Package 'ggrain'

January 23, 2026

Title A Rainclouds Geom for 'ggplot2'

Version 0.1.2

Description
The 'geom_rain()' function adds different geoms together using 'ggplot2' to create raincloud plots.

License MIT + file LICENSE

Encoding UTF-8

Depends ggplot2 (>= 4.0.0), R (>= 3.4.0)

Imports grid, ggpp (>= 0.5.6), rlang, vctrs (>= 0.5.0), cli

RoxygenNote 7.3.3

URL https://github.com/njudd/ggrain

BugReports https://github.com/njudd/ggrain/issues

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Nicholas Judd [aut, cre] (ORCID:
<https://orcid.org/0000-0002-0196-9871>),
Jordy van Langen [aut] (ORCID: <https://orcid.org/0000-0003-2504-2381>),
Micah Allen [ctb] (ORCID: <https://orcid.org/0000-0001-9399-4179>),
Rogier Kievit [aut] (ORCID: <https://orcid.org/0000-0003-0700-4568>)

Maintainer Nicholas Judd <nickkjudd@gmail.com>

Repository CRAN

Date/Publication 2026-01-23 10:30:02 UTC

Contents

`geom_paired_raincloud` *Paired raincloud plot*

Description

Taking from https://raw.githubusercontent.com/yjunechoe/geom_paired_raincloud/master/geom_paired_raincloud.R on 30-10-22 attribution to https://yjunechoe.github.io/

Usage

```
geom_paired_raincloud(
  mapping = NULL,
  data = NULL,
  stat = "ydensity",
  position = "dodge",
  trim = TRUE,
  scale = "area",
  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.

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

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

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().

... — 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.

Details

Create a paired raincloud plot (useful for visualizing difference between experimental conditions tested on the same subjects or items).

Adopted from the see::geom_violinhalf() source code from the see package

See Also

https://github.com/easystats/see/blob/master/R/geom_violinhalf.R

Examples

```
library(ggplot2)
```

`geom_rain` *Raincloud Plots*

Description

This function displays individual data points, a boxplot and half a violin plot. It also has the option to connect data points with lines across groups by specifying an id to connect by. Lastly, if desired one can color the dots based of another variable.

Usage

```
geom_rain(
  mapping = NULL,
  data = NULL,
  inherit.aes = TRUE,
  id.long.var = NULL,
  cov = NULL,
  rain.side = NULL,
  likert = FALSE,
  seed = 42,
  ...,
  point.args = rlang::list2(...),
 point.args.pos = rlang::list2(position = position_jitter(width = 0.04, height = 0, seed
    = seed)),
  line.args = rlang::list2(alpha = 0.2, ...),
 line.args.pos = rlang::list2(position = position_jitter(width = 0.04, height = 0, seed
    = seed), ),
  boxplot.args = rlang::list2(outlier.shape = NA, ...),
 boxplot.args.pos = rlang::list2(width = 0.05, position = position_nudge(x = 0.1), ),
  violin.args = rlang::list2(...),
 violin.args.pos = rlang::list2(side = "r", width = 0.7, quantiles = NULL, position =
    position_nudge(x = 0.15), )
)
```

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)).
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().
id.long.var	A group to connect the lines by - must be a string (e.g., "id").
cov	A covariate to color the dots by - must be as a string (e.g., "cov")
rain.side	How you want the raincloud displayed, right ("r"), left ("l") or flanking ("f"), for a 1-by-1 flanking raincloud use ("f1x1") and for a 2-by-2 use ("f2x2").
likert	Currently developing, right now just addes y-jitter.
seed	For the jittering in point & line to match.
...	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.

point.args	A list of args for the dots
point.args.pos	A list of positional args for the points
line.args	A list of args for the lines, you need to specify a group to connect them with id.long.var
line.args.pos	A list of positional args for the lines
boxplot.args	A list of args for the boxplot
boxplot.args.pos	A list of positional args for the boxplot
violin.args	A list of args for the violin
violin.args.pos	A list of positional args for the violin

Value

Returns a list of three environments to be used with the 'ggplot()' function in the 'ggplot2' package.

If the id.long.var argument is used the output will be a list of 4 environments.

These 4 environments have a similar structure to 'ggplot2::geom_boxplot()', 'ggplot2::geom_violin()', 'ggplot2::geom_point()' and 'ggplot2::geom_line()' from 'ggplot2'. need library(rlang) need library(ggplot2) depends = ggplot2

References

Allen, M., Poggiali, D., Whitaker, K., Marshall, T. R., van Langen, J., & Kievit, R. A. Raincloud plots: a multi-platform tool for robust data visualization Wellcome Open Research 2021, 4:63. https://doi.org/10.12688/wellcomeopenres.15191.2

Examples

```
e1 <- ggplot(iris, aes(Species, Sepal.Width, fill = Species))
e1 + geom_rain()

# x must be the discrete variable
# orinetation can be changed with coord_flip()
e1 + geom_rain(alpha = .5) + coord_flip()

# we can color the dots by a covariate
e1 + geom_rain(cov = "Sepal.Length")

# we can edit elements individually
e1 + geom_rain(violin.args = list(alpha = .3, color = NA))

# we can flip them
e1 + geom_rain(rain.side = 'l')
# and move them
e1 +
geom_rain(boxplot.args.pos = list(width = .1, position = position_nudge(x = -.2)))

# they also work longitudinally
```

```
e2 <- ggplot(sleep, aes(group, extra, fill = group))
e2 + geom_rain(id.long.var = "ID")

# we can add groups
sleep_dat <- cbind(sleep, data.frame(sex = c(rep("male", 5),
rep("female", 5), rep("male", 5), rep("female", 5))))
e3 <- ggplot(sleep_dat, aes(group, extra, fill = sex))
e3 + geom_rain(alpha = .6)

# add likert example
e4 <- ggplot(mpg, aes(1, hwy, fill = manufacturer))
e4 + geom_rain(likert= TRUE)

# lets make it look nicer
e4 + geom_rain(likert= TRUE,
 boxplot.args.pos = list(position = ggpp::position_dodgenudge(x = .095), width = .1),
 violin.args = list(color = NA, alpha = .5))
```

`stat_half_ydensity` *This is taken from gghalves great R package I am trying to remove the dependency*

Description

This is taken from gghalves great R package I am trying to remove the dependency

Usage

```
stat_half_ydensity(
  mapping = NULL,
  data = NULL,
  geom = "half_violin",
  position = "dodge",
  ...,
  bw = "nrd0",
  adjust = 1,
  kernel = "gaussian",
  trim = TRUE,
  scale = "area",
  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` This is only half_voilin

`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.

`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().
trim	If TRUE (default), trim the tails of the violins to the range of the data. If FALSE, don't trim the tails.
scale	if "area" (default), all violins have the same area (before trimming the tails). If "count", areas are scaled proportionally to the number of observations. If "width", all violins have the same maximum width.
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().

Index