Package: ggfields (via r-universe)

October 30, 2026

Title Add Vector Field Layers to Ggplots

Version 0.0.7

Author Pepijn de Vries [aut, cre] (0000-0002-7961-6646)

Maintainer Pepijn de Vries <pepijn.devries@outlook.com>

Description Add vector field layers to ggplots. Ideal for visualising wind speeds, water currents, electric/magnetic fields, etc. Accepts data.frames, simple features (sf), and spatiotemporal arrays (stars) objects as input. Vector fields are depicted as arrows starting at specified locations, and with specified angles and radii.

Depends R (>= 4.1.0)

Imports dplyr (>= 1.1.4), ggplot2 (>= 3.4.4), grid (>= 4.1.0), rlang (>= 1.1.2), sf (>= 1.0-15), scales (>= 1.3.0)

Suggests CopernicusMarine, knitr, rmarkdown, stars (>= 0.6-4), testthat (>= 3.0.0), vdiffr (>= 1.0.7), svglite (>= 2.1.3)

License GPL (>= 3)

URL https://pepijn-devries.github.io/ggfields/, https://github.com/pepijn-devries/ggfields/

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Collate 'angle_correction.r' 'helpers.r' 'stat_fields.r' 'draw_key_fields.r' 'geom_fields.r' 'annotation_fields.r' 'data.r' 'ggfields-package.r' 'import.r' 'pythagoras.r' 'scale.r'

LazyData true

VignetteBuilder knitr

Config/testthat/edition 3

Config/pak/sysreqs libabsl-dev cmake libgdal-dev gdal-bin libgeos-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev

Repository https://pepijn-devries.r-universe.dev

Date/Publication 2025-09-05 17:02:26 UTC

RemoteUrl https://github.com/pepijn-devries/ggfields

RemoteRef master

RemoteSha b69aa8be2f42085b01ee6ca3fafb379519c0fdd1

Contents

angle_correction 2
annotation_fields 3
draw_key_fields 6
GeomFields 7
pythagoras 10
scale_radius_continuous 11
seawatervelocity 12
StatFields 12

Index **15**

`angle_correction` *Calculate correction for angle in the plot coordinate system*

Description

The angle of a vector may be distorted when your plot uses a different coordinate system than the one for which the angle is specified. If `data` is a simple feature object (sf), the angle will be corrected for the displayed coordinate reference system (crs). When the crs is missing, an aspect ratio of 1 is assumed. For any other data, the angle is corrected for the aspect ratio in the plot.

Usage

```
angle_correction(data, panel_params, coord)
```

Arguments

`data`	fortified data used in a `geom_fields()`. Should at least contain `numeric` columns `x`, `y` and `angle`.
`panel_params`	panel parameters as returned by GeomFields$setup_params()
`coord`	A coord object.

Details

This function is used by default by `geom_fields()`. For more details on why this correction is required and how to customize corrections please see `vignette("angle_correction")`.

Value

A `data.frame` with an additional `angle_correction` column. The corrected angle is given by `angle_correction + angle`.

Author(s)

Pepijn de Vries

Examples

```
if (requireNamespace("ggplot2") && requireNamespace("stars")) {
  library(ggplot2)
  library(stars)

  ggplot() +
    geom_fields(
      data    = seawatervelocity,
      mapping = aes(radius = as.numeric(v),
                    angle  = as.numeric(angle),
                    col    = as.numeric(v)),
 ## You can provide the `angle_correction()` as argument explicitly
 ## (it is already the default). Note that the plotted region requires
 ## hardly any correction for the angles.
                .angle_correction = angle_correction)
}
```

`annotation_fields` *Annotate a ggplot with vector fields*

Description

Functions exactly the same as `geom_fields()`, with that difference that this function does not train the `x` and `y` scales. This makes the data central, and uses this layer to support it. Consequently, `annotation_fields()` does not accept a `stat` argument.

Usage

```
annotation_fields(
  mapping = NULL,
  data = NULL,
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  max_radius = ggplot2::unit(0.5, "cm"),
  .angle_correction = angle_correction,
  arrow = grid::arrow(length = ggplot2::unit(0.2, "cm")),
  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	Can be one of four things: • NULL: in that case data from the parent ggplot call is inherited. • data.frame: you need to assign the x and y aesthetics. • sf object: it should contain a geometry column with only POINT geometries. • stars object: it will be converted automatically to an sf object.
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.
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.
max_radius	Maximum radius to which the radius aesthetic is scaled in the plot. You can use absolute ("e.g., "cm", "in", "pt") and relative ("npc") units to set its value. Default is 0.5 cm.
.angle_correction	Function to correct the angle in the aesthetics for the projection and/or aspect ratio used in the plot. When set to NULL the angle is not corrected and is treated as the angle in the final plot. A custom function can be provided which should accept at least three arguments (data, panel_params and coord). See angle_correction() and vignette("angle_correction") for more details.
arrow	specification for arrow heads, as created by grid::arrow().
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().
...	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.

Value

A ggplot2::layer_sf().

Author(s)

Pepijn de Vries

Examples

```
if (requireNamespace("stars") && requireNamespace("ggplot2")) {
  library(stars)
  library(ggplot2)

  data("seawatervelocity")
  sw_sub <- seawatervelocity[,8:13,1:5]

  ## Note that the `seawatervelocity` spans a much larger area,
  ## but the plot only focuses on `sw_sub`
  ggplot() +
    geom_stars(data = sw_sub) +
    annotation_fields(data = seawatervelocity,
                      aes(angle = as.numeric(atan2(vo, uo)),
                          radius = as.numeric(pythagoras(uo, vo)))) +
    labs(radius = "v [m/s]")
}
```

`draw_key_fields` *Key glyphs for 'radius' legends*

Description

Each geom has an associated function that draws the key when the geom needs to be displayed in a legend. These functions are called `draw_key_*()`, where `*` stands for the name of the respective key glyph. The key glyphs can be customized for individual geoms by providing a geom with the `key_glyph` argument (see `layer()` or examples below.)

Usage

```
draw_key_fields(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.

Details

The layer `geom_fields()` allows for a special aesthetic `radius`. This function draws a key glyph for this aesthetics, where the radius of the arrow corresponds with the scalar value listed with this radius. Note that the width of the key glyph cannot be adjusted by the aesthetic itself. Therefore, if your `max_radius` parameter exceeds the glyph width, you need to change the width of the guides yourself, see `vignette("radius_aes")` for more details.

Value

A grid grob

Author(s)

Pepijn de Vries

Examples

```
if (requireNamespace("ggplot2")) {
  library(ggplot2)
  p <- ggplot(economics, aes(date, psavert, color = "savings rate"))
  p + geom_line(key_glyph = "fields")
}
```

GeomFields *Arrows depicting a vector field*

Description

Visualise vector fields (such as, electric/magnetic fields, wind speed, or water currents) with arrows as a layer in a ggplot.

Usage

```
GeomFields

geom_fields(
  mapping = NULL,
  data = NULL,
  stat = "fields",
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  max_radius = ggplot2::unit(0.5, "cm"),
  .angle_correction = angle_correction,
  arrow = grid::arrow(length = ggplot2::unit(0.2, "cm")),
  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	Can be one of four things: • NULL: in that case data from the parent ggplot call is inherited. • data.frame: you need to assign the x and y aesthetics. • sf object: it should contain a geometry column with only POINT geometries. • stars object: it will be converted automatically to an sf object.
stat	The statistical transformation to use on the data for this layer. By default it is set to GeomFields() ("fields").
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.

`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.

`max_radius` Maximum radius to which the `radius` aesthetic is scaled in the plot. You can use absolute ("e.g., "cm", "in", "pt") and relative ("npc") units to set its value. Default is `0.5 cm`.

`.angle_correction`
Function to correct the angle in the aesthetics for the projection and/or aspect ratio used in the plot. When set to `NULL` the angle is not corrected and is treated as the angle in the final plot. A custom function can be provided which should accept at least three arguments (`data`, `panel_params` and `coord`). See `angle_correction()` and `vignette("angle_correction")` for more details.

`arrow` specification for arrow heads, as created by `grid::arrow()`.

`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()`.

`...` 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.

Format

An object of class `GeomFields` (inherits from `GeomSegment`, `Geom`, `ggproto`, `gg`) of length 8.

Details

Adds a layer with vector fields to a ggplot. In order to achieve this two special aesthetic are required: `radius` and `angle`.

Value

A layer which can be added to a ggplot.

Aesthetics

- `geometry|x`: Either a `geometry` column or `x` coordinate. In case of `geometry` the column should be of class sf::sfc_POINT. In case of `x`, it should be a `numeric vector`, and the aesthetic `y` needs to be specified as well. It specifies the location of the origin of each vector.
- `radius`: This aesthetic will be used to scale the radius of the vector arrows in the field you wish to display. The maximum radius of the arrows is given by parameter `max_radius`. See `vignette("radius_aes")` for more details.
- `angle`: This aesthetic represent the angles of the vectors in your field in radians. Contrary to the mathematical definition, an angle of 0 radians will point upwards (instead of to the right). This was chosen such because in most geographical applications an angle of zero degrees points Northwards. Before plotting these angles are corrected by the function passed to the `.angle_correction` argument. See `vignette("angle_corrections)` for more details.
- `y`: This aesthetic needs to be used in combination with the `x` aesthetic. It needs to be a `numeric vector`.
- `fill`: See `vignette("ggplot2-specs", "ggplot2")`
- `colour`: See `vignette("ggplot2-specs", "ggplot2")`
- `linetype`: See `vignette("ggplot2-specs", "ggplot2")`
- `linewidth`: See `vignette("ggplot2-specs", "ggplot2")`
- `alpha`: A variable to control the opacity of an element.

Author(s)

Pepijn de Vries

Examples

```
data(seawatervelocity)

if (requireNamespace("ggplot2") && requireNamespace("stars") &&
      requireNamespace("scales")) {
  library(ggplot2)
  library(stars)

  sw_df <- as.data.frame(seawatervelocity)
```

```
  ggplot(sw_df, aes(x = x, y = y, radius = as.numeric(v), angle = as.numeric(angle))) +
    geom_fields(max_radius = unit(0.5, "cm"), na.rm = TRUE)

  ggplot() +
    geom_fields(data    = seawatervelocity,
                mapping = aes(radius = as.numeric(v),
                              angle  = as.numeric(angle),
                              col    = as.numeric(v)),
                max_radius = unit(0.5, "cm")) +
    scale_colour_viridis_c()
}
```

`pythagoras` *A helper function to calculate vector lengths*

Description

Calculates the length of a vector using the Pythagorean theorem.

Usage

```
pythagoras(x, y)
```

Arguments

`x`	A `numeric` `vector` with the same length as `y`. It should represent the lengths of the first leg (cathetus) of right triangles.
`y`	A `numeric` `vector` with the same length as `x`. It should represent the lengths of the second leg (cathetus) of right triangles.

Value

Returns a `numeric` `vector` with the same length as `x` and `y`, reflecting the lengths of the hypotenuse of the right triangles.

Author(s)

Pepijn de Vries

Examples

```
pythagoras(x = c(1, 2), y = c(1, 2))
```

scale_radius_continuous *Vector field radius scales*

Description

Scales to set up the visualisation of the `radius` aesthetic. These scales are also automatically used in plot guides. Note that `scale_radius_identity()` does *not* exist as it would be impossible to relate such a scale to the `max_radius` parameter. For more details see `vignette("radius_aes")`.

Usage

```
scale_radius_continuous(..., range = c(1e-08, 1))

scale_radius_binned(..., range = c(1e-08, 1))

scale_radius_discrete(..., range = c(1e-08, 1))
```

Arguments

`...`	Arguments passed on to underpinning `ggplot2::scale_*` functions.
`range`	Relative output range of radii. Must lie between 0 and 1.

Value

An object of class Scale.

Author(s)

Pepijn de Vries

Examples

```
if (requireNamespace("ggplot2")) {
  library(ggplot2)
  data(seawatervelocity)

  g_num <-
    ggplot() +
    geom_fields(data = seawatervelocity,
                aes(radius = as.numeric(v), angle = as.numeric(angle)))
  g_discr <-
    ggplot() +
    geom_fields(data = seawatervelocity,
                aes(radius = cut(as.numeric(v), 4), angle = as.numeric(angle)))

  g_num + scale_radius_continuous()
  g_num + scale_radius_binned()
```

```
  g_discr + scale_radius_discrete()
}
```

`seawatervelocity` *A small subset of the global ocean physics analysis and forecast product*

Description

A small subset of ocean currents data retrieved with CopernicusMarine from the source listed below serving as an example.

Format

A stars object with `x`, `y`, `depth` and `time` dimensions. It has the attributes `vo` (northward seawater velocity [m/s]) and `uo` (eastward seawater velocity [m/s]).

References

E.U. Copernicus Marine Service Information; Global Ocean Physics Analysis and Forecast - GLOBAL_ANALYSISFORECA
(2016-10-14). doi:10.48670/moi00016

Examples

```
data("seawatervelocity")
```

`StatFields` *Stat method for geom_fields*

Description

Prepares data before being handled by `geom_fields()`

Usage

```
StatFields

stat_fields(
  mapping = NULL,
  data = NULL,
  geom = "fields",
  position = "identity",
  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	Can be one of four things: • NULL: in that case data from the parent ggplot call is inherited. • data.frame: you need to assign the x and y aesthetics. • sf object: it should contain a geometry column with only POINT geometries. • stars object: it will be converted automatically to an sf object.
geom	The layer type for which the data is prepared. In this case "fields".
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.
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().
...	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.

Format

An object of class `StatFields` (inherits from `StatSf`, `Stat`, `ggproto`, `gg`) of length 3.

Value

Returns a layer that can be further modified by `geom_fields()`.

Author(s)

Pepijn de Vries

Examples

```
stat_fields()
```

Index

∗ **datasets**
    GeomFields, 7
    StatFields, 12

aes(), *4*, *7*, *13*
angle_correction, 2
angle_correction(), *4*, *8*
annotation_fields, 3

borders(), *4*, *8*, *13*

coord, *2*
CopernicusMarine, *12*
crs, *2*

draw_key_fields, 6

geom_fields *(*GeomFields*)*, 7
geom_fields(), *2, 3*, *6*, *12*, *14*
GeomFields, 7
GeomFields$setup_params(), *2*
ggplot, *4*, *7*, *9*, *13*
ggplot2::layer_sf(), *5*
grid::arrow(), *4*, *8*

key glyphs, *5*, *8*, *14*

layer, *9*, *14*
layer position, *4*, *8*, *13*
layer(), *4–6*, *8*, *13, 14*

pythagoras, 10

Scale, *11*
scale_radius_binned
        *(*scale_radius_continuous*)*, 11
scale_radius_continuous, 11
scale_radius_discrete
        *(*scale_radius_continuous*)*, 11
seawatervelocity, 12
sf, *2*
sf object, *4*, *7*, *13*
sf::sfc_POINT, *9*
stars, *12*
stars object, *4*, *7*, *13*
stat_fields *(*StatFields*)*, 12
StatFields, 12

units, *4*, *8*