Package ‘ggborderline’

August 23, 2025

URL https://github.com/wurli/ggborderline,
https://wurli.github.io/ggborderline/

Type Package

Title Line Plots that Pop

Version 0.3.0

Description A set of geometries to make line plots a little bit nicer. Use
along with 'ggplot2' to:
- Improve the clarity of line plots with many overlapping lines
- Draw more realistic worms.

License MIT + file LICENSE

Encoding UTF-8

Imports cli, ggplot2, rlang, scales, utils, vctrs

RoxygenNote 7.3.1

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation no

Author Jacob Scott [aut, cre]

Maintainer Jacob Scott <jscott2718@gmail.com>

Repository CRAN

Date/Publication 2025-08-22 23:00:02 UTC

Contents

geom_borderpath 2
scale_bordercolour_continuous 5

Index **6**

geom_borderpath *Connect observations*

Description

This set of geoms is very similar to `ggplot2::geom_path()`, `ggplot2::geom_line()` and `ggplot2::geom_step()`, with the only difference being that they accept two additional aesthetics, `bordercolour` and `borderwidth`.
For additional documentation, please refer to the ggplot2 geoms.

Usage

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

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

geom_borderstep(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  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 the override the default coupling between geoms and stats. The `stat` argument accepts the following: • A `Stat` ggproto subclass, for example `StatCount`. • A string naming the stat. To give the stat as a string, strip the function name of the `stat_` prefix. For example, to use `stat_count()`, give the stat as `"count"`. • For more information and other ways to specify the stat, see the layer stat documentation.
`position`	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The `position` argument accepts the following: • The result of calling a position function, such as `position_jitter()`. This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the `position_` prefix. For example, to use `position_jitter()`, give the position as `"jitter"`. • For more information and other ways to specify the position, see the layer position documentation.
`...`	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.

`lineend`	Line end style (round, butt, square).
`linejoin`	Line join style (round, mitre, bevel).
`linemitre`	Line mitre limit (number greater than 1).
`arrow`	Arrow specification, as created by `grid::arrow()`.
`na.rm`	If `FALSE`, the default, missing values are removed with a warning. If `TRUE`, missing values are silently removed.
`show.legend`	logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display.
`inherit.aes`	If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. `borders()`.
`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 ggproto layer object

Examples

```
require(ggplot2)

# geom_borderline() adds a border around lines
ggplot(economics_long, aes(date, value01, colour = variable)) +
  geom_borderline()

# You can control the linewidth and colour of the border with the
# borderwidth and bordercolour aesthetics:
ggplot(economics_long, aes(date, value01, bordercolour = variable)) +
  geom_borderline(borderwidth = .4, colour = "white")
```

```
# The background 'border' part of the geom is always solid, however this
# can be used to create some nice effects:
x <- seq(0, 4 * pi, length.out = 500)
test_data <- data.frame(
  x = rep(x, 2), y = c(sin(x), cos(x)),
  fun = rep(c("sin", "cos"), each = 500)
)
ggplot(test_data, aes(x, y, colour = fun)) +
  geom_borderline(linewidth = 1, linetype = "dashed", lineend = "round")
```

`scale_bordercolour_continuous` *Scales for borderlines*

Description

These scales control the linewidth and colour of the borders in borderlines. They work in much the same way as `ggplot2::scale_colour_continuous()`, `ggplot2::scale_linewidth_discrete()`, etc.

Usage

```
scale_bordercolour_continuous(..., aesthetics = "bordercolour")

scale_bordercolour_discrete(..., aesthetics = "bordercolour")

scale_borderwidth_continuous(..., aesthetics = "borderwidth")

scale_borderwidth_discrete(..., aesthetics = "borderwidth")
```

Arguments

`...`	Passed to the corresponding ggplot2 scales
`aesthetics`	Character string or vector of character strings listing the name(s) of the aesthetic(s) that this scale works with. This can be useful, for example, to apply colour settings to the bordercolour and colour aesthetics at the same time, via aesthetics = c("bordercolour", "colour").

Value

A ggproto scale object

Index

aes(), *3*

borders(), *4*

fortify(), *3*

geom_borderline *(*geom_borderpath*)*, 2
geom_borderpath, 2
geom_borderstep *(*geom_borderpath*)*, 2
ggplot(), *3*
grid::arrow(), *4*

key glyphs, *4*

layer position, *3*
layer stat, *3*
layer(), *3*, *4*

scale_bordercolour_continuous, 5
scale_bordercolour_discrete *(*scale_bordercolour_continuous*)*, 5
scale_borderwidth_continuous *(*scale_bordercolour_continuous*)*, 5
scale_borderwidth_discrete *(*scale_bordercolour_continuous*)*, 5