Package 'ggquiver'

August 20, 2026

Version 0.5.0

Title Quiver Plots for 'ggplot2'

Description An extension of 'ggplot2' to provide quiver plots to visualise vector fields. This functionality is implemented using a geom to produce a new graphical layer, which allows aesthetic options. This layer can be overlaid on a map to improve visualisation of mapped data.

Depends R (>= 3.2.0)

Imports cli, ggplot2 (>= 3.5.0), grid

Suggests maps, sf, pkgdown, vdiffr, testthat

URL https://github.com/mitchelloharawild/ggquiver, https://pkg.mitchelloharawild.com/ggquiver/

BugReports https://github.com/mitchelloharawild/ggquiver/issues

License GPL-3

Encoding UTF-8

ByteCompile true

RoxygenNote 7.3.3

NeedsCompilation no

Author Mitchell O'Hara-Wild [aut, cre]

Maintainer Mitchell O'Hara-Wild <mail@mitchelloharawild.com>

Repository CRAN

Date/Publication 2026-08-20 14:10:22 UTC

Contents

geom_quiver 2

Index **6**

geom_quiver *Quiver plots for ggplot2*

Description

Displays the direction and length of vectors on a graph.

Usage

```
geom_quiver(
  mapping = NULL,
  data = NULL,
  stat = "quiver",
  position = "identity",
  center = FALSE,
  rescale = FALSE,
  vecsize = NULL,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  ...
)

GeomQuiver

stat_quiver(
  mapping = NULL,
  data = NULL,
  geom = "quiver",
  position = "identity",
  center = FALSE,
  rescale = FALSE,
  vecsize = NULL,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  ...
)

StatQuiver
```

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.

`center` If `FALSE` (the default), the vector lines will start at the specified x and y. If `TRUE`, the arrows will be centered about x and y.

`rescale` If `FALSE` (the default), the vectors will not be rescaled. If `TRUE`, the vectors given by (u, v) will be rescaled using the `scale` function.

`vecsize` By default (NULL), vectors sizing is automatically determined. If a grid can be identified, they will be scaled to the grid, if not, the vectors will not be scaled. By specifying a numeric input here, the length of all arrows can be adjusted. Setting vecsize to zero will prevent scaling the arrows.

`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().

... — 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.

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.

Format

An object of class GeomQuiver (inherits from GeomSegment, Geom, ggproto, gg) of length 3.

An object of class StatQuiver (inherits from Stat, ggproto, gg) of length 3.

Vector fields from functions

No special handling is needed to plot the vector field of a mathematical function: u and v are ordinary aesthetics, so a grid of coordinates from expand.grid() can be mapped through them directly, e.g. aes(u = cos(x), v = sin(y)). The automatic grid-based vecsize scaling detects the spacing of the generated grid and sizes the arrows to fit, so the resolution of the grid can be changed

freely without arrows overlapping. When u and v are more naturally computed together from a single function(x, y) (for example, a system of differential equations), evaluate it once over the grid and bind the result instead — see the last example below.

Computed variables

x centered x start position for velocity arrow

y centered y start position for velocity arrow

xend centered x end position for velocity arrow

yend centered y end position for velocity arrow

Examples

```
library(ggplot2)
# Quiver plots of mathematical functions
field <- expand.grid(x=seq(0,pi,pi/12), y=seq(0,pi,pi/12))
ggplot(field, aes(x=x,y=y,u=cos(x),v=sin(y))) +
  geom_quiver()

# Removing automatic scaling
ggplot(seals, aes(x=long, y=lat, u=delta_long, v=delta_lat)) +
  geom_quiver(vecsize=NULL) +
  borders("state")

# Vector field from a joint function of x and y
spiral <- function(x, y) data.frame(u = -y - 0.3*x, v = x - 0.3*y)
field <- expand.grid(x=seq(-2,2,length.out=11), y=seq(-2,2,length.out=11))
field <- cbind(field, spiral(field$x, field$y))
ggplot(field, aes(x, y, u=u, v=v)) +
  geom_quiver()
```

Index

∗ **datasets**
geom_quiver, 2

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

fortify(), *3*

geom_quiver, 2
GeomQuiver *(*geom_quiver*)*, 2
ggplot(), *3*

key glyphs, *4*

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

stat_quiver *(*geom_quiver*)*, 2
StatQuiver *(*geom_quiver*)*, 2