# Package: ggbumpribbon (via r-universe)

October 16, 2026

**Title** Sigmoid-Curved Ribbons for Rank Comparison Charts

**Version** 0.2.0

**Description** Provides 'ggplot2' geoms and stats for creating bump charts with sigmoid-curved filled ribbons and lines. Supports rank comparison visualizations where smooth S-shaped curves connect categorical positions across an ordered axis. Offers two interpolation methods (logistic sigmoid and cubic Hermite) with C1-continuous segment joins. Includes custom scales and themes optimized for bump chart readability.

**License** MIT + file LICENSE

**URL** https://github.com/sondreskarsten/ggbumpribbon

**BugReports** https://github.com/sondreskarsten/ggbumpribbon/issues

**Depends** R (>= 4.1.0)

**Imports** cli, ggplot2 (>= 3.5.0), rlang (>= 1.0.0), scales

**Suggests** knitr, rmarkdown, testthat (>= 3.0.0), vdiffr (>= 1.0.0)

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**Encoding** UTF-8

**Language** en-US

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.3

**Repository** https://sondreskarsten.r-universe.dev

**Date/Publication** 2026-03-19 20:10:52 UTC

**RemoteUrl** https://github.com/sondreskarsten/ggbumpribbon

**RemoteRef** HEAD

**RemoteSha** 3bd45d7a4ba1d33ea5d704c98e734770a64c6217

# Contents

geom_bump_line . . . . . 2
geom_bump_ribbon . . . . . 5
scale_fill_rank . . . . . 9
StatBumpLine . . . . . 10
theme_bump . . . . . 10

**Index** **12**

---

`geom_bump_line` *Smooth-curved lines between rank positions*

---

## Description

`geom_bump_line()` renders smooth curves connecting discrete rank positions across time periods. It is the line counterpart to `geom_bump_ribbon()`, producing stroked paths instead of filled areas.

## Usage

```
geom_bump_line(
  mapping = NULL,
  data = NULL,
  position = "identity",
  ...,
  smooth = 8,
  n = 100,
  method = "sigmoid",
  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:<br>If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`.<br>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.<br>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.

`smooth` Steepness of the sigmoid curve. Higher values produce sharper S-shaped transitions. Only used when `method = "sigmoid"`. Default is `8`.

`n` Number of interpolation points per segment. Default is `100`.

`method` Interpolation method. `"sigmoid"` (default) uses logistic S-curves with C1 derivative correction at segment joins. `"hermite"` uses cubic Hermite smoothstep interpolation via `stats::splinefunH()`. See `geom_bump_ribbon()` for details.

`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 that can be added to a plot.

## Aesthetics

`geom_bump_line()` understands the following aesthetics (required aesthetics are in **bold**):

- `x`
- `y`
- `alpha`
- `colour`
- `group`
- `linetype`
- `linewidth`

## Computed variables

`avg_y` Mean of y values in the group, inverse-transformed to original data space. Works correctly with `scale_y_reverse()`. Map to colour via `after_stat(avg_y)`.

## Multi-bend curves

The number of bends is controlled by the data, not by parameters. Two x-values produce one sigmoid. Four x-values (with the two middle points holding the start/end y positions) produce three sigmoids that mimic the "exit–channel–enter" pattern common in infographics:

```
df <- data.frame(
  x     = rep(c(1, 1.4, 1.6, 2), each = 5),
  y     = c(from, from, to, to),
  group = rep(groups, 4)
)
```

## See Also

`geom_bump_ribbon()`, `ggplot2::geom_path()`

Other bump geoms: `geom_bump_ribbon()`

## Examples

```
library(ggplot2)

# basic 2-point line bump
df <- data.frame(
  x     = rep(1:2, each = 5),
  y     = c(1, 2, 3, 4, 5, 3, 1, 5, 2, 4),
  group = rep(LETTERS[1:5], 2)
)
ggplot(df, aes(x, y, group = group, colour = after_stat(avg_y))) +
  geom_bump_line(linewidth = 1) +
  scale_colour_viridis_c() +
  scale_y_reverse()

# 3-bend pattern (exit-channel-enter)
df3 <- data.frame(
  x     = rep(c(1, 1.4, 1.6, 2), each = 5),
  y     = c(1,2,3,4,5, 1,2,3,4,5, 3,1,5,2,4, 3,1,5,2,4),
  group = rep(LETTERS[1:5], 4)
)
ggplot(df3, aes(x, y, group = group, colour = after_stat(avg_y))) +
  geom_bump_line(linewidth = 1.2) +
  scale_colour_viridis_c() +
  scale_y_reverse()
```

---

# geom_bump_ribbon *Smooth-curved filled ribbons for rank comparison*

---

## Description

`geom_bump_ribbon()` renders filled ribbons that follow smooth curves between discrete rank positions. It is the filled-area counterpart to ggbump's `geom_bump()` (https://github.com/davidsjoberg/ggbump).

## Usage

```
geom_bump_ribbon(
  mapping = NULL,
  data = NULL,
  position = "identity",
  ...,
  smooth = 8,
  n = 100,
  width = 0.8,
  method = "sigmoid",
  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:<br>If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot().<br>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.<br>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:<br>• The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position.<br>• 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".<br>• 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.<br>• 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.<br>• 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.<br>• 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.<br>• 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. |

| | |
|---|---|
| smooth | Steepness of the sigmoid curve. Higher values produce sharper S-shaped transitions. Only used when method = "sigmoid". Default is 8. |
| n | Number of interpolation points per segment. Default is 100. |
| width | Ribbon full width in data units. Default is 0.8. |
| method | Interpolation method. "sigmoid" (default) uses logistic S-curves with C1 derivative correction at segment joins. "hermite" uses cubic Hermite smoothstep interpolation via stats::splinefunH(). See section **Interpolation methods**. |
| 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 that can be added to a plot.

## Aesthetics

geom_bump_ribbon() understands the following aesthetics (required aesthetics are in **bold**):

- x
- y
- alpha
- colour
- fill
- group
- linetype
- linewidth

## Computed variables

ymin Lower ribbon boundary.

ymax Upper ribbon boundary.

avg_y Mean of y values in the group, inverse-transformed to original data space. Works correctly with scale_y_reverse() and other scale transforms. Map to fill via after_stat(avg_y).

### Interpolation methods

When a group has 3 or more time points, adjacent segments must be joined.

`"sigmoid"` **(default)** Each segment follows a logistic sigmoid $\sigma(t) = 1/(1 + e^{-t})$ on $t \in [-s, s]$ where $s$ is the `smooth` parameter. Endpoints are clamped to exact knot values by rescaling $\sigma$ to $[0, 1]$, and a Hermite-basis derivative correction drives the slope to zero at every interior knot, guaranteeing C1 (first-derivative) continuity across segments. The `smooth` parameter controls steepness: lower values (e.g. 2–3) give gentle S-curves, higher values (e.g. 12–15) give near-step-function transitions.

`"hermite"` Uses `stats::splinefunH()` with zero slopes at all knots (the cubic Hermite "smoothstep"). This is mathematically simpler: a single spline is evaluated across all knots, with C1 continuity guaranteed by construction. The `smooth` parameter is ignored. The visual shape is similar to `"sigmoid"` but not identical — the cubic polynomial $3t^2 - 2t^3$ has slightly different curvature distribution than the logistic function.

Both methods produce identical results for 2-point groups (a single segment has no join to worry about).

### See Also

`geom_bump_line()`, `ggplot2::geom_ribbon()`

Other bump geoms: `geom_bump_line()`

### Examples

```
library(ggplot2)

# basic: 5 items, 2 time points
df <- data.frame(
  x     = rep(1:2, each = 5),
  y     = c(1, 2, 3, 4, 5, 3, 1, 5, 2, 4),
  group = rep(LETTERS[1:5], 2)
)
ggplot(df, aes(x, y, group = group, fill = after_stat(avg_y))) +
  geom_bump_ribbon() +
  scale_fill_viridis_c() +
  scale_y_reverse()

# multi-period: 3 time points
df3 <- data.frame(
  x     = rep(1:3, each = 4),
  y     = c(1,2,3,4, 3,1,4,2, 2,4,1,3),
  group = rep(LETTERS[1:4], 3)
)
ggplot(df3, aes(x, y, group = group, fill = after_stat(avg_y))) +
  geom_bump_ribbon(alpha = 0.7) +
  scale_fill_viridis_c() +
  scale_y_reverse()

# Hermite method
```

```
ggplot(df3, aes(x, y, group = group, fill = after_stat(avg_y))) +
  geom_bump_ribbon(method = "hermite", alpha = 0.7) +
  scale_fill_viridis_c() +
  scale_y_reverse()

# mtcars: MPG rank vs HP rank
mt <- mtcars[1:10, ]
mt$car   <- rownames(mt)
mt_long  <- data.frame(
  x     = rep(1:2, each = 10),
  y     = c(rank(-mt$mpg), rank(-mt$hp)),
  group = rep(mt$car, 2)
)
ggplot(mt_long, aes(x, y, group = group, fill = after_stat(avg_y))) +
  geom_bump_ribbon() +
  scale_fill_gradientn(colours = c("#2ecc71", "#f7dc6f", "#eb4d4b"),
                       guide = "none") +
  scale_y_reverse() +
  theme_void()
```

---

`scale_fill_rank` *Rank-based gradient fill scale*

---

## Description

A convenience wrapper around `ggplot2::scale_fill_gradientn()` with defaults suited to rank comparison charts (green = best, red = worst).

## Usage

```
scale_fill_rank(
  colors = c("#2ecc71", "#a8e063", "#f7dc6f", "#f0932b", "#eb4d4b", "#c0392b"),
  limits = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| `colors` | Character vector of gradient colours. Default is a six-colour green-yellow-red ramp. |
| `limits` | Numeric vector of length 2 giving the scale limits, or `NULL` (default) to compute limits from the data range. |
| `...` | Passed to `ggplot2::scale_fill_gradientn()`. |

## Value

A ggplot2 scale object.

## See Also

ggplot2::scale_fill_gradientn(), geom_bump_ribbon()

Other bump scales: theme_bump()

## Examples

```
library(ggplot2)
df <- data.frame(
  x     = rep(1:2, each = 5),
  y     = c(1,2,3,4,5, 3,1,5,2,4),
  group = rep(LETTERS[1:5], 2)
)
ggplot(df, aes(x, y, group = group, fill = after_stat(avg_y))) +
  geom_bump_ribbon() +
  scale_fill_rank() +
  scale_y_reverse()
```

---

# StatBumpLine *Base ggproto classes for ggbumpribbon*

---

## Description

These ggproto objects implement the statistical transformations for bump ribbon and line charts. They are exported for extensibility but should typically be used through geom_bump_ribbon() and geom_bump_line().

## Value

ggproto objects that should not be called directly.

## See Also

ggplot2::Stat, ggplot2::ggproto()

---

# theme_bump *Dark theme for bump charts*

---

## Description

A minimal dark theme based on ggplot2::theme_void() with a dark background and light text, suited to rank comparison infographics.

## Usage

```
theme_bump(bg = "#1a1a2e", title_color = "#e74c3c", base_size = 10)
```

## Arguments

| | |
|---|---|
| bg | Background fill colour. Default "#1a1a2e". |
| title_color | Title text colour. Default "#e74c3c". |
| base_size | Base font size. Default 10. |

## Value

A ggplot2 theme object.

## See Also

ggplot2::theme_void(), geom_bump_ribbon()

Other bump scales: scale_fill_rank()

## Examples

```
library(ggplot2)
ggplot(mtcars, aes(wt, mpg)) +
  geom_point(colour = "white") +
  labs(title = "Motor Trend Cars") +
  theme_bump()
```

# Index


∗ **bump geoms**
    geom_bump_line, 2
    geom_bump_ribbon, 5
∗ **bump scales**
    scale_fill_rank, 9
    theme_bump, 10
∗ **datasets**
    StatBumpLine, 10

aes(), *2*, *6*
annotation_borders(), *4*, *7*

fortify(), *2*, *6*

geom_bump_line, 2, *8*
geom_bump_line(), *8*, *10*
geom_bump_ribbon, *4*, 5
geom_bump_ribbon(), *2–4*, *10*, *11*
ggbumpribbon-ggproto *(*StatBumpLine*)*, 10
ggplot(), *2*, *6*
ggplot2 layer, *4*, *7*
ggplot2::geom_path(), *4*
ggplot2::geom_ribbon(), *8*
ggplot2::ggproto(), *10*
ggplot2::scale_fill_gradientn(), *9*, *10*
ggplot2::Stat, *10*
ggplot2::theme_void(), *10*, *11*

key glyphs, *3*, *6*

layer position, *3*, *6*
layer(), *3*, *6*

scale_fill_rank, 9, *11*
StatBumpLine, 10
StatBumpRibbon *(*StatBumpLine*)*, 10
stats::splinefunH(), *3*, *7*, *8*

theme, *11*
theme_bump, 10, *10*