Package ‘ggarchery’

July 24, 2025

Type Package

Depends R (>= 4.0.0)

Imports ggplot2, purrr, magrittr, tidyr, dplyr, glue, rlang, grid

Title Flexible Segment Geoms with Arrows for 'ggplot2'

Version 0.4.4

Description Geoms for placing arrowheads at multiple points along a segment, not just at the end; position function to shift starts and ends of arrows to avoid exactly intersecting points.

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.2

URL https://github.com/mdhall272/ggarchery

Collate 'legend-draw-ggarchery.R' 'geom-arrowsegment.R' 'ggproto.R' 'position-attractsegment.R'

NeedsCompilation no

Author Matthew Hall [aut, cre]

Maintainer Matthew Hall <matthew.hall@lshtm.ac.uk>

Repository CRAN

Date/Publication 2025-07-24 13:10:08 UTC

Contents

`draw_key_arrowpath` *This function replaces `ggplot2::draw_key_path` and displays all the requested arrowheads.*

Description

This function replaces `ggplot2::draw_key_path` and displays all the requested arrowheads.

Usage

```
draw_key_arrowpath(data, params, size)
```

Arguments

`data`	A single row data frame containing the scaled aesthetics to display in this key
`params`	A list of additional parameters supplied to the geom.
`size`	Width and height of key in mm.

Value

A grid grob.

Examples

```
library(ggplot2)
library(magrittr)
library(tidyr)

# Generate some dummy data

ten.points <- data.frame(line.no = rep(1:5, each = 2), x = runif(10), y = runif(10),
                         position = rep(c("start", "end"), 5))
five.segments <- ten.points %>% pivot_wider(names_from = position, values_from = c(x,y))

ggplot(five.segments) +
   geom_point(data = ten.points, aes(x = x, y = y)) +
  geom_segment(aes(x = x_start, xend = x_end, y = y_start, yend = y_end), arrow = arrow(),
                key_glyph = draw_key_arrowpath)
```

geom_arrowsegment — *Line segments with flexible arrows*

Description

The basic `geom_arrowsegment()` is equivalent to `geom_segment(arrow = arrow())`. (It is assumed that the user wants some kind of arrow.) The extended functionality is to allow free placement of the arrowhead anywhere along the segment, and also multiple arrowheads, and to allow a fill aesthetic (which will only be visible for closed arrowheads).

The function works by dividing the line up into 1 or more segment grobs, each of which is generated by `grid::arrow()` except potentially the last (the one closest to the point (`xend`, `yend`)). The vector `arrow_positions`, whose entries must lie between 0 and 1, defines where each arrow segment ends, as a proportional position along the line. If the last entry of `arrow_positions` is 1, then the last grob has an arrow; otherwise it does not.

The function is designed with the expectation that arrows point from (`x`, `y`) to (`xend`, `yend`) but the `arrows` argument will happily accept `arrow(ends = "first")` or `arrow(ends = "both")` if you prefer. Just remember that the final segment is only an arrow at all if the last entry of `arrow_positions` is 1.

Usage

```
geom_arrowsegment(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
  arrows = list(arrow()),
  arrow_fills = NULL,
  arrow_positions = 1,
  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 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.

arrows	Either an arrow generated by grid::arrow() of a list of such arrows. In the former case or if the list has length 1, the arrowhead so defined is used every time; otherwise the list is expected to have the same length as arrow_positions and each segment defined by that argument is ended by the respective element of this one. The default is grid::arrow() with default parameters.
arrow_fills	A vector of fill colours for the arrowheads, behaves as the arrow_fill option in ggplot2::geom_segment. This will overrule a fill aesthetic in the same way that specifying a single fill outside aes specification will.
arrow_positions	A vector of distinct points on the unit interval. 0 is not permitted but arbitrarily small values are; 1 is permitted. The default behaviour is that arrowheads will be placed proportionally along the line connecting (x, y) to (xend,yend) at these points. In more detail: The first arrow segment begins at (x, y) and ends a proportional distance along the straight line joining (x, y) and (xend, yend) equal to the first entry of this vector. The second bridges the first two entries, and so on. If the final entry is 1 then the last segment is an arrow (and hence usually an arrowhead will be placed at the end of the line). If it is not, then the last segment is simply a line. These will be sorted into order from 0 to 1 if they are not already.
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.
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().

Value

A ggproto object

Examples

```
library(ggplot2)
library(magrittr)
library(tidyr)

# Generate some dummy data

ten.points <- data.frame(line.no = rep(1:5, each = 2), x = runif(10), y = runif(10),
                         position = rep(c("start", "end"), 5))
```

```
five.segments <- ten.points %>% pivot_wider(names_from = position, values_from = c(x,y))

# Default behaviour

ggplot(five.segments) +
   geom_point(data = ten.points, aes(x = x, y = y)) +
   geom_arrowsegment(aes(x = x_start, xend = x_end, y = y_start, yend = y_end))

# Midpoint arrowheads

ggplot(five.segments) +
   geom_point(data = ten.points, aes(x = x, y = y)) +
   geom_arrowsegment(aes(x = x_start, xend = x_end, y = y_start, yend = y_end),
                     arrow_positions = 0.5)

# Double arrows

ggplot(five.segments) +
   geom_point(data = ten.points, aes(x = x, y = y)) +
   geom_arrowsegment(aes(x = x_start, xend = x_end, y = y_start, yend = y_end),
                     arrow_positions = c(0.25, 0.75))

# Double arrows, last arrowhead at the end point

ggplot(five.segments) +
   geom_point(data = ten.points, aes(x = x, y = y)) +
   geom_arrowsegment(aes(x = x_start, xend = x_end, y = y_start, yend = y_end),
                     arrow_positions = c(0.25, 1))

# Double arrowheads of varying appearance and position

ggplot(five.segments) +
    geom_point(data = ten.points, aes(x = x, y = y)) +
    geom_arrowsegment(aes(x = x_start, xend = x_end, y = y_start, yend = y_end),
                      arrow_positions = c(0.25, 0.75),
                      arrows = list(arrow(angle = 45, type = "closed"),
                                    arrow(angle = 25, ends = "both")),
                      arrow_fills = "indianred")
```

---

position_attractsegment

Nudge points towards each other along a line

Description

This position function is primarily intended for use with `ggplot2::geom_segment()` or `geom_arrowsegment()`, and solves the problem that the user may, for reasons of clarity or aesthetics, not want their arrows to actually start or end at the position that they are "pointing from" or "pointing to". It works by shifting the points towards each other along the line joining them, by either a proportional amount or a fixed distance.

Usage

```
position_attractsegment(
  start_shave = 0,
  end_shave = 0,
  type_shave = c("proportion", "distance")
)
```

Arguments

`start_shave, end_shave`
: The amount of distance to "shave" off the line between (`x`, `y`) and (`xend`, `yend`), at, respectively, the start and the end. Can be zero; cannot be negative. Units are determined by `type_shave`.

`type_shave`
: If `"proportion"` (the default) then this is a proportion of the total line length. If "distance" then it is instead the raw distance along the line. The is only really recommended in combination with `ggplot2::coord_fixed()`; results can be quite odd otherwise.

Value

A ggproto object

Examples

```
library(ggplot2)
library(magrittr)
library(tidyr)

# Generate some dummy data

ten.points <- data.frame(line.no = rep(1:5, each = 2), x = runif(10), y = runif(10),
                         position = rep(c("start", "end"), 5))
five.segments <- ten.points %>% pivot_wider(names_from = position, values_from = c(x,y))

# Ten percent off the start and end

ggplot(five.segments) +
  geom_point(data = ten.points, aes(x = x, y = y)) +
  geom_arrowsegment(aes(x = x_start, xend = x_end, y = y_start, yend = y_end),
               position = position_attractsegment(start_shave = 0.1, end_shave = 0.1))

# Absolute distance of 0.02 at the end only

ggplot(five.segments) +
  geom_point(data = ten.points, aes(x = x, y = y)) +
  geom_arrowsegment(aes(x = x_start, xend = x_end, y = y_start, yend = y_end),
                  position = position_attractsegment(end_shave = 0.02,
                                                     type_shave = "distance")) +
  coord_fixed()
```

Index