mapping = NULL,
  data = NULL,
  stat = "identity_sample",
  position = "nudge",
  ...,
  times = 10,
  seed = NULL,
  parse = FALSE,
  check_overlap = FALSE,
  size.unit = "mm",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
times	A parameter used to control the number of values sampled from each distribution.
seed	Set the seed for the layers random draw, allows you to plot the same draw across multiple layers.
stat	The statistical transformation to use on the data for this layer. When using a geom_*() function to construct a layer, the stat argument can be used to override the default coupling between geoms and stats. The stat argument accepts the following: • A Stat ggproto subclass, for example StatCount. • A string naming the stat. To give the stat as a string, strip the function name of the stat_ prefix. For example, to use stat_count(), give the stat as "count". • For more information and other ways to specify the stat, see the layer stat documentation.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following:

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

`...` Other arguments passed on to `layer()`'s `params` argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the `position` argument, or aesthetics that are required can *not* be passed through `...`. Unknown arguments that are not part of the 4 categories below are ignored.

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

`parse` If `TRUE`, the labels will be parsed into expressions and displayed as described in `?plotmath`.

`label.padding` Amount of padding around label. Defaults to 0.25 lines.

`label.r` Radius of rounded corners. Defaults to 0.15 lines.

`label.size` **[Deprecated]** Replaced by the `linewidth` aesthetic. Size of label border, in mm.

`border.colour`, `border.color`
Colour of label border. When `NULL` (default), the `colour` aesthetic determines the colour of the label border. `border.color` is an alias for `border.colour`.

`text.colour`, `text.color`
Colour of the text. When `NULL` (default), the `colour` aesthetic determines the colour of the text. `text.color` is an alias for `text.colour`.

`size.unit` How the `size` aesthetic is interpreted: as millimetres (`"mm"`, default), points (`"pt"`), centimetres (`"cm"`), inches (`"in"`), or picas (`"pc"`).

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

show.legend — logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.

inherit.aes — If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. annotation_borders().

check_overlap — If TRUE, text that overlaps previous text in the same layer will not be plotted. check_overlap happens at draw time and in the order of the data. Therefore data should be arranged by the label column before calling geom_text(). Note that this argument is not supported by geom_label().

Value

A ggplot2 geom representing a point_sample which can be added to a ggplot object

A ggplot2 layer

Examples

```
library(ggplot2)

p <- ggplot(mtcars, aes(wt, mpg, label = rownames(mtcars)))
q <- ggplot(uncertain_mtcars, aes(wt, mpg, label = rownames(uncertain_mtcars)))

# Text example
p + geom_text() # ggplot
q + geom_text_sample(times=3, alpha=0.5) #ggdibbler
# Labels with background
p + geom_label() #ggplot
q + geom_label_sample(times=3, alpha=0.5) #ggdibbler

# Random text with constant position (harder to read signal supression)
# ggplot
ggplot(mtcars, aes(wt, mpg, label = cyl)) +
 geom_text(size=6)
# ggdibbler
ggplot(uncertain_mtcars, aes(mean(wt), mean(mpg), lab = cyl)) +
 geom_text_sample(aes(label = after_stat(lab)), size=6, alpha=0.3)
```

---

geom_path_sample *Uncertain Connected observations*

Description

Identical to geom_path, geom_line, and geom_step, except that it will accept a distribution in place of any of the usual aesthetics.

Usage

```
geom_path_sample(
  mapping = NULL,
  data = NULL,
  stat = "identity_sample",
  position = "identity",
  ...,
  times = 10,
  seed = NULL,
  arrow = NULL,
  arrow.fill = NULL,
  lineend = "butt",
  linejoin = "round",
  linemitre = 10,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

geom_line_sample(
  mapping = NULL,
  data = NULL,
  stat = "identity_sample",
  position = "identity",
  ...,
  times = 10,
  seed = NULL,
  orientation = NA,
  arrow = NULL,
  arrow.fill = NULL,
  lineend = "butt",
  linejoin = "round",
  linemitre = 10,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

geom_step_sample(
  mapping = NULL,
  data = NULL,
  stat = "identity_sample",
  position = "identity",
  ...,
  times = 10,
  seed = NULL,
  orientation = NA,
  lineend = "butt",
```

```
  linejoin = "round",
  linemitre = 10,
  arrow = NULL,
  arrow.fill = NULL,
  direction = "hv",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
stat	The statistical transformation to use on the data for this layer. When using a geom_*() function to construct a layer, the stat argument can be used to override the default coupling between geoms and stats. The stat argument accepts the following: • A Stat ggproto subclass, for example StatCount. • A string naming the stat. To give the stat as a string, strip the function name of the stat_ prefix. For example, to use stat_count(), give the stat as "count". • For more information and other ways to specify the stat, see the layer stat documentation.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.
...	Other arguments passed on to layer()'s params argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the

position argument, or aesthetics that are required can *not* be passed through .... Unknown arguments that are not part of the 4 categories below are ignored.

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

times	A parameter used to control the number of values sampled from each distribution.
seed	Set the seed for the layers random draw, allows you to plot the same draw across multiple layers.
arrow	Arrow specification, as created by grid::arrow().
arrow.fill	fill colour to use for the arrow head (if closed). NULL means use colour aesthetic.
lineend	Line end style (round, butt, square).
linejoin	Line join style (round, mitre, bevel).
linemitre	Line mitre limit (number greater than 1).
na.rm	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. annotation_borders().
orientation	The orientation of the layer. The default (NA) automatically determines the orientation from the aesthetic mapping. In the rare event that this fails it can be given explicitly by setting orientation to either "x" or "y". See the *Orientation* section for more detail.
direction	direction of stairs: 'vh' for vertical then horizontal, 'hv' for horizontal then vertical, or 'mid' for step half-way between adjacent x-values.

Value

A ggplot2 layer

Examples

```
library(ggplot2)
library(dplyr)
library(distributional)

#ggplot
ggplot(economics, aes(date, unemploy)) + geom_line()
#ggdibbler
ggplot(uncertain_economics, aes(date, unemploy)) +
  geom_line_sample(alpha=0.1)

# geom_step() is useful when you want to highlight exactly when
# the y value changes
recent <- economics[economics$date > as.Date("2013-01-01"), ]
uncertain_recent <- uncertain_economics[uncertain_economics$date > as.Date("2013-01-01"), ]
# geom line
ggplot(recent, aes(date, unemploy)) + geom_step()#ggplot
ggplot(uncertain_recent, aes(date, unemploy)) + geom_step_sample(alpha=0.5)#ggdibbler

# geom_path lets you explore how two variables are related over time,
# ggplot
m <- ggplot(economics, aes(unemploy, psavert))
m + geom_path(aes(colour = as.numeric(date)))
# ggdibbler
n <- ggplot(uncertain_economics, aes(unemploy, psavert))
n  + geom_path_sample(aes(colour = as.numeric(date)), alpha=0.15)

# You can use NAs to break the line.
df <- data.frame(x = 1:5, y = c(1, 2, NA, 4, 5))
uncertain_df <- df |> mutate(y=dist_normal(y, 0.3))
# ggplot
ggplot(df, aes(x, y)) + geom_point() + geom_line()
# ggdibbler
ggplot(uncertain_df, aes(x, y)) +
  geom_point_sample(seed=33) +
  geom_line_sample(seed=33)
```

---

geom_point_sample *Visualise Uncertain Points*

Description

Identical to geom_point, except that it will accept a distribution in place of any of the usual aesthetics.

Usage

```
geom_point_sample(
  mapping = NULL,
  data = NULL,
  stat = "identity_sample",
  position = "identity",
  ...,
  times = 10,
  seed = NULL,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
stat	The statistical transformation to use on the data for this layer. When using a geom_*() function to construct a layer, the stat argument can be used to override the default coupling between geoms and stats. The stat argument accepts the following: • A Stat ggproto subclass, for example StatCount. • A string naming the stat. To give the stat as a string, strip the function name of the stat_ prefix. For example, to use stat_count(), give the stat as "count". • For more information and other ways to specify the stat, see the layer stat documentation.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter".

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

`...` Other arguments passed on to `layer()`'s `params` argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the `position` argument, or aesthetics that are required can *not* be passed through `....` Unknown arguments that are not part of the 4 categories below are ignored.

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

`times` A parameter used to control the number of values sampled from each distribution.

`seed` Set the seed for the layers random draw, allows you to plot the same draw across multiple layers.

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

`show.legend` logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use `TRUE`. If `NA`, all levels are shown in legend, but unobserved levels are omitted.

`inherit.aes` If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. `annotation_borders()`.

Value

A ggplot2 layer

Examples

```
library(ggplot2)
library(distributional)
```

```
  # ggplot
p <- ggplot(mtcars, aes(wt, mpg))
p + geom_point()

  # ggdibbler - set the sample size with times
q <- ggplot(uncertain_mtcars, aes(wt, mpg))
q + geom_point_sample(times=50, alpha=0.5)

# Add aesthetic mappings

 # ggplot
p + geom_point(aes(colour = factor(cyl)))
  # ggdibbler - a
q + geom_point_sample(aes(colour = dist_transformed(cyl, factor, as.numeric))) +
labs(colour = "factor(cyl)")
 # ggplot
p + geom_point(aes(shape = factor(cyl)))
  # ggdibbler
q + geom_point_sample(aes(shape = dist_transformed(cyl, factor, as.numeric))) +
labs(shape = "factor(cyl)")

# A "bubblechart":
# ggplot2
p + geom_point(aes(size = qsec))
# ggdibbler
q + geom_point_sample(aes(size = qsec), alpha=0.15)
```

`geom_polygon_sample` *Uncertain Polygons*

Description

Identical to geom_polygon, except that it will accept a distribution in place of any of the usual aesthetics.

Usage

```
geom_polygon_sample(
  mapping = NULL,
  data = NULL,
  stat = "identity_sample",
  position = "identity",
  ...,
  times = 10,
  seed = NULL,
  rule = "evenodd",
  lineend = "butt",
  linejoin = "round",
```

```
  linemitre = 10,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
stat	The statistical transformation to use on the data for this layer. When using a geom_*() function to construct a layer, the stat argument can be used to override the default coupling between geoms and stats. The stat argument accepts the following: • A Stat ggproto subclass, for example StatCount. • A string naming the stat. To give the stat as a string, strip the function name of the stat_ prefix. For example, to use stat_count(), give the stat as "count". • For more information and other ways to specify the stat, see the layer stat documentation.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.
...	Other arguments passed on to layer()'s params argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the position argument, or aesthetics that are required can *not* be passed through .... Unknown arguments that are not part of the 4 categories below are ignored. • Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, colour = "red" or linewidth

= 3. The geom's documentation has an **Aesthetics** section that lists the available options. The 'required' aesthetics cannot be passed on to the params. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data.

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

times — A parameter used to control the number of values sampled from each distribution.

seed — Set the seed for the layers random draw, allows you to plot the same draw across multiple layers.

rule — Either "evenodd" or "winding". If polygons with holes are being drawn (using the subgroup aesthetic) this argument defines how the hole coordinates are interpreted. See the examples in grid::pathGrob() for an explanation.

lineend — Line end style (round, butt, square).

linejoin — Line join style (round, mitre, bevel).

linemitre — Line mitre limit (number greater than 1).

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

show.legend — logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.

inherit.aes — If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. annotation_borders().

Value

A ggplot2 layer

Examples

```
library(ggplot2)
library(distributional)
library(dplyr)
```

```
ids <- factor(c("1.1", "2.1", "1.2", "2.2", "1.3", "2.3"))

values <- data.frame(
  id = ids,
  value = c(3, 3.1, 3.1, 3.2, 3.15, 3.5)
)
positions <- data.frame(
  id = rep(ids, each = 4),
  x = c(2, 1, 1.1, 2.2, 1, 0, 0.3, 1.1, 2.2, 1.1, 1.2, 2.5, 1.1, 0.3,
        0.5, 1.2, 2.5, 1.2, 1.3, 2.7, 1.2, 0.5, 0.6, 1.3),
  y = c(-0.5, 0, 1, 0.5, 0, 0.5, 1.5, 1, 0.5, 1, 2.1, 1.7, 1, 1.5,
        2.2, 2.1, 1.7, 2.1, 3.2, 2.8, 2.1, 2.2, 3.3, 3.2)
)
#' Currently we need to manually merge the two together
datapoly <- merge(values, positions, by = c("id"))

#' Make uncertain version of datapoly
uncertain_datapoly <- datapoly |>
  mutate(x = dist_uniform(x-0.1, x + 0.1),
         y = dist_uniform(y-0.1, y + 0.1),
         value = dist_uniform(value-0.5, value + 0.5))


p <- ggplot(datapoly, aes(x = x, y = y)) +
  geom_polygon(aes(fill = value, group = id))
p

q <- ggplot(uncertain_datapoly, aes(x = x, y = y)) +
  geom_polygon_sample(aes(fill = value, group = id), alpha=0.15)
q
```

`geom_quantile_sample` *Quantile regression with uncertainty*

Description

Identical to geom_quantile, except that it will accept a distribution in place of any of the usual aesthetics.

Usage

```
geom_quantile_sample(
  mapping = NULL,
  data = NULL,
  stat = "quantile_sample",
  position = "identity",
  ...,
  times = 10,
```

```
  seed = NULL,
  arrow = NULL,
  arrow.fill = NULL,
  lineend = "butt",
  linejoin = "round",
  linemitre = 10,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

stat_quantile_sample(
  mapping = NULL,
  data = NULL,
  geom = "quantile",
  position = "identity",
  ...,
  seed = NULL,
  times = 10,
  quantiles = c(0.25, 0.5, 0.75),
  formula = NULL,
  method = "rq",
  method.args = list(),
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position.

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

`...` Other arguments passed on to `layer()`'s `params` argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the `position` argument, or aesthetics that are required can *not* be passed through `...`. Unknown arguments that are not part of the 4 categories below are ignored.

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

`times` A parameter used to control the number of values sampled from each distribution.

`seed` Set the seed for the layers random draw, allows you to plot the same draw across multiple layers.

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

`arrow.fill` fill colour to use for the arrow head (if closed). `NULL` means use `colour` aesthetic.

`lineend` Line end style (round, butt, square).

`linejoin` Line join style (round, mitre, bevel).

`linemitre` Line mitre limit (number greater than 1).

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

`show.legend` logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use `TRUE`. If `NA`, all levels are shown in legend, but unobserved levels are omitted.

inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. annotation_borders().
geom, stat	Use to override the default connection between geom_quantile() and stat_quantile(). For more information about overriding these connections, see how the stat and geom arguments work.
quantiles	conditional quantiles of y to calculate and display
formula	formula relating y variables to x variables
method	Quantile regression method to use. Available options are "rq" (for quantreg::rq()) and "rqss" (for quantreg::rqss()).
method.args	List of additional arguments passed on to the modelling function defined by method.

Value

A ggplot2 layer

Examples

```
library(ggplot2)
# ggplot
m <- ggplot(mpg, aes(displ, hwy)) +
  geom_point()
# ggdibbler
n <- ggplot(uncertain_mpg, aes(displ, hwy)) +
  geom_point_sample(alpha=0.3)
# ggplot
m + geom_quantile()
# ggdibbler
n + geom_quantile_sample(alpha=0.3)

# ggplot
m + geom_quantile(quantiles = 0.5)
# ggdibbler
n + geom_quantile_sample(quantiles = 0.5, alpha=0.3)
```

---

geom_raster_sample *Plot rectangles with uncertainty*

Description

Identical to geom_tile and geom_rect, except that they will accept a distribution in place of any of the usual aesthetics.

Usage

```
geom_raster_sample(
  mapping = NULL,
  data = NULL,
  stat = "identity_sample",
  position = "identity_dodge",
  ...,
  times = 10,
  seed = NULL,
  interpolate = FALSE,
  hjust = 0.5,
  vjust = 0.5,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

geom_rect_sample(
  mapping = NULL,
  data = NULL,
  stat = "identity_sample",
  position = "identity",
  ...,
  times = 10,
  seed = NULL,
  lineend = "butt",
  linejoin = "mitre",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

geom_tile_sample(
  mapping = NULL,
  data = NULL,
  stat = "identity_sample",
  position = "identity_dodge",
  ...,
  times = 10,
  seed = NULL,
  lineend = "butt",
  linejoin = "mitre",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
stat	The statistical transformation to use on the data for this layer. When using a geom_*() function to construct a layer, the stat argument can be used to override the default coupling between geoms and stats. The stat argument accepts the following: • A Stat ggproto subclass, for example StatCount. • A string naming the stat. To give the stat as a string, strip the function name of the stat_ prefix. For example, to use stat_count(), give the stat as "count". • For more information and other ways to specify the stat, see the layer stat documentation.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.
...	Other arguments passed on to layer()'s params argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the position argument, or aesthetics that are required can *not* be passed through .... Unknown arguments that are not part of the 4 categories below are ignored. • Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, colour = "red" or linewidth = 3. The geom's documentation has an **Aesthetics** section that lists the available options. The 'required' aesthetics cannot be passed on to the params. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data.

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

`times` A parameter used to control the number of values sampled from each distribution.

`seed` Set the seed for the layers random draw, allows you to plot the same draw across multiple layers.

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

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

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

`show.legend` logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use `TRUE`. If `NA`, all levels are shown in legend, but unobserved levels are omitted.

`inherit.aes` If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. `annotation_borders()`.

`lineend` Line end style (round, butt, square).

`linejoin` Line join style (round, mitre, bevel).

Value

A ggplot2 layer

Examples

```
library(ggplot2)
library(distributional)
library(dplyr)

# Rasters
#ggplot
ggplot(faithfuld, aes(waiting, eruptions)) +
  geom_raster(aes(fill = density))
#ggdibbler
```

```
ggplot(uncertain_faithfuld, aes(waiting, eruptions)) +
  geom_raster_sample(aes(fill = density))

# Justification controls where the cells are anchored
df <- expand.grid(x = 0:5, y = 0:5)
set.seed(1)
df$z <- runif(nrow(df))
uncertain_df <- df |>
  group_by(x,y) |>
  mutate(z = dist_normal(z, runif(1, 0, 0.1))) |>
  ungroup()

# default is compatible with geom_tile()
# ggplot
ggplot(df, aes(x, y, fill = z)) +
  geom_raster()
#ggdibbler
ggplot(uncertain_df, aes(x, y, fill = z)) +
  geom_raster_sample()

# If you want to draw arbitrary rectangles,
# use geom_tile_sample() or geom_rect_sample()
tile_df <- data.frame(
  x = rep(c(2, 5, 7, 9, 12), 2),
  y = rep(c(1, 2), each = 5),
  z = factor(rep(1:5, each = 2)),
  w = rep(diff(c(0, 4, 6, 8, 10, 14)), 2)
)
# most likely case that only colour is random
uncertain_tile_df <- tile_df
uncertain_tile_df$z <- dist_transformed((1 + dist_binomial(rep(1:5,
  each = 2), 0.5)), factor, as.numeric)

# ggplot
ggplot(tile_df, aes(x, y)) +
  geom_tile(aes(fill = z), colour = "grey50")
# ggdibbler
ggplot(uncertain_tile_df, aes(x, y)) +
  geom_tile_sample(aes(fill = z), position="identity_dodge") +
  geom_tile(fill = NA, colour = "grey50", linewidth=1) +
  labs(fill = "z")

# Rectangles
rect_df  <- tile_df |>
  mutate(xmin = x - w / 2,
         xmax = x + w / 2,
         ymin = y,
         ymax = y + 1)

uncertain_rect  <- rect_df|>
  mutate(xmin = dist_normal(xmin, 0.5),
         xmax = dist_normal(xmax, 0.5),
         ymin = dist_normal(ymin, 0.5),
```

```
        ymax = dist_normal(ymax, 0.5))

# ggplot
ggplot(data = rect_df,
       aes(xmin= xmin, xmax = xmax, ymin = ymin, ymax = ymax)) +
  geom_rect(aes(fill = z), colour = "grey50")
# ggdibbler
ggplot(data = uncertain_rect,
       aes(xmin= xmin, xmax = xmax, ymin = ymin, ymax = ymax, f = z)) +
  geom_rect_sample(aes(fill = as.factor(after_stat(f))),
    colour = "grey50", alpha=0.2) +
  labs(fill = "z")
```

`geom_ribbon_sample` *Ribbons and area plots with uncertainty*

Description

Identical to geom_ribbon and geom_area, except that it will accept a distribution in place of any of the usual aesthetics.

Usage

```
geom_ribbon_sample(
  mapping = NULL,
  data = NULL,
  stat = "identity_sample",
  position = "identity",
  ...,
  seed = NULL,
  times = 10,
  lineend = "butt",
  linejoin = "round",
  linemitre = 10,
  outline.type = "both",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

geom_area_sample(
  mapping = NULL,
  data = NULL,
  stat = "align_sample",
  position = "stack_identity",
  ...,
  times = 10,
  seed = NULL,
```

```
  orientation = NA,
  outline.type = "upper",
  lineend = "butt",
  linejoin = "round",
  linemitre = 10,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

stat_align_sample(
  mapping = NULL,
  data = NULL,
  geom = "area",
  position = "identity",
  ...,
  times = 10,
  seed = NULL,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
stat	The statistical transformation to use on the data for this layer. When using a geom_*() function to construct a layer, the stat argument can be used to override the default coupling between geoms and stats. The stat argument accepts the following: • A Stat ggproto subclass, for example StatCount. • A string naming the stat. To give the stat as a string, strip the function name of the stat_ prefix. For example, to use stat_count(), give the stat as "count". • For more information and other ways to specify the stat, see the layer stat documentation.

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

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

... — Other arguments passed on to layer()'s params argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the position argument, or aesthetics that are required can *not* be passed through .... Unknown arguments that are not part of the 4 categories below are ignored.

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

seed — Set the seed for the layers random draw, allows you to plot the same draw across multiple layers.

times — A parameter used to control the number of values sampled from each distribution.

lineend — Line end style (round, butt, square).

linejoin — Line join style (round, mitre, bevel).

linemitre — Line mitre limit (number greater than 1).

outline.type — Type of the outline of the area; "both" draws both the upper and lower lines, "upper"/"lower" draws the respective lines only. "full" draws a closed polygon around the area.

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

show.legend — logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.

inherit.aes — If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. annotation_borders().

orientation — The orientation of the layer. The default (NA) automatically determines the orientation from the aesthetic mapping. In the rare event that this fails it can be given explicitly by setting orientation to either "x" or "y". See the *Orientation* section for more detail.

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

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

Value

A ggplot2 layer

Examples

```
library(distributional)
library(dplyr)
library(ggplot2)

# Generate data
huron <- data.frame(year = 1875:1972, level = as.vector(LakeHuron))
uncertain_huron <- huron |>
  group_by(year) |>
  mutate(level = dist_normal(level, runif(1,0,2)))

# ggplot
h <- ggplot(huron, aes(year))
# ggdibbler
q <- ggplot(uncertain_huron, aes(year))

# ggplot
h + geom_ribbon(aes(ymin=0, ymax=level))
# ggdibbler
q + geom_ribbon_sample(aes(ymin=0, ymax=level), alpha=0.15)
```

```
# Add aesthetic mappings
h + # ggplot
  geom_ribbon(aes(ymin = level - 1, ymax = level + 1), fill = "grey70") +
  geom_line(aes(y = level))
q + # ggdibbler
  geom_ribbon_sample(aes(ymin = level - 1, ymax = level + 1),
    fill = "grey70", seed=4, alpha=0.15) +
  geom_line_sample(aes(y = level), seed=4, alpha=0.15)

df <- data.frame(
  g = c("a", "a", "a", "b", "b", "b"),
  x = c(1, 3, 5, 2, 4, 6),
  y = c(2, 5, 1, 3, 6, 7)
)

uncertain_df <- df |>
  mutate(x = dist_normal(x, 0.8),
         y = dist_normal(y, 0.8))

# ggplot
ggplot(df, aes(x, y, fill = g)) +
  geom_area() +
  facet_grid(g ~ .)
# ggdibbler
ggplot(uncertain_df, aes(x, y, fill = g)) +
  geom_area_sample(seed=100, alpha=0.15) +
  geom_point_sample(seed=100) +
  facet_grid(g ~ .)
```

`geom_rug_sample` *Uncertain Rug plots in the margins*

Description

Identical to geom_rug, except that it will accept a distribution in place of any of the usual aesthetics.

Usage

```
geom_rug_sample(
  mapping = NULL,
  data = NULL,
  stat = "identity_sample",
  position = "identity",
  ...,
  times = 10,
  seed = NULL,
  lineend = "butt",
  sides = "bl",
  outside = FALSE,
```

```
  length = unit(0.03, "npc"),
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
stat	The statistical transformation to use on the data for this layer. When using a geom_*() function to construct a layer, the stat argument can be used to override the default coupling between geoms and stats. The stat argument accepts the following: • A Stat ggproto subclass, for example StatCount. • A string naming the stat. To give the stat as a string, strip the function name of the stat_ prefix. For example, to use stat_count(), give the stat as "count". • For more information and other ways to specify the stat, see the layer stat documentation.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.
...	Other arguments passed on to layer()'s params argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the position argument, or aesthetics that are required can *not* be passed through .... Unknown arguments that are not part of the 4 categories below are ignored. • Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, colour = "red" or linewidth

= 3. The geom's documentation has an **Aesthetics** section that lists the available options. The 'required' aesthetics cannot be passed on to the params. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data.

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

times — A parameter used to control the number of values sampled from each distribution.

seed — Set the seed for the layers random draw, allows you to plot the same draw across multiple layers.

lineend — Line end style (round, butt, square).

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

outside — logical that controls whether to move the rug tassels outside of the plot area. Default is off (FALSE). You will also need to use coord_cartesian(clip = "off"). When set to TRUE, also consider changing the sides argument to "tr". See examples.

length — A grid::unit() object that sets the length of the rug lines. Use scale expansion to avoid overplotting of data.

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

show.legend — logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.

inherit.aes — If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. annotation_borders().

Value

A ggplot2 layer

Examples

```
library(ggplot2)

# ggplot
p <- ggplot(mtcars, aes(wt, mpg)) +
  geom_point()
# ggdibbler
q <- ggplot(uncertain_mtcars, aes(wt, mpg)) +
  geom_point_sample(seed=4)

p + geom_rug() #ggplot
q + geom_rug_sample(seed=4, alpha=0.5) #ggdibbler
```

`geom_sf_sample` *Visualise Sf Objects with Uncertainty*

Description

Identical to geom_sf, except that the fill for each area will be a distribution. This function will replace the fill area with a grid, where each cell is filled with an outcome from the fill distribution.

Usage

```
geom_sf_sample(
  mapping = aes(),
  data = NULL,
  position = "subdivide",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  times = 10,
  seed = NULL,
  n = deprecated(),
  ...
)
```

Arguments

`mapping`	Set of aesthetic mappings created by `aes()`. If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.
`data`	The data to be displayed in this layer. There are three options: If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`. A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created.

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

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

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

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

show.legend — logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes.

You can also set this to one of "polygon", "line", and "point" to override the default legend.

inherit.aes — If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. annotation_borders().

times — A parameter used to control the number of values sampled from each distribution.

seed — Set the seed for the layers random draw, allows you to plot the same draw across multiple layers.

n — Deprecated in favour of times.

... — Other arguments passed on to layer()'s params argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the position argument, or aesthetics that are required can *not* be passed through .... Unknown arguments that are not part of the 4 categories below are ignored.

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

- The key_glyph argument of layer() may also be passed on through .... This can be one of the functions described as key glyphs, to change the display of the layer in the legend.

Value

A ggplot2 geom representing a sf_sample which can be added to a ggplot object

Examples

```
# In it's most basic form, the geom will make a subdivision
library(ggplot2)
library(dplyr)
library(sf)
basic_data <- toy_temp_dist |>
  filter(county_name %in% c("Pottawattamie County", "Mills County", "Cass County"))
basic_data |>
  ggplot() +
  geom_sf_sample(times=100, linewidth=0,
                 aes(geometry = county_geometry, fill=temp_dist))
# The original borders of the sf object can be hard to see,
 # so layering the original geometry on top can help to see the original boundaries
basic_data |>
  ggplot() +
 geom_sf_sample(aes(geometry = county_geometry, fill=temp_dist), linewidth=0, times=100) +
  geom_sf(aes(geometry=county_geometry), fill=NA, linewidth=1)
```

---

geom_smooth_sample *Uncertain Smooth*

Description

Identical to geom_smooth, except that it will accept a distribution in place of any of the usual aesthetics.

Usage

```
geom_smooth_sample(
  mapping = NULL,
  data = NULL,
  times = 10,
  seed = NULL,
  stat = "smooth_sample",
  position = "identity",
  ...,
  method = NULL,
  formula = NULL,
  se = TRUE,
  na.rm = FALSE,
```

```
  orientation = NA,
  show.legend = NA,
  inherit.aes = TRUE
)

stat_smooth_sample(
  mapping = NULL,
  data = NULL,
  geom = "smooth",
  position = "identity",
  ...,
  times = 10,
  seed = NULL,
  method = NULL,
  formula = NULL,
  se = TRUE,
  n = 80,
  span = 0.75,
  fullrange = FALSE,
  xseq = NULL,
  level = 0.95,
  method.args = list(),
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
times	A parameter used to control the number of values sampled from each distribution.
seed	Set the seed for the layers random draw, allows you to plot the same draw across multiple layers.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following:

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

`...` Other arguments passed on to `layer()`'s `params` argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the `position` argument, or aesthetics that are required can *not* be passed through `...`. Unknown arguments that are not part of the 4 categories below are ignored.

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

`method` Smoothing method (function) to use, accepts either `NULL` or a character vector, e.g. `"lm"`, `"glm"`, `"gam"`, `"loess"` or a function, e.g. `MASS::rlm` or `mgcv::gam`, `stats::lm`, or `stats::loess`. `"auto"` is also accepted for backwards compatibility. It is equivalent to `NULL`.

For `method = NULL` the smoothing method is chosen based on the size of the largest group (across all panels). `stats::loess()` is used for less than 1,000 observations; otherwise `mgcv::gam()` is used with `formula = y ~ s(x, bs = "cs")` with `method = "REML"`. Somewhat anecdotally, `loess` gives a better appearance, but is $O(N^2)$ in memory, so does not work for larger datasets.

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

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

`se` Display confidence band around smooth? (`TRUE` by default, see `level` to control.)

na.rm	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
orientation	The orientation of the layer. The default (NA) automatically determines the orientation from the aesthetic mapping. In the rare event that this fails it can be given explicitly by setting orientation to either "x" or "y". See the *Orientation* section for more detail.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. annotation_borders().
geom, stat	Use to override the default connection between geom_smooth() and stat_smooth(). For more information about overriding these connections, see how the stat and geom arguments work.
n	Number of points at which to evaluate smoother.
span	Controls the amount of smoothing for the default loess smoother. Smaller numbers produce wigglier lines, larger numbers produce smoother lines. Only used with loess, i.e. when method = "loess", or when method = NULL (the default) and there are fewer than 1,000 observations.
fullrange	If TRUE, the smoothing line gets expanded to the range of the plot, potentially beyond the data. This does not extend the line into any additional padding created by expansion.
xseq	A numeric vector of values at which the smoother is evaluated. When NULL (default), xseq is internally evaluated as a sequence of n equally spaced points for continuous data.
level	Level of confidence band to use (0.95 by default).
method.args	List of additional arguments passed on to the modelling function defined by method.

Value

A ggplot2 layer

Examples

```
library(ggplot2)
# ggplot
ggplot(mpg, aes(displ, hwy)) +
  geom_point() +
  geom_smooth()

# ggdibbbler
ggplot(uncertain_mpg, aes(displ, hwy)) +
  geom_point_sample(alpha=0.5, size=0.2, seed = 22) +
```

```
  geom_smooth_sample(linewidth=0.2, alpha=0.1, seed = 22)

# Smooths are automatically fit to each group (defined by categorical
# aesthetics or the group aesthetic) and for each facet.
# ggplot
ggplot(mpg, aes(displ, hwy, colour = class)) +
  geom_point() +
  geom_smooth(se = FALSE, method = lm)
# ggdibbler
ggplot(uncertain_mpg, aes(displ, hwy, colour = class)) +
  geom_point_sample(alpha=0.5, size=0.2, seed = 22) +
  geom_smooth_sample(linewidth=0.2, alpha=0.1,
    se = FALSE, method = lm, seed = 22)
```

`geom_spoke_sample` *Line segments parameterised by location, direction and distance, with uncertainty*

Description

Identical to geom_spoke except that it will accept a distribution in place of any of the usual aesthetics.

Usage

```
geom_spoke_sample(
  mapping = NULL,
  data = NULL,
  stat = "identity_sample",
  position = "identity",
  ...,
  times = 10,
  seed = NULL,
  arrow = NULL,
  arrow.fill = NULL,
  lineend = "butt",
  linejoin = "round",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

`mapping`	Set of aesthetic mappings created by `aes()`. If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.

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

If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`.

A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created.

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

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

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

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

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

`...` Other arguments passed on to `layer()`'s `params` argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the `position` argument, or aesthetics that are required can *not* be passed through `...`. Unknown arguments that are not part of the 4 categories below are ignored.

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

An example of this is `geom_area(stat = "density", adjust = 0.5)`. The stat's documentation lists which parameters it can accept.

- The `key_glyph` argument of `layer()` may also be passed on through `...`. This can be one of the functions described as key glyphs, to change the display of the layer in the legend.

`times` A parameter used to control the number of values sampled from each distribution.

`seed` Set the seed for the layers random draw, allows you to plot the same draw across multiple layers.

`arrow` specification for arrow heads, as created by `grid::arrow()`.

`arrow.fill` fill colour to use for the arrow head (if closed). `NULL` means use `colour` aesthetic.

`lineend` Line end style (round, butt, square).

`linejoin` Line join style (round, mitre, bevel).

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

`show.legend` logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use `TRUE`. If `NA`, all levels are shown in legend, but unobserved levels are omitted.

`inherit.aes` If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. `annotation_borders()`.

Value

A ggplot2 layer

Examples

```
library(ggplot2)
library(dplyr)
library(distributional)

# deterministic data
set.seed(1)
df <- expand.grid(x = 1:10, y=1:10)
df$angle <- runif(100, 0, 2*pi)
df$speed <- runif(100, 0, sqrt(0.1 * df$x))

# uncertain data
uncertain_df <- df |>
  group_by(x,y) |>
  mutate(angle = dist_normal(angle, runif(1,0, 0.5)),
         speed = dist_normal(speed, runif(1,0, 0.1))) |>
  ungroup()
```

```
# ggplot
ggplot(df, aes(x, y)) +
  geom_point() +
  geom_spoke(aes(angle = angle, radius = speed))

# ggdibbler
ggplot(uncertain_df, aes(x, y)) +
  geom_point_sample() + #' and here we used geom_point_sample
  geom_spoke_sample(aes(angle = angle, radius = speed), alpha=0.3)
```

`geom_violin_sample` *Violin plots with uncertainty*

Description

Identical to geom_violin, except that it will accept a distribution in place of any of the usual aesthetics.

Usage

```
geom_violin_sample(
  mapping = NULL,
  data = NULL,
  times = 10,
  seed = NULL,
  stat = "ydensity_sample",
  position = "dodge_identity",
  ...,
  trim = TRUE,
  bounds = c(-Inf, Inf),
  quantile.colour = NULL,
  quantile.color = NULL,
  quantile.linetype = 0L,
  quantile.linewidth = NULL,
  draw_quantiles = deprecated(),
  scale = "area",
  na.rm = FALSE,
  orientation = NA,
  show.legend = NA,
  inherit.aes = TRUE
)

stat_ydensity_sample(
  mapping = NULL,
  data = NULL,
  geom = "violin",
  position = "identity",
```

```
  ...,
  times = 10,
  seed = NULL,
  orientation = NA,
  bw = "nrd0",
  adjust = 1,
  kernel = "gaussian",
  trim = TRUE,
  scale = "area",
  drop = TRUE,
  bounds = c(-Inf, Inf),
  quantiles = c(0.25, 0.5, 0.75),
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
times	A parameter used to control the number of values sampled from each distribution.
seed	Set the seed for the layers random draw, allows you to plot the same draw across multiple layers.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.

`...` Other arguments passed on to `layer()`'s `params` argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the `position` argument, or aesthetics that are required can *not* be passed through `...`. Unknown arguments that are not part of the 4 categories below are ignored.

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

`trim` If `TRUE` (default), trim the tails of the violins to the range of the data. If `FALSE`, don't trim the tails.

`bounds` Known lower and upper bounds for estimated data. Default `c(-Inf, Inf)` means that there are no (finite) bounds. If any bound is finite, boundary effect of default density estimation will be corrected by reflecting tails outside `bounds` around their closest edge. Data points outside of bounds are removed with a warning.

`quantile.colour`, `quantile.color`, `quantile.linewidth`, `quantile.linetype`
Default aesthetics for the quantile lines. Set to `NULL` to inherit from the data's aesthetics. By default, quantile lines are hidden and can be turned on by changing `quantile.linetype`. Quantile values can be set using the `quantiles` argument when using `stat = "ydensity"` (default).

`draw_quantiles` **[Deprecated]** Previous specification of drawing quantiles.

`scale` if "area" (default), all violins have the same area (before trimming the tails). If "count", areas are scaled proportionally to the number of observations. If "width", all violins have the same maximum width.

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

`orientation` The orientation of the layer. The default (`NA`) automatically determines the orientation from the aesthetic mapping. In the rare event that this fails it can be given explicitly by setting `orientation` to either `"x"` or `"y"`. See the *Orientation* section for more detail.

show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. annotation_borders().
geom, stat	Use to override the default connection between geom_violin() and stat_ydensity(). For more information about overriding these connections, see how the stat and geom arguments work.
bw	The smoothing bandwidth to be used. If numeric, the standard deviation of the smoothing kernel. If character, a rule to choose the bandwidth, as listed in stats::bw.nrd(). Note that automatic calculation of the bandwidth does not take weights into account.
adjust	A multiplicate bandwidth adjustment. This makes it possible to adjust the bandwidth while still using the a bandwidth estimator. For example, adjust = 1/2 means use half of the default bandwidth.
kernel	Kernel. See list of available kernels in density().
drop	Whether to discard groups with less than 2 observations (TRUE, default) or keep such groups for position adjustment purposes (FALSE).
quantiles	A numeric vector with numbers between 0 and 1 to indicate quantiles marked by the quantile computed variable. The default marks the 25th, 50th and 75th percentiles. The display of quantiles can be turned on by setting quantile.linetype to non-blank when using geom = "violin" (default).

Value

A ggplot2 layer

Examples

```
library(ggplot2)
library(dplyr)
library(distributional)

# plot set up
p <- ggplot(mtcars,
  aes(factor(cyl), mpg))
q <- ggplot(uncertain_mtcars,
  aes(dist_transformed(cyl, factor, as.numeric), mpg))

# ggplot
p + geom_violin()
# ggdibbler
q + geom_violin_sample(alpha=0.1)

# Default is to trim violins to the range of the data. To disable:
```

```
# ggplot
p + geom_violin(trim = FALSE)
# ggdibbler
q + geom_violin_sample(trim = FALSE, alpha=0.1)
```

`position_dodge_nested` *Nested dodge positions*

Description

These functions use nested positioning for distributional data, where one of the positions is dodged. This allows you to set different position adjustments for the "main" and "distribution" parts of your plot.

Usage

```
position_dodge_dodge(
  width = NULL,
  preserve = "single",
  orientation = "x",
  reverse = FALSE
)

position_dodge_identity(
  width = NULL,
  preserve = "single",
  orientation = "x",
  reverse = FALSE
)

position_identity_dodge(
  width = NULL,
  preserve = "single",
  orientation = "x",
  reverse = FALSE
)
```

Arguments

`width`	Dodging width, when different to the width of the individual elements. This is useful when you want to align narrow geoms with wider geoms. See the examples.
`preserve`	Should dodging preserve the `"total"` width of all elements at a position, or the width of a `"single"` element?
`orientation`	Fallback orientation when the layer or the data does not indicate an explicit orientation, like `geom_point()`. Can be `"x"` (default) or `"y"`.

reverse If TRUE, will reverse the default stacking order. This is useful if you're rotating both the plot and legend.

Value

A ggplot2 position

Aesthetics

position_dodge() understands the following aesthetics. Required aesthetics are displayed in bold and defaults are displayed for optional aesthetics:

- order → NULL

Learn more about setting these aesthetics in vignette("ggplot2-specs").

Examples

```
library(ggplot2)

# ggplot dodge
ggplot(mpg, aes(class)) +
  geom_bar(aes(fill = drv),
           position = position_dodge(preserve = "single"))

# normal dodge without nesting
ggplot(uncertain_mpg, aes(class)) +
  geom_bar_sample(aes(fill = drv), position = "dodge")

# dodge_identity
ggplot(uncertain_mpg, aes(class)) +
  geom_bar_sample(aes(fill = drv), position = "dodge_identity", alpha=0.2)

# dodge_dodge
ggplot(uncertain_mpg, aes(class)) +
  geom_bar_sample(aes(fill = drv), position = "dodge_dodge")

# identity_dodge
ggplot(mpg, aes(class)) +
  geom_bar(aes(fill = drv), alpha=0.5, position = "identity")
ggplot(uncertain_mpg, aes(class)) +
  geom_bar_sample(aes(fill = drv), position = "identity_dodge", alpha=0.7)
```

position_identity_identity *Nested identity positions*

Description

These functions use nested positioning for distributional data, where both of the positions are an identity. This allows you to set different position adjustments for the "main" and "distribution" parts of your plot.

Usage

```
position_identity_identity()
```

Value

A ggplot2 position

Examples

```
# Standard ggplots often have a position adjustment to fix overplotting
# plot with overplotting
library(ggplot2)
ggplot(mpg, aes(class)) +
  geom_bar(aes(fill = drv), alpha=0.5,
           position = "identity")

# sometimes ggdibbler functions call for more control over these
# overplotting adjustments
ggplot(uncertain_mpg, aes(class)) +
  geom_bar_sample(aes(fill = drv), position = "identity", alpha=0.1)
# is the same as...
ggplot(uncertain_mpg, aes(class)) +
  geom_bar_sample(aes(fill = drv), position = "identity_identity", alpha=0.1)

# nested positions allows us to differentiate which postion adjustments
# are used for the plot groups vs the distribution samples
```

position_nest *Any combination of nested positions*

Description

This function lets you nest any two positions available in ggplot2 (your results may vary). This allows you to set different position adjustments for the "main" and "distribution" parts of your plot.

Usage

```
position_nest(position = "identity_identity")
```

Arguments

position	a character of the nested position you want to use

Value

A ggplot2 position

Examples

```
# nested positions allows us to differentiate which postion adjustments
# are used for the plot groups vs the distribution samples
library(ggplot2)
ggplot(uncertain_mpg, aes(class)) +
  geom_bar_sample(aes(fill = drv), alpha=0.9,
                  position = position_nest("stack_dodge"))
```

---

position_stack_nested *Nested stack positions*

Description

These functions use nested positioning for distributional data, where one of the positions is stacked. This allows you to set different position adjustments for the "main" and "distribution" parts of your plot.

Usage

```
position_stack_identity(vjust = 1, reverse = FALSE)

position_stack_dodge(
  vjust = 1,
  reverse = FALSE,
  width = NULL,
  preserve = "single",
  orientation = "x"
)
```

Arguments

vjust	Vertical adjustment for geoms that have a position (like points or lines), not a dimension (like bars or areas). Set to `0` to align with the bottom, `0.5` for the middle, and `1` (the default) for the top.
reverse	If `TRUE`, will reverse the default stacking order. This is useful if you're rotating both the plot and legend.

width	Dodging width, when different to the width of the individual elements. This is useful when you want to align narrow geoms with wider geoms. See the examples.
preserve	Should dodging preserve the "total" width of all elements at a position, or the width of a "single" element?
orientation	Fallback orientation when the layer or the data does not indicate an explicit orientation, like geom_point(). Can be "x" (default) or "y".

Value

A ggplot2 position

Examples

```
# Standard ggplots often have a position adjustment to fix overplotting
# plot with overplotting
library(ggplot2)
ggplot(mpg, aes(class)) +
  geom_bar(aes(fill = drv),
           position = "stack")

# normal stack warps the scale and doesn't communicate useful info
ggplot(uncertain_mpg, aes(class)) +
  geom_bar_sample(aes(fill = drv), position = "stack")

# stack_identity
ggplot(uncertain_mpg, aes(class)) +
  geom_bar_sample(aes(fill = drv), position = "stack_identity", alpha=0.2)

# stack_dodge
ggplot(uncertain_mpg, aes(class)) +
  geom_bar_sample(aes(fill = drv), position = "stack_dodge")
```

---

position_subdivide *Subdivide position aesthetic in a geometry*

Description

If the outline of a polygon is deterministic but the fill is random, you should use position subdivide rather than varying the alpha value. This subdivide position can be used with geom_polygon_sample (soon to be extended to others such as geom_sf, geom_map, etc).

Usage

```
position_subdivide()
```

Value

A ggplot2 position

Examples

```
library(ggplot2)
library(distributional)
library(dplyr)

# make data polygon with uncertain fill values
ids <- factor(c("1.1", "2.1", "1.2", "2.2", "1.3", "2.3"))

values <- data.frame(
  id = ids,
  value = c(1, 2, 3, 4, 5, 6)
)
positions <- data.frame(
  id = rep(ids, each = 4),
  x = c(2, 1, 1.1, 2.2, 1, 0, 0.3, 1.1, 2.2, 1.1, 1.2, 2.5, 1.1, 0.3,
        0.5, 1.2, 2.5, 1.2, 1.3, 2.7, 1.2, 0.5, 0.6, 1.3),
  y = c(-0.5, 0, 1, 0.5, 0, 0.5, 1.5, 1, 0.5, 1, 2.1, 1.7, 1, 1.5,
        2.2, 2.1, 1.7, 2.1, 3.2, 2.8, 2.1, 2.2, 3.3, 3.2)
)
datapoly <- merge(values, positions, by = c("id"))
uncertain_datapoly <- datapoly |>
  mutate(value = dist_uniform(value, value + 0.8))

# ggplot
ggplot(datapoly , aes(x = x, y = y)) +
  geom_polygon(aes(fill = value, group = id))

# ggdibbler
ggplot(uncertain_datapoly , aes(x = x, y = y)) +
  geom_polygon_sample(aes(fill = value, group = id), times=50,
                      position = "subdivide")
```

`sample_expand` *Simulate outcomes from dibble to make a tibble*

Description

Simulates outcomes from all distributions in the dataset to make an "expanded" data set that can be intepreted by ggplot2. This can be used to debug ggdibbler plots, or used to make an uncertainty visualisation for a geom that doesn't exist. If (and only if) you are implementing a ggdibbler version of a ggplot stat extension, you should use dibble_to_tibble instead.

Usage

```
sample_expand(data, times = 10, seed = NULL)

dibble_to_tibble(data, params)
```

Arguments

data	Distribution dataset to expand into samples
times	A parameter used to control the number of values sampled from each distribution.
seed	Set the seed for the layers random draw, allows you to get the same draw from repeated sample_expand calls
params	the params argument for the stat function sample_expand(uncertain_mpg, times=10)

Value

A data frame of resampled values from the input distributions

---

scale_continuous_distribution

Position scales for continuous distributions

Description

These scales allow for distributions to be passed to the x and y position by mapping distribution objects to continuous aesthetics. These scale can be used similarly to the scale_*_continuous functions, but they do not accept transformations. If you want to transform your scale, you should apply a transformation through the coord_* functions, as they are applied after the stat, so the existing ggplot infastructure can be used. For example, if you would like a log transformation of the x axis, plot + coord_transform(x = "log") would work fine.

Usage

```
scale_x_continuous_distribution(
  name = waiver(),
  breaks = waiver(),
  labels = waiver(),
  limits = NULL,
  expand = waiver(),
  oob = oob_keep,
  guide = waiver(),
  position = "bottom",
  sec.axis = waiver(),
  ...
)

scale_y_continuous_distribution(
  name = waiver(),
  breaks = waiver(),
  labels = waiver(),
  limits = NULL,
  expand = waiver(),
```

```
  oob = scales::oob_keep,
  guide = waiver(),
  position = "left",
  sec.axis = waiver(),
  ...
)
```

Arguments

name	The name of the scale. Used as the axis or legend title. If `waiver()`, the default, the name of the scale is taken from the first mapping used for that aesthetic. If `NULL`, the legend title will be omitted.
breaks	One of: • `NULL` for no breaks • `waiver()` for the default breaks computed by the transformation object • A numeric vector of positions • A function that takes the limits as input and returns breaks as output (e.g., a function returned by `scales::extended_breaks()`). Note that for position scales, limits are provided after scale expansion. Also accepts rlang lambda function notation.
labels	One of the options below. Please note that when `labels` is a vector, it is highly recommended to also set the `breaks` argument as a vector to protect against unintended mismatches. • `NULL` for no labels • `waiver()` for the default labels computed by the transformation object • A character vector giving labels (must be same length as `breaks`) • An expression vector (must be the same length as breaks). See ?plotmath for details. • A function that takes the breaks as input and returns labels as output. Also accepts rlang lambda function notation.
limits	One of: • `NULL` to use the default scale range • A numeric vector of length two providing limits of the scale. Use `NA` to refer to the existing minimum or maximum • A function that accepts the existing (automatic) limits and returns new limits. Also accepts rlang lambda function notation. Note that setting limits on positional scales will **remove** data outside of the limits. If the purpose is to zoom, use the limit argument in the coordinate system (see `coord_cartesian()`).
expand	For position scales, a vector of range expansion constants used to add some padding around the data to ensure that they are placed some distance away from the axes. Use the convenience function `expansion()` to generate the values for the expand argument. The defaults are to expand the scale by 5% on each side for continuous variables, and by 0.6 units on each side for discrete variables.
oob	One of:

- Function that handles limits outside of the scale limits (out of bounds). Also accepts rlang lambda function notation.
- The default (`scales::censor()`) replaces out of bounds values with `NA`.
- `scales::squish()` for squishing out of bounds values into range.
- `scales::squish_infinite()` for squishing infinite values into range.

`guide` A function used to create a guide or its name. See `guides()` for more information.

`position` For position scales, The position of the axis. `left` or `right` for y axes, `top` or `bottom` for x axes.

`sec.axis` `sec_axis()` is used to specify a secondary axis.

`...` Other arguments passed on to `scale_(x|y)_continuous()`

Value

A ggplot2 scale

Examples

```
library(ggplot2)
library(distributional)
set.seed(1997)
point_data <- data.frame(xvar = c(dist_uniform(2,3),
                                  dist_normal(3,2),
                                  dist_exponential(3)),
                         yvar = c(dist_gamma(2,1),
                                  dist_sample(x = list(rnorm(100, 5, 1))),
                                  dist_exponential(1)))
ggplot(data = point_data) +
  geom_point_sample(aes(x=xvar, y=yvar)) +
 scale_x_continuous_distribution(name="Hello, I am a random variable", limits = c(-5, 10)) +
  scale_y_continuous_distribution(name="I am also a random variable")
```

---

scale_discrete_distribution

Position scales for discrete distributions

Description

These scales allow for discrete distributions to be passed to the x and y position by mapping distribution objects to discrete aesthetics. These scale can be used similarly to the scale_*_discrete functions. If you want to transform your scale, you should apply a transformation through the coord_* functions, as they are applied after the stat, so the existing ggplot infastructure can be used.

Usage

```
scale_x_discrete_distribution(
  name = waiver(),
  palette = seq_len,
  expand = waiver(),
  guide = waiver(),
  position = "bottom",
  sec.axis = waiver(),
  continuous.limits = NULL,
  drop = TRUE,
  ...
)

scale_y_discrete_distribution(
  name = waiver(),
  palette = seq_len,
  expand = waiver(),
  guide = waiver(),
  position = "left",
  sec.axis = waiver(),
  continuous.limits = NULL,
  drop = TRUE,
  ...
)
```

Arguments

name	The name of the scale. Used as the axis or legend title. If `waiver()`, the default, the name of the scale is taken from the first mapping used for that aesthetic. If `NULL`, the legend title will be omitted.
palette	A palette function that when called with a single integer argument (the number of levels in the scale) returns the numerical values that they should take.
expand	For position scales, a vector of range expansion constants used to add some padding around the data to ensure that they are placed some distance away from the axes. Use the convenience function `expansion()` to generate the values for the expand argument. The defaults are to expand the scale by 5% on each side for continuous variables, and by 0.6 units on each side for discrete variables.
guide	A function used to create a guide or its name. See `guides()` for more information.
position	For position scales, The position of the axis. `left` or `right` for y axes, `top` or `bottom` for x axes.
sec.axis	`dup_axis()` is used to specify a secondary axis.
continuous.limits	One of: • `NULL` to use the default scale range

- A numeric vector of length two providing a display range for the scale. Use NA to refer to the existing minimum or maximum.
- A function that accepts the limits and returns a numeric vector of length two.

drop TRUE, will drop factor levels not associated with data

... Arguments passed on to discrete_scale

breaks One of:
- NULL for no breaks
- waiver() for the default breaks (the scale limits)
- A character vector of breaks
- A function that takes the limits as input and returns breaks as output. Also accepts rlang lambda function notation.

limits One of:
- NULL to use the default scale values
- A character vector that defines possible values of the scale and their order
- A function that accepts the existing (automatic) values and returns new ones. Also accepts rlang lambda function notation.

drop Should unused factor levels be omitted from the scale? The default, TRUE, uses the levels that appear in the data; FALSE includes the levels in the factor. Please note that to display every level in a legend, the layer should use show.legend = TRUE.

na.translate Unlike continuous scales, discrete scales can easily show missing values, and do so by default. If you want to remove missing values from a discrete scale, specify na.translate = FALSE.

na.value If na.translate = TRUE, what aesthetic value should the missing values be displayed as? Does not apply to position scales where NA is always placed at the far right.

aesthetics The names of the aesthetics that this scale works with.

minor_breaks One of:
- NULL for no minor breaks
- waiver() for the default breaks (none for discrete, one minor break between each major break for continuous)
- A numeric vector of positions
- A function that given the limits returns a vector of minor breaks. Also accepts rlang lambda function notation. When the function has two arguments, it will be given the limits and major break positions.

labels One of the options below. Please note that when labels is a vector, it is highly recommended to also set the breaks argument as a vector to protect against unintended mismatches.
- NULL for no labels
- waiver() for the default labels computed by the transformation object
- A character vector giving labels (must be same length as breaks)
- An expression vector (must be the same length as breaks). See ?plotmath for details.

- A function that takes the breaks as input and returns labels as output. Also accepts rlang lambda function notation.

`call` The `call` used to construct the scale for reporting messages.

`super` The super class to use for the constructed scale

Value

A ggplot2 scale

Examples

```
library(ggplot2)
# ggplot
ggplot(smaller_diamonds, aes(x = cut, y = clarity)) +
 geom_count(aes(size = after_stat(prop)))
# ggdibbler
ggplot(smaller_uncertain_diamonds, aes(x = cut, y = clarity)) +
 geom_count_sample(aes(size = after_stat(prop)), times=10, alpha=0.1)
```

---

scale_type.distribution

Sets scale for distributions

Description

Generates a single value from the distribution and uses it to set the default ggplot scale. The scale can be changed later in the ggplot by using any scale_* function

Usage

```
## S3 method for class 'distribution'
scale_type(x)
```

Arguments

`x`	value being scaled

Value

A character vector of scale types. The scale type is the ggplot scale type of the outcome of the distribution.

`smaller_uncertain_diamonds` *An uncertain (and shrunk down) version of the diamonds data from‘ggplot2‘*

Description

This dataset is a subset of the diamonds data. There is a deterministic version that is only a subset (smaller_diamonds) and a version that has random variables (uncertain_smaller_diamonds). The data is only a subset as the ggdibbler approach can take quite a long time when applied to the full sized diamonds data set. An uncertain version of the original diamonds data is also available as uncertain_diamonds, although it isn't used in any examples.

Usage

```
smaller_diamonds

uncertain_diamonds
```

Format

A data frame with almost 54000 observations and 10 variables:

price Binomial random variable - price in US dollars ($326–$18,823)

carat Normal random variable - weight of the diamond (0.2–5.01)

cut Categorical random variable - quality of the cut (Fair, Good, Very Good, Premium, Ideal)

color Categorical random variable - diamond colour, from D (best) to J (worst)

clarity Categorical random variable - a measurement of how clear the diamond is (I1 (worst), SI2, SI1, VS2, VS1, VVS2, VVS1, IF (best))

x Normal random variable - length in mm (0–10.74)

y Normal random variable - width in mm (0–58.9)

z Normal random variable - depth in mm (0–31.8)

depth Normal random variable - total depth percentage = z / mean(x, y) = 2 * z / (x + y) (43–79)

table Normal random variable - width of top of diamond relative to widest point (43–95)

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 1000 rows and 10 columns.

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 5000 rows and 20 columns.

StatConnectSample *Connect uncertain observations*

Description

Identical to stat_connect, except that it will accept a distribution in place of any of the usual aesthetics.

Usage

```
stat_connect_sample(
  mapping = NULL,
  data = NULL,
  geom = "path",
  position = "identity",
  ...,
  times = 10,
  seed = NULL,
  connection = "hv",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
geom	The geometric object to use to display the data for this layer. When using a stat_*() function to construct a layer, the geom argument can be used to override the default coupling between stats and geoms. The geom argument accepts the following: • A Geom ggproto subclass, for example GeomPoint. • A string naming the geom. To give the geom as a string, strip the function name of the geom_ prefix. For example, to use geom_point(), give the geom as "point".

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

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

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

`...` Other arguments passed on to `layer()`'s `params` argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the `position` argument, or aesthetics that are required can *not* be passed through `...`. Unknown arguments that are not part of the 4 categories below are ignored.

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

`times` A parameter used to control the number of values sampled from each distribution.

`seed` Set the seed for the layers random draw, allows you to plot the same draw across multiple layers.

`connection` A specification of how two points are connected. Can be one of the folloing:

- A string giving a named connection. These options are:
  - `"hv"` to first jump horizontally, then vertically.
  - `"vh"` to first jump vertically, then horizontally.
  - `"mid"` to step half-way between adjacent x-values.
  - `"linear"` to use a straight segment.

- A numeric matrix with two columns giving x and y coordinates respectively. The coordinates should describe points on a path that connect point A at location (0, 0) and point B at location (1, 1). At least one of these two points is expected to be included in the coordinates.

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

`show.legend` logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use `TRUE`. If `NA`, all levels are shown in legend, but unobserved levels are omitted.

`inherit.aes` If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. `annotation_borders()`.

Value

A ggplot2 layer

Examples

```
# set up data
library(ggplot2)
x <- seq(0, 1, length.out = 20)[-1]
smooth <- cbind(x, scales::rescale(1 / (1 + exp(-(x * 10 - 5)))))
zigzag <- cbind(c(0.4, 0.6, 1), c(0.75, 0.25, 1))

# ggplot
ggplot(head(economics, 10), aes(date, unemploy)) +
  stat_connect(aes(colour = "zigzag"), connection = zigzag) +
  stat_connect(aes(colour = "smooth"), connection = smooth) +
  geom_point()
# ggdibbler
ggplot(head(uncertain_economics, 10), aes(date, unemploy)) +
  stat_connect_sample(aes(colour = "zigzag"), connection = zigzag, seed=64) +
  stat_connect_sample(aes(colour = "smooth"), connection = smooth, seed=64) +
  geom_point_sample(seed=64)
```

StatEcdfSample *Compute uncertain empirical cumulative distributions*

Description

Identical to stat_ecdf, except that it will accept a distribution in place of any of the usual aesthetics.

Usage

```
stat_ecdf_sample(
  mapping = NULL,
  data = NULL,
  geom = "step",
  position = "identity",
  ...,
  times = 10,
  seed = NULL,
  n = NULL,
  pad = TRUE,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
geom	The geometric object to use to display the data for this layer. When using a stat_*() function to construct a layer, the geom argument can be used to override the default coupling between stats and geoms. The geom argument accepts the following: • A Geom ggproto subclass, for example GeomPoint. • A string naming the geom. To give the geom as a string, strip the function name of the geom_ prefix. For example, to use geom_point(), give the geom as "point". • For more information and other ways to specify the geom, see the layer geom documentation.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position.

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

`...` Other arguments passed on to `layer()`'s `params` argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the `position` argument, or aesthetics that are required can *not* be passed through `...`. Unknown arguments that are not part of the 4 categories below are ignored.

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

`times` A parameter used to control the number of values sampled from each distribution.

`seed` Set the seed for the layers random draw, allows you to plot the same draw across multiple layers.

`n` if NULL, do not interpolate. If not NULL, this is the number of points to interpolate with.

`pad` If `TRUE`, pad the ecdf with additional points (-Inf, 0) and (Inf, 1)

`na.rm` If `FALSE` (the default), removes missing values with a warning. If `TRUE` silently removes missing values.

`show.legend` logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use `TRUE`. If `NA`, all levels are shown in legend, but unobserved levels are omitted.

`inherit.aes` If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. `annotation_borders()`.

Examples

```
library(ggplot2)
library(dplyr)
library(distributional)
set.seed(44)
# df
df <- data.frame(
  x = c(rnorm(100, 0, 3), rnorm(100, 0, 10)),
  g = gl(2, 100)
)
uncertain_df <- df |>
  group_by(x) |>
  mutate(x = dist_normal(x, runif(1,0,5)),
         g_pred = dist_bernoulli(0.9-0.8*(2-as.numeric(g)))
  )
# ggplot
ggplot(df, aes(x)) +
  stat_ecdf(geom = "step")
# ggdibbler
ggplot(uncertain_df, aes(x)) +
  stat_ecdf_sample(geom = "step", alpha=0.3)

# Multiple ECDFs
# ggplot
ggplot(df, aes(x, colour = g)) +
  stat_ecdf()
# ggdibbler 1
ggplot(uncertain_df, aes(x, colour = g)) +
  stat_ecdf_sample(alpha=0.3)
```

StatEllipseSample *Compute normal data ellipses with uncertainty*

Description

Identical to stat_ellipse, except that it will accept a distribution in place of any of the usual aesthetics.

Usage

```
stat_ellipse_sample(
  mapping = NULL,
  data = NULL,
  geom = "path",
  position = "identity",
  ...,
  times = 10,
  seed = NULL,
  type = "t",
  level = 0.95,
```

```
  segments = 51,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
geom	The geometric object to use to display the data for this layer. When using a stat_*() function to construct a layer, the geom argument can be used to override the default coupling between stats and geoms. The geom argument accepts the following: • A Geom ggproto subclass, for example GeomPoint. • A string naming the geom. To give the geom as a string, strip the function name of the geom_ prefix. For example, to use geom_point(), give the geom as "point". • For more information and other ways to specify the geom, see the layer geom documentation.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.
...	Other arguments passed on to layer()'s params argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the position argument, or aesthetics that are required can *not* be passed through .... Unknown arguments that are not part of the 4 categories below are ignored. • Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, colour = "red" or linewidth

= 3. The geom's documentation has an **Aesthetics** section that lists the available options. The 'required' aesthetics cannot be passed on to the params. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data.

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

times — A parameter used to control the number of values sampled from each distribution.

seed — Set the seed for the layers random draw, allows you to plot the same draw across multiple layers.

type — The type of ellipse. The default "t" assumes a multivariate t-distribution, and "norm" assumes a multivariate normal distribution. "euclid" draws a circle with the radius equal to level, representing the euclidean distance from the center. This ellipse probably won't appear circular unless coord_fixed() is applied.

level — The level at which to draw an ellipse, or, if type="euclid", the radius of the circle to be drawn.

segments — The number of segments to be used in drawing the ellipse.

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

show.legend — logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.

inherit.aes — If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. annotation_borders().

Value

A ggplot2 layer

Examples

```
library(ggplot2)
```

```
library(distributional)
# ggplot
ggplot(faithful, aes(waiting, eruptions)) +
  geom_point() +
  stat_ellipse()
# ggdibbler
ggplot(uncertain_faithful, aes(waiting, eruptions)) +
  geom_point_sample() +
  stat_ellipse_sample()

# ggplot
ggplot(faithful, aes(waiting, eruptions, color = eruptions > 3)) +
  geom_point() +
  stat_ellipse(type = "t")
# ggdibbler
ggplot(uncertain_faithful,
       aes(waiting, eruptions,
           color = dist_transformed(eruptions,function(x) x > 3, identity))) +
  geom_point_sample() +
  stat_ellipse_sample(type = "t") +
  labs(colour = "eruptions > 3")

# ggplot
ggplot(faithful, aes(waiting, eruptions, fill = eruptions > 3)) +
  stat_ellipse(geom = "polygon")
# ggdibbler
ggplot(uncertain_faithful,
       aes(waiting, eruptions,
           fill = dist_transformed(eruptions, function(x) x > 3, identity))) +
  stat_ellipse_sample(geom = "polygon", alpha=0.1) +
  labs(fill = "eruptions > 3")
```

`StatIdentitySample` *Generates a sample from a distribution*

Description

Can think of as the ggdibbler equivalent to "stat_identity". It is the default stat that we used for most geoms.

Usage

```
stat_identity_sample(
  mapping = NULL,
  data = NULL,
  geom = "point",
  position = "identity",
  ...,
  times = 10,
```

```
  seed = NULL,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
geom	The geometric object to use to display the data for this layer. When using a stat_*() function to construct a layer, the geom argument can be used to override the default coupling between stats and geoms. The geom argument accepts the following: • A Geom ggproto subclass, for example GeomPoint. • A string naming the geom. To give the geom as a string, strip the function name of the geom_ prefix. For example, to use geom_point(), give the geom as "point". • For more information and other ways to specify the geom, see the layer geom documentation.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.
...	Other arguments passed on to layer()'s params argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the position argument, or aesthetics that are required can *not* be passed through .... Unknown arguments that are not part of the 4 categories below are ignored. • Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, colour = "red" or linewidth

= 3. The geom's documentation has an **Aesthetics** section that lists the available options. The 'required' aesthetics cannot be passed on to the params. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data.

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

times — A parameter used to control the number of values sampled from each distribution.

seed — Set the seed for the layers random draw, allows you to plot the same draw across multiple layers.

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

show.legend — logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.

inherit.aes — If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. annotation_borders().

Value

A ggplot2 geom representing a point_sample which can be added to a ggplot object

A ggplot2 layer

Examples

```
library(ggplot2)

p <- ggplot(mtcars, aes(wt, mpg))
p + stat_identity()

q <- ggplot(uncertain_mtcars, aes(wt, mpg))
q + stat_identity_sample(aes(colour = after_stat(drawID)))
```

`StatManualSample` *Manually compute transformations with uncertainty*

Description

Identical to stat_manual, except that it will accept a distribution in place of any of the usual aesthetics.

Usage

```
stat_manual_sample(
  mapping = NULL,
  data = NULL,
  geom = "point",
  position = "identity",
  ...,
  times = 10,
  seed = NULL,
  fun = identity,
  args = list(),
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

`mapping`	Set of aesthetic mappings created by `aes()`. If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.
`data`	The data to be displayed in this layer. There are three options: If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`. A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created. A `function` will be called with a single argument, the plot data. The return value must be a `data.frame`, and will be used as the layer data. A `function` can be created from a `formula` (e.g. `~ head(.x, 10)`).
`geom`	The geometric object to use to display the data for this layer. When using a `stat_*()` function to construct a layer, the `geom` argument can be used to override the default coupling between stats and geoms. The `geom` argument accepts the following: • A `Geom` ggproto subclass, for example `GeomPoint`.

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

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

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

... — Other arguments passed on to layer()'s params argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the position argument, or aesthetics that are required can *not* be passed through .... Unknown arguments that are not part of the 4 categories below are ignored.

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

times — A parameter used to control the number of values sampled from each distribution.

seed — Set the seed for the layers random draw, allows you to plot the same draw across multiple layers.

fun — Function that takes a data frame as input and returns a data frame or data frame-like list as output. The default (identity()) returns the data unchanged.

args — A list of arguments to pass to the function given in fun.

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

show.legend — logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.

inherit.aes — If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. annotation_borders().

Value

A ggplot2 layer

Examples

```
library(ggplot2)
library(distributional)
# A standard scatterplot
p <- ggplot(mtcars,
            aes(disp, mpg, colour = factor(cyl))) +
  geom_point()

q <- ggplot(uncertain_mtcars,
            aes(disp, mpg,
                colour = dist_transformed(cyl, factor, as.numeric))) +
  labs(colour="factor(cyl)") +
  geom_point_sample()

# Using a custom function
make_hull <- function(data) {
  hull <- chull(x = data$x, y = data$y)
  data.frame(x = data$x[hull], y = data$y[hull])
}

p + stat_manual(
  geom = "polygon",
  fun  = make_hull,
  fill = NA
)

q + stat_manual_sample(
  geom = "polygon",
  fun  = make_hull,
  fill = NA,
)
# Using the `transform` function with quoting
p + stat_manual(
  geom = "segment",
  fun  = transform,
  args = list(
    xend = quote(mean(x)),
    yend = quote(mean(y))
```

```
  )
)

q + stat_manual_sample(
  geom = "segment",
  fun  = transform,
  args = list(
    xend = quote(mean(x)),
    yend = quote(mean(y))
  )
)

# Using dplyr verbs with `vars()`
if (requireNamespace("dplyr", quietly = TRUE)) {

  # Get centroids with `summarise()`
  p + stat_manual(
    size = 10, shape = 21,
    fun  = dplyr::summarise,
    args = vars(x = mean(x), y = mean(y))
  )

  q + stat_manual_sample(
    size = 10, shape = 21,
    fun  = dplyr::summarise,
    args = vars(x = mean(x), y = mean(y))
  )
}
```

`StatQqLineSample` *A quantile-quantile plot with uncertainty*

Description

Identical to geom_qq, stat_qq, geom_gg_line, and stat_qq_line, except that they accept a distribution in place of any of the usual aesthetics.

Usage

```
geom_qq_line_sample(
  mapping = NULL,
  data = NULL,
  geom = "abline",
  position = "identity",
  ...,
  times = 10,
  seed = NULL,
  distribution = stats::qnorm,
  dparams = list(),
```

```
  line.p = c(0.25, 0.75),
  fullrange = FALSE,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

stat_qq_line_sample(
  mapping = NULL,
  data = NULL,
  geom = "abline",
  position = "identity",
  ...,
  times = 10,
  seed = NULL,
  distribution = stats::qnorm,
  dparams = list(),
  line.p = c(0.25, 0.75),
  fullrange = FALSE,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

geom_qq_sample(
  mapping = NULL,
  data = NULL,
  geom = "point",
  position = "identity",
  ...,
  times = 10,
  seed = NULL,
  distribution = stats::qnorm,
  dparams = list(),
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

stat_qq_sample(
  mapping = NULL,
  data = NULL,
  geom = "point",
  position = "identity",
  ...,
  times = 10,
  seed = NULL,
  distribution = stats::qnorm,
```

```
  dparams = list(),
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
geom	The geometric object to use to display the data for this layer. When using a stat_*() function to construct a layer, the geom argument can be used to override the default coupling between stats and geoms. The geom argument accepts the following: • A Geom ggproto subclass, for example GeomPoint. • A string naming the geom. To give the geom as a string, strip the function name of the geom_ prefix. For example, to use geom_point(), give the geom as "point". • For more information and other ways to specify the geom, see the layer geom documentation.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.
...	Other arguments passed on to layer()'s params argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the position argument, or aesthetics that are required can *not* be passed through .... Unknown arguments that are not part of the 4 categories below are ignored. • Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, colour = "red" or linewidth

= 3. The geom's documentation has an **Aesthetics** section that lists the available options. The 'required' aesthetics cannot be passed on to the params. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data.

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

times	A parameter used to control the number of values sampled from each distribution.
seed	Set the seed for the layers random draw, allows you to plot the same draw across multiple layers.
distribution	Distribution function to use, if x not specified
dparams	Additional parameters passed on to distribution function.
line.p	Vector of quantiles to use when fitting the Q-Q line, defaults defaults to c(.25, .75).
fullrange	Should the q-q line span the full range of the plot, or just the data
na.rm	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. annotation_borders().

Value

A ggplot2 layer

Examples

```
library(ggplot2)
library(distributional)
df <- data.frame(y = rt(200, df = 5))
uncertain_df <- data.frame(y=dist_normal(rt(200, df = 5), runif(200)))
```

```
# ggplot
p <- ggplot(df, aes(sample = y))
p + stat_qq() + stat_qq_line()

# ggdibbler
q <- ggplot(uncertain_df, aes(sample = y))
q + stat_qq_sample() +
  stat_qq_line_sample()

# Using to explore the distribution of a variable
# ggplot
ggplot(mtcars, aes(sample = mpg)) +
  stat_qq() +
  stat_qq_line()
# ggdibbler
ggplot(uncertain_mtcars, aes(sample = mpg)) +
  stat_qq_sample() +
  stat_qq_line_sample()
```

StatSummary2dSample *Bin and summarise in 2d (rectangle & hexagons) with uncertain inputs*

Description

Identical to stat_summary_2d, except that it will accept a distribution in place of any of the usual aesthetics.

Usage

```
stat_summary_2d_sample(
  mapping = NULL,
  data = NULL,
  geom = "tile",
  position = "identity_dodge",
  ...,
  times = 10,
  seed = NULL,
  binwidth = NULL,
  bins = 30,
  breaks = NULL,
  drop = TRUE,
  fun = "mean",
  fun.args = list(),
  boundary = 0,
  closed = NULL,
  center = NULL,
  na.rm = FALSE,
```

```
  show.legend = NA,
  inherit.aes = TRUE
)

stat_summary_hex_sample(
  mapping = NULL,
  data = NULL,
  geom = "hex",
  position = "identity",
  ...,
  times = 10,
  seed = NULL,
  binwidth = NULL,
  bins = 30,
  drop = TRUE,
  fun = "mean",
  fun.args = list(),
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
geom	The geometric object to use to display the data for this layer. When using a stat_*() function to construct a layer, the geom argument can be used to override the default coupling between stats and geoms. The geom argument accepts the following: • A Geom ggproto subclass, for example GeomPoint. • A string naming the geom. To give the geom as a string, strip the function name of the geom_ prefix. For example, to use geom_point(), give the geom as "point". • For more information and other ways to specify the geom, see the layer geom documentation.

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

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

`...` Other arguments passed on to `layer()`'s `params` argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the `position` argument, or aesthetics that are required can *not* be passed through `...`. Unknown arguments that are not part of the 4 categories below are ignored.

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

`times` A parameter used to control the number of values sampled from each distribution.

`seed` Set the seed for the layers random draw, allows you to plot the same draw across multiple layers.

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

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

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

breaks	Alternatively, you can supply a numeric vector giving the bin boundaries. Overrides binwidth, bins, center, and boundary. Can also be a function that takes group-wise values as input and returns bin boundaries.
drop	drop if the output of fun is NA.
fun	function for summary.
fun.args	A list of extra arguments to pass to fun
closed	One of "right" or "left" indicating whether right or left edges of bins are included in the bin.
center, boundary	bin position specifiers. Only one, center or boundary, may be specified for a single plot. center specifies the center of one of the bins. boundary specifies the boundary between two bins. Note that if either is above or below the range of the data, things will be shifted by the appropriate integer multiple of binwidth. For example, to center on integers use binwidth = 1 and center = 0, even if 0 is outside the range of the data. Alternatively, this same alignment can be specified with binwidth = 1 and boundary = 0.5, even if 0.5 is outside the range of the data.
na.rm	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. annotation_borders().

Examples

```
library(ggplot2)
d <- ggplot(smaller_diamonds,
            aes(carat, depth, z = price))
d + stat_summary_2d()

b <- ggplot(smaller_uncertain_diamonds,
            aes(carat, depth, z = price))
b + stat_summary_2d_sample()

# summary_hex
d + stat_summary_hex(fun = ~ sum(.x^2))
b + stat_summary_hex_sample(fun = ~ sum(.x^2), alpha=0.3)
```

StatSummaryBinSample *Summarise y values at unique/binned x with uncertainty*

Description

Identical to stat_summary, except that it will accept a distribution in place of any of the usual aesthetics.

Usage

```
stat_summary_bin_sample(
  mapping = NULL,
  data = NULL,
  times = 10,
  seed = NULL,
  geom = "pointrange",
  position = "identity",
  ...,
  fun.data = NULL,
  fun = NULL,
  fun.max = NULL,
  fun.min = NULL,
  fun.args = list(),
  bins = 30,
  binwidth = NULL,
  breaks = NULL,
  na.rm = FALSE,
  orientation = NA,
  show.legend = NA,
  inherit.aes = TRUE,
  fun.y = deprecated(),
  fun.ymin = deprecated(),
  fun.ymax = deprecated()
)

stat_summary_sample(
  mapping = NULL,
  data = NULL,
  times = 10,
  seed = NULL,
  geom = "pointrange",
  position = "identity",
  ...,
  fun.data = NULL,
  fun = NULL,
  fun.max = NULL,
  fun.min = NULL,
```

```
  fun.args = list(),
  na.rm = FALSE,
  orientation = NA,
  show.legend = NA,
  inherit.aes = TRUE,
  fun.y = deprecated(),
  fun.ymin = deprecated(),
  fun.ymax = deprecated()
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
times	A parameter used to control the number of values sampled from each distribution.
seed	Set the seed for the layers random draw, allows you to plot the same draw across multiple layers.
geom	The geometric object to use to display the data for this layer. When using a stat_*() function to construct a layer, the geom argument can be used to override the default coupling between stats and geoms. The geom argument accepts the following: • A Geom ggproto subclass, for example GeomPoint. • A string naming the geom. To give the geom as a string, strip the function name of the geom_ prefix. For example, to use geom_point(), give the geom as "point". • For more information and other ways to specify the geom, see the layer geom documentation.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter".

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

`...` Other arguments passed on to `layer()`'s `params` argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the `position` argument, or aesthetics that are required can *not* be passed through `...`. Unknown arguments that are not part of the 4 categories below are ignored.

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

`fun.data` A function that is given the complete data and should return a data frame with variables `ymin`, `y`, and `ymax`.

`fun.min`, `fun`, `fun.max`
Alternatively, supply three individual functions that are each passed a vector of values and should return a single number.

`fun.args` Optional additional arguments passed on to the functions.

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

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

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

`breaks` Alternatively, you can supply a numeric vector giving the bin boundaries. Overrides `binwidth` and `bins`.

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

`orientation` The orientation of the layer. The default (`NA`) automatically determines the orientation from the aesthetic mapping. In the rare event that this fails it can be

given explicitly by setting `orientation` to either `"x"` or `"y"`. See the *Orientation* section for more detail.

`show.legend` logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use `TRUE`. If `NA`, all levels are shown in legend, but unobserved levels are omitted.

`inherit.aes` If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. `annotation_borders()`.

`fun.ymin`, `fun.y`, `fun.ymax`
[Deprecated] Use the versions specified above instead.

Examples

```
library(ggplot2)
library(distributional)
d <- ggplot(mtcars, aes(cyl, mpg)) + geom_point()
b  <- ggplot(uncertain_mtcars, aes(cyl, mpg)) + geom_point_sample(seed=4)

d + stat_summary(fun = "median", colour = "red", geom = "point")
b + stat_summary_sample(fun = "median", colour = "red", geom = "point")

d + aes(colour = factor(vs)) + stat_summary(fun = mean, geom="line")
b + aes(colour = dist_transformed(vs, factor, as.numeric)) +
  stat_summary_sample(fun = mean, geom="line", seed=4) +
  labs(colour = "factor(vs)")
```

StatUniqueSample *Remove duplicates (with uncertainty?)*

Description

Identical to stat_unique, except that it will accept a distribution in place of any of the usual aesthetics. Note that the values will only be unique within each draw, (at the final plot level you might still have double ups).

Usage

```
stat_unique_sample(
  mapping = NULL,
  data = NULL,
  geom = "point",
  position = "identity",
  ...,
  times = 10,
  seed = NULL,
```

```
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by `aes()`. If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`. A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created. A `function` will be called with a single argument, the plot data. The return value must be a `data.frame`, and will be used as the layer data. A `function` can be created from a `formula` (e.g. `~ head(.x, 10)`).
geom	The geometric object to use to display the data for this layer. When using a `stat_*()` function to construct a layer, the `geom` argument can be used to override the default coupling between stats and geoms. The `geom` argument accepts the following: • A `Geom` ggproto subclass, for example `GeomPoint`. • A string naming the geom. To give the geom as a string, strip the function name of the `geom_` prefix. For example, to use `geom_point()`, give the geom as `"point"`. • For more information and other ways to specify the geom, see the layer geom documentation.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The `position` argument accepts the following: • The result of calling a position function, such as `position_jitter()`. This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the `position_` prefix. For example, to use `position_jitter()`, give the position as `"jitter"`. • For more information and other ways to specify the position, see the layer position documentation.
...	Other arguments passed on to `layer()`'s `params` argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the `position` argument, or aesthetics that are required can *not* be passed through .... Unknown arguments that are not part of the 4 categories below are ignored. • Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, `colour = "red"` or `linewidth = 3`. The geom's documentation has an **Aesthetics** section that lists the

available options. The 'required' aesthetics cannot be passed on to the `params`. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data.

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

`times` A parameter used to control the number of values sampled from each distribution.

`seed` Set the seed for the layers random draw, allows you to plot the same draw across multiple layers.

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

`show.legend` logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use `TRUE`. If `NA`, all levels are shown in legend, but unobserved levels are omitted.

`inherit.aes` If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. `annotation_borders()`.

Examples

```
library(ggplot2)
# ggplot
ggplot(mtcars, aes(vs, am)) +
  geom_point(alpha = 0.1)
# ggdibbler
ggplot(uncertain_mtcars, aes(vs, am)) +
  geom_point_sample(alpha = 0.01)

# ggplot
ggplot(mtcars, aes(vs, am)) +
  geom_point(alpha = 0.1, stat = "unique")
# ggdibbler
ggplot(uncertain_mtcars, aes(vs, am)) +
  geom_point_sample(alpha = 0.01, stat = "unique_sample")
```

toy_temp — *A toy data set that has the ambient temperature as measured by a collection of citizen scientists for each Iowa county*

Description

There are several measurements for each county, with no location marker for individual scientists to preserve anonyminity. Counties can have different numbers of observations as well as a different levels of variance between the observations in the county.

Format

A tibble with 99 observations and 4 variables

county_name the name of each Iowa county

recorded_temp the ambient temperature recorded by the citizen scientist

scientistID the ID number for the scientist who made the recording

county_geometry the shape file for each county of Iowa

county_longitude the centroid longitude for each county of Iowa

county_latitude the centroid latitude for each county of Iowa

toy_temp_dist — *A toy data set that provides data for a map with the temperature of each area represented by a random variable.*

Description

The map shows a wave pattern in temperature on the state of Iowa. Each estimate also has an uncertainty component added, and is represented as a distribution

Format

A tibble with 99 observations and 4 variables

county_name the name of each Iowa county

temp_dist the temperature of each county as a distribution

county_geometry the shape file for each county of Iowa

uncertain_economics *An uncertain version of the economics data from 'ggplot2'*

Description

This dataset is identical to the economics data, except that every variable in the data set is represented by a normal random variable. The original 'economics' dataset is based on real US economic time series data, but the uncertainty we added is hypothetical and included for illustrative purposes.

Usage

```
uncertain_economics_long
```

Format

A data frame with almost 574 observations and 6 variables:

date A deterministic variable - Month of data collection

pce Normal random variable - personal consumption expenditures, in billions of dollars

pop Normal random variable - total population, in thousands

psavert Normal random variable - personal savings rate

uempmed Normal random variable - median duration of unemployment, in weeks

unemploy Normal random variable - number of unemployed in thousands

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 2870 rows and 4 columns.

uncertain_faithful *Old Faithful data with uncertainty*

Description

The old faithful data from the datasets package but with added uncertainty.

Format

A data frame:

eruptions Eruption time in mins

waiting Waiting time to next eruption in mins

uncertain_faithfuld *2d density estimate of Old Faithful data with uncertainty*

Description

A 2d density estimate of the waiting and eruptions variables data faithful. Unlike other uncertain datasets, the only uncertain variable is density. Since this is based on a model, it wouldn't make sense for erruptions or waiting to be represented as random variables.

Format

A data frame with 5,625 observations and 3 variables:

eruptions Eruption time in mins

waiting Waiting time to next eruption in mins

density0 A 2d density estimate that is normally distributed with a low variance

density A 2d density estimate that is normally distributed with a medium variance

density2 A 2d density estimate that is normally distributed with a high variance

uncertain_mpg *An uncertain version of the MPG data from 'ggplot2'*

Description

This dataset is based on the Fuel economy data from 1999 to 2008 from 'ggplot2', but every value is represented by a distribution. Every variable in the data set is represetned by a categorical, discrete, or continuous random variable. The original MPG dataset in ggplot is a real a subset of the fuel economy data from the EPA, but the uncertainty is hypothetical uncertainty for each data type, added by us for illustrative purposes.

Format

A data frame with 234 rows and 11 variables:

manufacturer manufacturer, as a categorical random variable

model model name as a categorical random variable

displ engine displacement, as a uniform random variable to represent bounded data

year year of manufacture, as a sample of possible years

cyl number of cylinders, as a categorical random variable

trans type of transmission, as a categorical random variable

drv the type of drive train, as a categorical random variable

cty city miles per gallon, as a discrete random variable

hwy highway miles per gallon, as a discrete random variable

fl fuel type, as a categorical random variable

class "type" of car, as a categorical random variable

uncertain_mtcars *An uncertain version of the mtcars data from base R 'datasets'*

Description

This dataset is identical to the mtcars data, except that every variable in the data set is represented by a categorical, discrete, or continuous random variable. The original 'mtcars' dataset in datasets is based on real data extracted from the 1974 Motor Trend US magazine, but the uncertainty we added is hypothetical and included for illustrative purposes.

Format

A data frame with 32 observations and 11 variables:

mpg Uniform random variable - Miles/(US) gallon as

cyl Categorical random variable - Number of cylinders

disp Uniform random variable - Displacement (cu.in.)

hp Normal random variable - Gross horsepower

drat Uniform random variable - Rear axle ratio

wt Uniform random variable - Weight (1000 lbs)

qsec Uniform random variable - 1/4 mile time

vs Bernouli random variable - Engine (0 = V-shaped, 1 = straight)

am Bernouli random variable - Transmission (0 = automatic, 1 = manual)

gear Categorical random variable - Number of forward gears

carb Categorical random variable- Number of carburetors

walktober *Step Counts from Walktober 2025 Challenge*

Description

Daily step counts during October 2025 for five teams of four people competing in the Walktober 2025 Challenge.

Format

A data frame with 744 observations and 4 variables:

team Team name

name Name of team member

date Date steps were recorded

steps Number of steps recorded on 'date'

Index