Package 'ggvfields'

March 15, 2025

Type Package

Title Vector Field Visualizations with 'ggplot2'

Version 1.0.0

Maintainer Dusty Turner <dusty.s.turner@gmail.com>

Description A 'ggplot2' extension for visualizing vector fields in two-dimensional space. Provides flexible tools for creating vector and stream field layers, visualizing gradients and potential fields, and smoothing vector and scalar data to estimate underlying patterns.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Depends R (>= 4.1.0), ggplot2 (>= 3.3.0)

Imports farver, tibble, numDeriv, deSolve, grid, utils, scales, cli

URL https://github.com/dusty-turner/ggvfields

BugReports https://github.com/dusty-turner/ggvfields/issues

Suggests ggdensity, knitr, rmarkdown, testthat (>= 3.0.0)

Date 2025-03-08

Config/testthat/edition 3

BuildVignettes true

NeedsCompilation no

Author Dusty Turner [aut, cre],
David Kahle [aut],
Rodney X. Sturdivant [aut]

Repository CRAN

Date/Publication 2025-03-15 17:10:05 UTC

Contents

`efield` *Electric field*

Description

The vector field generated by collection of fixed electrical charges, as dictated by Coulomb's law. This function is mainly used to provide examples for visualizing vector fields with ggvfields.

Usage

```
efield(u, charge_positions, charges, k = 1, q_test = +1)

efield_maker(
  charge_positions = rbind(c(-1, -1), c(1, 1)),
  charges = c(-1, +1),
  k = 1,
  q_test = +1
)
```

Arguments

`u`	The position of the test charge.
`charge_positions`	The positions of the fixed charges generating the electric field. Defaulted in `efield_maker()`.
`charges`	The charges of the points placed at the positions of charge_positions. Defaulted in `efield_maker()`.
`k`	The constant of proportionality, defaulted to 1. See examples for a more rigorous use of physical constants.
`q_test`	The test charge, defaulted to +1.

Value

A vector containing the force felt by the test charge on account of the electric field.

See Also

https://en.wikipedia.org/wiki/Coulomb%27s_law

Examples

```
# set a - charge at (-1,-1) and a + charge at (1,1)
charge_positions <- rbind(c(-1,-1), c(1,1))
charges <- c(-1, +1)


# calculate force on test charge (+1) at c(0,1), ignoring physical constants
efield(c(0,1), charge_positions, charges)


# efield_maker() simply wraps this function, defaulting to those charges
f <- efield_maker()
f(c(0,1))

ggplot() +
  geom_stream_field(fun = f, xlim = c(-2,2), ylim = c(-2,2)) +
  scale_color_viridis_c(trans = "log10")

# electric constant from https://en.wikipedia.org/wiki/Vacuum_permittivity
ep0 <- 8.854187818814e-12
k <- (4*pi*ep0)^-1
efield(c(0,1), charge_positions, charges, k)
```

---

geom_gradient_field *Create a Gradient Field Layer in ggplot2*

Description

These functions provide convenient ggplot2 layers for drawing gradient fields by computing the gradient of a scalar field. A user-defined function (`fun`) specifies the behavior of the scalar field by taking a numeric vector of length 2 (representing (x, y)) and returning a single numeric value. The underlying StatStreamField computes the gradient via numerical differentiation (using `numDeriv::grad()`) and GeomStream renders the resulting vectors.

Usage

```
geom_gradient_field(
  mapping = NULL,
  data = NULL,
  stat = StatStreamField,
```

```
  position = "identity",
  ...,
  na.rm = FALSE,
  show.legend = TRUE,
  inherit.aes = TRUE,
  fun,
  xlim = NULL,
  ylim = NULL,
  n = 11,
  max_it = 1000,
  T = NULL,
  L = NULL,
  center = TRUE,
  type = "vector",
  normalize = TRUE,
  tail_point = FALSE,
  eval_point = FALSE,
  grid = NULL,
  lineend = "butt",
  linejoin = "round",
  linemitre = 10,
 arrow = grid::arrow(angle = 30, length = grid::unit(0.02, "npc"), type = "closed")
)

stat_gradient_field(
  mapping = NULL,
  data = NULL,
  geom = GeomStream,
  position = "identity",
  ...,
  na.rm = FALSE,
  show.legend = TRUE,
  inherit.aes = TRUE,
  fun,
  xlim = NULL,
  ylim = NULL,
  n = 11,
  max_it = 1000,
  T = NULL,
  L = NULL,
  center = TRUE,
  type = "vector",
  normalize = TRUE,
  tail_point = FALSE,
  eval_point = FALSE,
  grid = NULL,
  lineend = "butt",
  linejoin = "round",
```

```
  linemitre = 10,
  arrow = grid::arrow(angle = 30, length = grid::unit(0.02, "npc"), type = "closed")
)

geom_gradient_field2(
  mapping = NULL,
  data = NULL,
  stat = StatStreamField,
  position = "identity",
  ...,
  na.rm = FALSE,
  show.legend = TRUE,
  inherit.aes = TRUE,
  fun,
  xlim = NULL,
  ylim = NULL,
  n = 11,
  max_it = 1000,
  T = NULL,
  L = NULL,
  center = FALSE,
  type = "stream",
  normalize = TRUE,
  tail_point = TRUE,
  eval_point = FALSE,
  grid = NULL,
  lineend = "butt",
  linejoin = "round",
  linemitre = 10,
  arrow = NULL
)

stat_gradient_field2(
  mapping = NULL,
  data = NULL,
  geom = GeomStream,
  position = "identity",
  ...,
  na.rm = FALSE,
  show.legend = TRUE,
  inherit.aes = TRUE,
  fun,
  xlim = NULL,
  ylim = NULL,
  n = 11,
  max_it = 1000,
  T = NULL,
  L = NULL,
```

```
  center = FALSE,
  type = "stream",
  normalize = TRUE,
  tail_point = TRUE,
  eval_point = FALSE,
  grid = NULL,
  lineend = "butt",
  linejoin = "round",
  linemitre = 10,
  arrow = NULL
)
```

Arguments

mapping	A set of aesthetic mappings created by `ggplot2::aes()`. Additional aesthetics such as `color`, `size`, `linetype`, and `alpha` can be defined. In **geom_gradient_field** the default mapping includes `color = after_stat(norm)`, whereas in **geom_gradient_field2** the default mapping includes `length = after_stat(norm)`.
data	A data frame containing the input data.
stat	The statistical transformation to use on the data for this layer. Defaults to StatStreamField.
position	Position adjustment, either as a string or the result of a position adjustment function.
...	Other arguments passed on to `grid::layer()`.
na.rm	Logical. If `FALSE` (the default), missing values are removed with a warning.
show.legend	Logical. Should this layer be included in the legends?
inherit.aes	Logical. If `FALSE`, overrides the default aesthetics rather than combining with them.
fun	A function that defines the scalar field. It should take a numeric vector of length 2 (representing (x, y)) and return a single numeric value. **(Required)**
xlim	Numeric vector of length two. Specifies the limits of the x-axis domain. Defaults to `c(-1, 1)`.
ylim	Numeric vector of length two. Specifies the limits of the y-axis domain. Defaults to `c(-1, 1)`.
n	Integer. Grid resolution specifying the number of seed points along each axis. Higher values produce a denser gradient field. Defaults to `11`.
max_it	Integer. Maximum number of integration steps allowed when computing the gradient stream. Defaults to `1000`.
T	Numeric. Time increment used for numerical integration when `normalize` is FALSE. If not provided, it is computed automatically based on grid spacing and the vector field's magnitude.
L	Numeric. Target length for the gradient vectors or streamlines. When `normalize` is TRUE, computed vectors are scaled to have length L. If not provided, L is computed automatically from the grid spacing.

center	Logical. If TRUE, centers the seed points so that the original (x, y) becomes the midpoint.
type	Character. Specifies the type of field to compute: use "stream" to generate integrated streamlines or "vector" for individual vector segments. Defaults to "stream".
normalize	Logical. If TRUE, gradient vectors are normalized based on grid spacing. Defaults to TRUE.
tail_point	Logical. If TRUE, a point is drawn at the tail of each gradient vector.
eval_point	Logical. If TRUE, a point is drawn at the evaluation point where the gradient was computed. Defaults to FALSE.
grid	A data frame containing precomputed grid points for seed placement. If NULL (default), a regular Cartesian grid is generated based on xlim, ylim, and n.
lineend	Line end style (round, butt, square).
linejoin	Line join style (round, mitre, bevel).
linemitre	Line mitre limit (number greater than 1).
arrow	A grid::arrow() specification to add arrowheads to the gradient vectors. In **geom_gradient_field**, the default is a closed arrow with a 30° angle and length 0.02 npc; in geom_gradient_field2(), the default is NULL.
geom	The geometric object used to render the streamline (only used in stat_stream(); defaults to GeomStream).

Details

Two variants are provided:

- **geom_gradient_field()** uses a default mapping that sets color = after_stat(norm).
- **geom_gradient_field2()** uses a default mapping that sets length = after_stat(norm) (with color unmapped by default).

Value

A ggplot2 layer that computes and plots a gradient field by numerically differentiating a scalar field.

Aesthetics

geom_gradient_field() and geom_gradient_field2() understand the following aesthetics (required aesthetics are in **bold**):

- x: The x-coordinate of the seed point.
- y: The y-coordinate of the seed point.
- color: In **geom_gradient_field**, the color of the gradient vector. In **geom_gradient_field2**, color is not mapped by default.
- length: In **geom_gradient_field2**, the computed vector norm.
- size, linetype, alpha: Additional aesthetics to control appearance.

Computed Variables

The following variables are computed internally by StatStreamField when generating the gradient field from a scalar function:

norm The Euclidean norm of the gradient vector, calculated as $\sqrt{fx^2 + fy^2}$. This value is used, by default, for mapping color or scaling arrow lengths in the visualization.

avg_spd This variable may represent an average speed computed from the gradient magnitude. In the default mapping for **geom_gradient_field**, the color aesthetic is mapped to `after_stat(avg_spd)`.

Examples

```
Si <- matrix(c(1, 0.75, 0.75, 1), nrow = 2)
f <- function(u) exp(-as.numeric(u %*% solve(Si) %*% u) / 2) / (2 * pi * det(Si))

ggplot() +
  geom_gradient_field(fun = f, xlim = c(-3, 3), ylim = c(-3, 3))


df <- expand.grid(x = seq(-3, 3, 0.1), y = seq(-3, 3, 0.1)) |>
  transform(fxy = apply(cbind(x, y), 1, f))

ggplot() +
  geom_raster(aes(x, y, fill = fxy), data = df) +
  geom_gradient_field(fun = f, xlim = c(-3, 3), ylim = c(-3, 3)) +
  coord_equal()

fxy <- function(x, y) apply(cbind(x,y), 1, f)

ggplot() +
  ggdensity::geom_hdr_fun(fun = fxy, xlim = c(-3,3), ylim = c(-3,3)) +
  geom_gradient_field(fun = f, xlim = c(-3,3), ylim = c(-3,3)) +
  coord_equal()

  library("ggdensity")
  fxy <- function(x, y) apply(cbind(x, y), 1, f)
  fxy(1, 2)
  f(1:2)

  ggplot() +
    geom_hdr_fun(fun = fxy, xlim = c(-3, 3), ylim = c(-3, 3)) +
    geom_gradient_field(fun = f, xlim = c(-3, 3), ylim = c(-3, 3)) +
    coord_equal()
```

`geom_gradient_smooth` *Create a Gradient Smoothed Field Layer*

Description

geom_gradient_smooth() creates a ggplot2 layer that visualizes the gradient of a scalar field computed from raw data. A linear model is fitted using the supplied formula (default: z ~ x + y + I(x^2) + I(y^2)) on the raw data, and the numerical gradient is computed using numDeriv::grad(). The computed gradient field is then visualized using GeomStream().

Usage

```
geom_gradient_smooth(
  mapping = NULL,
  data = NULL,
  stat = StatStreamField,
  position = "identity",
  ...,
  na.rm = FALSE,
  show.legend = TRUE,
  inherit.aes = TRUE,
  formula = z ~ x + y + I(x^2) + I(y^2),
  xlim = NULL,
  ylim = NULL,
  n = 11,
  max_it = 1000,
  T = NULL,
  L = NULL,
  center = TRUE,
  type = "vector",
  normalize = TRUE,
  tail_point = FALSE,
  eval_point = FALSE,
  grid = NULL,
  lineend = "butt",
  linejoin = "round",
  linemitre = 10,
 arrow = grid::arrow(angle = 30, length = grid::unit(0.02, "npc"), type = "closed")
)

stat_gradient_smooth(
  mapping = NULL,
  data = NULL,
  geom = GeomStream,
  position = "identity",
  ...,
  na.rm = FALSE,
  show.legend = TRUE,
  inherit.aes = TRUE,
  formula = z ~ x + y + I(x^2) + I(y^2),
  xlim = NULL,
  ylim = NULL,
```

```
  n = 11,
  max_it = 1000,
  T = NULL,
  L = NULL,
  center = TRUE,
  type = "vector",
  normalize = TRUE,
  tail_point = FALSE,
  eval_point = FALSE,
  grid = NULL,
  lineend = "butt",
  linejoin = "round",
  linemitre = 10,
 arrow = grid::arrow(angle = 30, length = grid::unit(0.02, "npc"), type = "closed")
)
```

Arguments

mapping	A set of aesthetic mappings created by ggplot2::aes(). **Required:** Must include x and y; vector displacements are defined by fx and fy.
data	A data frame containing the raw vector data.
stat	The statistical transformation to use on the data. Defaults to "vector_smooth".
position	Position adjustment, either as a string or the result of a position adjustment function.
...	Additional arguments passed to the layer.
na.rm	Logical. If FALSE (the default), missing values are removed with a warning.
show.legend	Logical. Should this layer be included in the legends?
inherit.aes	Logical. If FALSE, overrides the default aesthetics rather than combining with them.
formula	A formula specifying the linear model for the scalar field. Defaults to z ~ x + y + I(x^2) + I(y^2).
xlim	Numeric vector of length 2 specifying the domain limits in the x-direction. Defaults to $c(-1, 1)$.
ylim	Numeric vector of length 2 specifying the domain limits in the y-direction. Defaults to $c(-1, 1)$.
n	An integer vector specifying the grid resolution for smoothing.
max_it	Maximum number of iterations for field integration (when used in streamlines).
T	If normalize = FALSE, this controls the time length for growing streams.
L	If normalize = TRUE, this controls the fixed length of streams or vectors.
center	Logical. If TRUE, the vector is recentered so that the original (x, y) becomes the midpoint (default is TRUE for geom_vector() and FALSE for geom_vector2()).
type	Character. Either "stream" (default) or "vector". "stream" computes a full streamline by integrating in both directions (if center = TRUE), while "vector" computes a single vector.

normalize	Logical. If TRUE, the vector endpoints are scaled to unit length before being scaled by L (default: TRUE).
tail_point	Logical. If TRUE, draws the tail point of vectors/streams (default: FALSE).
eval_point	Logical. If TRUE, marks the evaluation points used to fit gradients.
grid	A user-supplied data frame or pattern (e.g., "hex") for specifying custom evaluation points.
lineend	Line end style (round, butt, square).
linejoin	Line join style (round, mitre, bevel).
linemitre	Line mitre limit (number greater than 1).
arrow	An optional grid::arrow() specification to add arrowheads to the vectors (default: grid::arrow(angle = 25, length = unit(0.025, "npc"), type = "closed")).
geom	The geometric object used to render the streamline (only used in stat_stream(); defaults to GeomStream).

Value

A ggplot2 layer that can be added to a ggplot object.

Aesthetics

geom_gradient_smooth() supports the following aesthetics (required aesthetics are in **bold**):

- x: The x-coordinate of the data point.
- y: The y-coordinate of the data point.
- z: The scalar value used for computing the gradient.
- color: The color used for the gradient vectors. Defaults depend on the selected type.

Details

Gradient Calculation: A linear model is fitted using the provided formula and the raw data. The scalar field defined by the model is then differentiated numerically with numDeriv::grad() to yield gradient vectors.

Visualization: The resulting gradient field is visualized using GeomStream(). Since z is only used internally, it is dropped from the final visual output.

Examples

```
# Define several scalar field functions:

# Example 1: f(x, y) = x^2 - y^2
f <- function(u) {
  x <- u[1]
  y <- u[2]
  x^2 - y^2
}

# Example 2: g(x, y) = sin(x) * cos(y)
```

```
g <- function(u) {
  x <- u[1]
  y <- u[2]
  sin(x) * cos(y)
}

# Example 3: h(x, y) = log(|x| + 1) + sqrt(|y|)
h <- function(u) {
  x <- u[1]
  y <- u[2]
  log(abs(x) + 1) + sqrt(abs(y))
}

# Create a grid of evaluation points
grid_data <- expand.grid(
  x = seq(-5, 5, length.out = 30),
  y = seq(-5, 5, length.out = 30)
)

# Compute the scalar field for f and plot its gradient
grid_data$z <- apply(grid_data, 1, f)

ggplot(grid_data, aes(x = x, y = y, z = z)) +
  geom_gradient_smooth()

# Compute and plot for g:
grid_data$z <- apply(grid_data, 1, g)
ggplot(grid_data, aes(x = x, y = y, z = z)) +
  geom_gradient_smooth()

# Compute and plot for h:
grid_data$z <- apply(grid_data, 1, h)
ggplot(grid_data, aes(x = x, y = y, z = z)) +
  geom_gradient_smooth()
```

`geom_potential` *Compute and Plot Potential Function from a Conservative Vector Field*

Description

`geom_potential()` adds a raster layer to a ggplot object, visualizing the potential function derived from a conservative vector field. It computes the potential numerically over a specified grid and displays it as a heatmap.

Usage

```
geom_potential(
  mapping = NULL,
```

```
  data = NULL,
  stat = StatPotential,
  position = "identity",
  ...,
  na.rm = FALSE,
  inherit.aes = TRUE,
  show.legend = NA,
  fun,
  xlim = NULL,
  ylim = NULL,
  n = 51,
  tol = 1e-06,
  verify_conservative = FALSE
)

stat_potential(
  mapping = NULL,
  data = NULL,
  geom = GeomPotential,
  position = "identity",
  ...,
  na.rm = FALSE,
  inherit.aes = TRUE,
  show.legend = NA,
  fun,
  xlim = NULL,
  ylim = NULL,
  n = 51,
  tol = 1e-06,
  verify_conservative = FALSE
)
```

Arguments

`mapping`	A set of aesthetic mappings created by `ggplot2::aes()`. (Optional)
`data`	The data to be displayed in this layer. If `NULL`, data is inherited from the plot.
`stat`	The statistical transformation to use on the data (default: StatPotential).
`position`	Position adjustment, either as a string or the result of a position adjustment function.
`...`	Other arguments passed to `grid::layer()` and underlying methods.
`na.rm`	Logical. If `FALSE` (default), missing values are removed with a warning.
`inherit.aes`	Logical. If `FALSE`, overrides the default aesthetics rather than combining with them.
`show.legend`	Logical. Should this layer be included in the legends?
`fun`	A function that takes a numeric vector of length 2 (`c(x, y)`) and returns a numeric value, defining the conservative vector field. **(Required)**

`xlim` Numeric vector of length 2 defining the domain limits on the x-axis. Defaults to `c(-1, 1)`.

`ylim` Numeric vector of length 2 defining the domain limits on the y-axis. Defaults to `c(-1, 1)`.

`n` Integer. Number of grid points along each axis for computing the potential. Defaults to `21`.

`tol` Numeric. Tolerance for verifying if the vector field is conservative. Defaults to `1e-6`.

`verify_conservative` Logical. If `TRUE`, the function verifies that the provided vector field is conservative (i.e., that the mixed partial derivatives are equal within the specified tolerance). Defaults to `FALSE`.

`geom` The geometric object used to render the potential function. Defaults to GeomPotential.

Details

Note: the potential is only known up to a constant. The point of reference used is the lower left corner `c(xlim[1], ylim[1])`, where the potential is assumed to be 0.

Value

A ggplot2 layer that produces a potential function heatmap.

Aesthetics

`geom_potential()` accepts all aesthetics supported by `GeomRaster`. In particular, the key aesthetics include:

- **fill**: The computed potential value at each grid cell, which is mapped to a color scale.
- `x` and `y`: The coordinates of the grid cell centers. (calculated)
- `alpha`: Controls the transparency of the raster fill.

Additional raster-specific aesthetics (e.g. those controlled by `scale_fill_gradient()`, `scale_fill_viridis_c()`, etc.) can be applied to modify the appearance of the potential heatmap.

Computed Variables

The following variable is computed internally by StatPotential during the potential function calculation:

Potential The scalar potential value computed numerically at each grid point. It represents the accumulated potential from a reference point (typically the lower bounds of `xlim` and `ylim`) to the given point. This value is mapped to the `fill` aesthetic in the raster layer.

Examples

```
scalar_field <- function(u){
  x <- u[1]; y <- u[2]
  (x + y)^2 + 4*(x - y)^2 - 8*(.5)^2
}

gradient_field <- function(u) numDeriv::grad(scalar_field, u)

s <- seq(-1, 1, length.out = 51)

expand.grid("x" = s, "y" = s) |>
  transform(phi = apply(cbind(x, y), 1, scalar_field)) |>
  ggplot(aes(x, y)) + geom_raster(aes(fill = phi))

ggplot() + geom_potential(fun = gradient_field)

ggplot() + geom_potential(fun = gradient_field, verify_conservative = TRUE)
```

`geom_stream` *Create a Streamline Plot Layer in ggplot2*

Description

`geom_stream()` generates a ggplot2 layer that visualizes data as continuous streams over a temporal variable `t`. Each stream is defined by the required aesthetics `x`, `y`, and `t`, and optionally grouped by `group` (or mapped from `id`). Within each group, data points are automatically ordered by `t` to form a continuous path.

Usage

```
geom_stream(
  mapping = NULL,
  data = NULL,
  stat = StatStream,
  position = "identity",
  ...,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  arrow.fill = NULL,
  lineend = "butt",
  linejoin = "round",
  linemitre = 10,
  arrow = grid::arrow(angle = 25, length = unit(0.025, "npc"), type = "closed")
)
```

```
stat_stream(
  mapping = NULL,
  data = NULL,
  geom = GeomStream,
  position = "identity",
  ...,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  arrow.fill = NULL,
  lineend = "butt",
  linejoin = "round",
  linemitre = 10,
  arrow = grid::arrow(angle = 25, length = unit(0.025, "npc"), type = "closed")
)
```

Arguments

mapping	A set of aesthetic mappings created by `ggplot2::aes()`. **Required:** Must include `x`, `y`, and `t`; additionally, `group` is used to differentiate streams (if not provided, `id` will be mapped to `group` automatically).
data	A data frame or other object, as in `grid::layer()`.
stat	The statistical transformation to use on the data for this layer; defaults to StatStream.
position	Position adjustment, either as a string or the result of a position adjustment function.
...	Other arguments passed to the underlying layers for further customization.
na.rm	Logical. 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?
inherit.aes	Logical. If `FALSE`, overrides the default aesthetics rather than combining with them.
arrow.fill	An optional parameter specifying the color of the arrow head. Defaults to `NULL` and inherets fill/alpha of `aes()`.
lineend	Line end style (round, butt, square).
linejoin	Line join style (round, mitre, bevel).
linemitre	Line mitre limit (number greater than 1).
arrow	An optional `grid::arrow()` specification to place arrowheads on the streamline.
geom	The geometric object used to render the streamline (only used in `stat_stream()`; defaults to GeomStream).

Details

There are two variants:

- `geom_stream()`: A convenient wrapper that sets `stat = StatStream` and uses ggplot2::GeomPath by default.
- `stat_stream()`: Provides direct access to the reordering stat (i.e. `StatStream`) for advanced customization, using GeomStream for drawing.

Value

A ggplot2 layer that can be added to a plot to produce a streamline visualization.

Aesthetics

`geom_stream()` and `stat_stream()` understand the following aesthetics (required aesthetics are in **bold**):

- `x`: Horizontal position.
- `y`: Vertical position.
- `t`: Temporal or ordered variable used to sequence data points.
- `group`: Grouping variable for multiple streams (automatically mapped from `id` if absent).
- `color`: Color of the stream.
- `linetype`: Type of line used to draw the stream.
- `linewidth`: Thickness of the stream line.
- `alpha`: Transparency of the stream.

Details

- **Data Ordering**: If `t` is not provided, an error is thrown. When present, points within each group are sorted by `t` prior to drawing the stream.
- **Arrows**: The `arrow` parameter can be used to indicate direction along each stream.

Computed Variables

These are calculated by the 'stat' part of layers and can be accessed with delayed evaluation.

norm This variable is calculated as the Euclidean distance derived from the ranges of the `x` and `y` values. It serves as a normalization factor for vector lengths when the `normalize` parameter is active.

avg_spd Represents the average speed, which is defined as the length of the stream divided by the time it took to traverse that distance.

Examples

```
n <- 25
s <- seq(0, 1, length.out = n+1)[-(n+1)]
df <- data.frame( "t" = s, "x" = cos(2*pi*s), "y" = sin(2*pi*s) )

ggplot(df) +
  geom_stream(aes(x, y, t = t)) +
  coord_equal()

ggplot(df) +
  geom_stream(aes(x, y, t = t, alpha = t), size = 5) +
  coord_equal()

ggplot(df) +
  geom_path(aes(x, y, alpha = t), size = 5) +
  coord_equal()


stream_1 <- data.frame(
  x = c(0, 3),
  y = c(0, 0),
  t = 0:1
)

stream_2 <- data.frame(
  x = c(1, 1),
  y = c(1, 5),
  t = 0:1
)

stream_3 <- data.frame(
  x = c(2, 5),
  y = c(2, 6),
  t = 0:1
)

streams <- rbind(
  cbind(stream_1, id = 1),
  cbind(stream_2, id = 2),
  cbind(stream_3, id = 3)
)

ggplot(stream_1) +
  geom_stream(aes(x = x, y = y, t = t))

# set group aes if multiple vectors
ggplot(streams) +
  geom_stream(aes(x = x, y = y, t = t, group = id))
```

geom_stream_field *Create a Stream Field Layer in ggplot2*

Description

geom_stream_field() creates a ggplot2 layer that integrates a user-defined vector field function $f(x, y) \to (dx, dy)$ over a grid of seed points within a specified domain. The function numerically integrates the field starting from these seeds, producing streamlines that visualize the flow. This is useful for visualizing vector fields, flow patterns, or trajectories, such as in fluid dynamics or gradient fields.

Usage

```
geom_stream_field(
  mapping = NULL,
  data = NULL,
  stat = StatStreamField,
  position = "identity",
  ...,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = FALSE,
  fun,
  xlim = NULL,
  ylim = NULL,
  n = 11,
  args = list(),
  max_it = 1000L,
  tol = sqrt(.Machine$double.eps),
  T = NULL,
  L = NULL,
  center = TRUE,
  type = "stream",
  normalize = TRUE,
  tail_point = FALSE,
  eval_point = FALSE,
  grid = NULL,
  method = "rk4",
  lineend = "butt",
  linejoin = "round",
  linemitre = 10,
  arrow = grid::arrow(angle = 30, length = unit(0.02, "npc"), type = "closed")
)

stat_stream_field(
  mapping = NULL,
  data = NULL,
```

```
  geom = GeomStream,
  position = "identity",
  ...,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  fun,
  xlim = NULL,
  ylim = NULL,
  n = 11,
  args = list(),
  max_it = 1000,
  tol = sqrt(.Machine$double.eps),
  T = NULL,
  L = NULL,
  center = TRUE,
  type = "stream",
  normalize = TRUE,
  tail_point = FALSE,
  eval_point = FALSE,
  grid = NULL,
  method = "rk4",
  lineend = "butt",
  linejoin = "round",
  linemitre = 10,
  arrow = grid::arrow(angle = 30, length = unit(0.02, "npc"), type = "closed")
)

geom_stream_field2(
  mapping = NULL,
  data = NULL,
  stat = StatStreamField,
  position = "identity",
  ...,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = FALSE,
  fun,
  xlim = NULL,
  ylim = NULL,
  n = 11,
  args = list(),
  max_it = 1000,
  tol = sqrt(.Machine$double.eps),
  L = NULL,
  center = FALSE,
  type = "stream",
  tail_point = TRUE,
```

```
  eval_point = FALSE,
  grid = NULL,
  lineend = "butt",
  linejoin = "round",
  linemitre = 10,
  method = "rk4"
)

stat_stream_field2(
  mapping = NULL,
  data = NULL,
  geom = GeomStream,
  position = "identity",
  ...,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = FALSE,
  fun,
  xlim = NULL,
  ylim = NULL,
  n = 11,
  args = list(),
  max_it = 1000,
  tol = sqrt(.Machine$double.eps),
  L = NULL,
  center = FALSE,
  type = "stream",
  tail_point = TRUE,
  eval_point = FALSE,
  grid = NULL,
  lineend = "butt",
  linejoin = "round",
  linemitre = 10,
  method = "rk4"
)
```

Arguments

mapping	A set of aesthetic mappings created by ggplot2::aes(). (Optional)
data	A data frame or other object, as in ggplot2::layer(). (Optional)
stat	The statistical transformation to use on the data (default: StatStreamField).
position	Position adjustment, either as a string or the result of a position adjustment function.
...	Other arguments passed to ggplot2::layer() and the underlying geometry/stat.
na.rm	Logical. 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?

inherit.aes	Logical. If FALSE, overrides the default aesthetics rather than combining with them.
fun	A function of two variables, fun(x, y), returning a two-element vector (dx, dy) that defines the local flow direction at any point.
xlim	Numeric vector of length 2 specifying the domain limits in the x-direction. Defaults to $c(-1, 1)$.
ylim	Numeric vector of length 2 specifying the domain limits in the y-direction. Defaults to $c(-1, 1)$.
n	Integer or two-element numeric vector specifying the grid resolution (number of seed points) along each axis. Defaults to 11, producing an 11×11 grid.
args	A list of additional arguments passed to fun.
max_it	integer(1); Maximum number of integration steps per streamline (default: 1000L).
tol	numeric(1); a tolerance used to determine if a sink has been hit, among other things (default: sqrt(.Machine$double.eps)).
T	Numeric. When normalize = FALSE, each streamline is integrated for a fixed time T before being cropped to match the duration of the fastest streamline reaching the arc length L. When normalize = TRUE, integration instead stops when the cumulative arc length reaches L, and the parameter T is ignored.
L	Numeric. Maximum arc length for each streamline. When normalize = TRUE, integration stops once the cumulative arc length reaches L. When normalize = FALSE, streamlines are initially computed for a fixed time T and then cropped so that all are truncated to the duration it takes the fastest streamline to reach the arc length L. Defaults to NULL (a suitable default is computed from the grid spacing).
center	Logical. If TRUE (default), centers the seed points (or resulting streamlines) so that the original (x, y) becomes the midpoint.
type	Character. Either "stream" (default) or "vector". "stream" computes a full streamline by integrating in both directions (if center = TRUE), while "vector" computes a single vector.
normalize	Logical. When normalize = TRUE (the default), each streamline is integrated until its cumulative arc length reaches the specified value L, ensuring that all streams have a uniform, normalized length based on grid spacing. When normalize = FALSE, the integration runs for a fixed time (T), and afterward, all streamlines are cropped to the duration it takes for the fastest one to reach the length L, allowing for variations in arc lengths that reflect differences in flow speeds.
tail_point	Logical. If TRUE, draws a point at the tail (starting point) of each streamline. Defaults to FALSE.
eval_point	Logical. If TRUE, draws a point at the evaluation point where the field was computed. Defaults to FALSE.
grid	A data frame containing precomputed grid points for seed placement. If NULL (default), a regular Cartesian grid is generated based on xlim, ylim, and n.
method	Character. Integration method (e.g. "rk4" for Runge-Kutta 4, "euler" for Euler's method). Defaults to "rk4".

lineend	Line end style (round, butt, square).
linejoin	Line join style (round, mitre, bevel).
linemitre	Line mitre limit (number greater than 1).
arrow	A grid::arrow() specification for adding arrowheads to the streamlines. Defaults to a closed arrow with a 30° angle and length 0.02 npc.
geom	The geometric object used to render the streamlines (defaults to GeomStream).

Value

A ggplot2 layer that computes and renders streamlines over the specified domain.

Aesthetics

geom_stream_field() (and its stat variant) inherit aesthetics from GeomStream and understand the following:

- x: x-coordinate of the seed point.
- y: y-coordinate of the seed point.
- color: Color, typically used to represent computed statistics (e.g. average speed).
- linetype: Type of line used to draw the streamlines.
- linewidth: Thickness of the streamlines.
- alpha: Transparency of the streamlines.

Details

The streamlines are generated by numerically integrating the vector field defined by fun(x, y). When normalize = TRUE, integration stops once the cumulative arc length reaches L; otherwise, integration runs until time T is reached. If both T and L are provided in incompatible combinations, one parameter is ignored. The computed paths are rendered by GeomStream.

Computed Variables

The following variables are computed internally by StatStreamField during the integration of the vector field:

avg_spd For vector fields, this is computed as the total arc length divided by the integration time, providing an estimate of the average speed. It is used to scale the vector lengths when mapping length = after_stat(norm).

t The integration time at each computed point along a streamline.

d The distance between consecutive points along the computed path.

l The cumulative arc length along the streamline, calculated as the cumulative sum of d.

Examples

```
f <- function(u) c(-u[2], u[1])

# the basic usage involves providing a fun, xlim, and ylim
ggplot() + geom_stream_field(fun = f, xlim = c(-1,1), ylim = c(-1,1))

# if unspecified, xlim and ylim default to c(-1,1). we use this in what
# follows to focus on other parts of the code
ggplot() + geom_stream_field(fun = f)
ggplot() + geom_stream_field(fun = f, center = FALSE)

ggplot() + geom_stream_field(fun = f, normalize = FALSE)
ggplot() + geom_stream_field(fun = f, normalize = FALSE, center = FALSE)

# run systems until specified lengths
ggplot() + geom_stream_field(fun = f, normalize = TRUE, L = .8)
ggplot() + geom_vector_field(fun = f, normalize = TRUE, L = .3)
ggplot() + geom_vector_field(fun = f, normalize = FALSE, L = 2)

# run systems for specified times
ggplot() + geom_stream_field(fun = f, normalize = FALSE, T = .1)

# tail and eval points
ggplot() + geom_stream_field(fun = f, tail_point = TRUE)
ggplot() + geom_stream_field(fun = f, eval_point = TRUE)

# changing the grid of evaluation
ggplot() + geom_stream_field(fun = f)
ggplot() + geom_stream_field(fun = f, grid = "hex")
ggplot() + geom_stream_field(fun = f, grid = "hex", n = 5)
ggplot() + geom_stream_field(fun = f, n = 5)
ggplot() + geom_stream_field(fun = f, xlim = c(-5, 5)) + coord_equal()
ggplot() + geom_stream_field(fun = f, xlim = c(-5, 5), n = c(21, 11)) + coord_equal()
ggplot() + geom_stream_field(fun = f)
ggplot() + geom_stream_field(fun = f, grid = grid_hex(c(-1,1), c(-1,1), .2))

# using other ggplot2 tools
f <- efield_maker()

ggplot() + geom_stream_field(fun = f, xlim = c(-2,2), ylim = c(-2,2))

ggplot() +
  geom_stream_field(fun = f, xlim = c(-2,2), ylim = c(-2,2)) +
  scale_color_viridis_c(trans = "log10")

ggplot() +
  geom_stream_field(fun = f, xlim = c(-2,2), ylim = c(-2,2)) +
  scale_color_viridis_c(trans = "log10") +
  coord_equal()


# other vector fields
```

```
f <- function(u) u
ggplot() + geom_stream_field(fun = f, xlim = c(-1,1), ylim = c(-1,1))

f <- function(u) c(2,1)
ggplot() + geom_stream_field(fun = f, xlim = c(-1,1), ylim = c(-1,1))



# neat examples
f <- function(u) {
  x <- u[1]; y <- u[2]
  c(y, y*(-x^2 - 2*y^2 + 1) - x)
}
ggplot() + geom_stream_field(fun = f, xlim = c(-2,2), ylim = c(-2,2))
ggplot() + geom_stream_field(fun = f, xlim = c(-2,2), ylim = c(-2,2), type = "vector")

f <- function(u) {
  x <- u[1]; y <- u[2]
  c(y, x - x^3)
}
ggplot() + geom_stream_field(fun = f, xlim = c(-2,2), ylim = c(-2,2))
ggplot() + geom_stream_field(fun = f, xlim = c(-2,2), ylim = c(-2,2),
  grid = grid_hex(c(-2,2), c(-2,2), .35))

f <- function(u) {
  x <- u[1]; y <- u[2]
  c(x^2 - y^2, x^2 + y^2 - 2)
}
ggplot() + geom_stream_field(fun = f, xlim = c(-2,2), ylim = c(-2,2))
ggplot() + geom_stream_field(fun = f, xlim = c(-2,2), ylim = c(-2,2),
  grid = grid_hex(c(-2,2), c(-2,2), .35))

ggplot() +
  geom_stream_field(fun = f, aes(alpha = after_stat(t)), xlim = c(-2,2), ylim = c(-2,2)) +
  scale_alpha(range  = c(0,1))

ggplot() +
  geom_stream_field(
    fun = f, xlim = c(-1,1), ylim = c(-1,1),
    linewidth = .75, arrow = arrow(length = unit(0.015, "npc"))
  )
```

`geom_stream_smooth` *Create a Smoothed Vector Field Layer*

Description

`geom_stream_smooth()` creates a ggplot2 layer that visualizes a smooth vector field based on raw vector data. The function fits a multivariate linear model (by default, using the formula `cbind(fx,`

fy) ~ x * y) to predict the vector displacements at any given location. It also handles different input formats by converting polar coordinates or endpoint data to vector displacements.

Usage

```
geom_stream_smooth(
  mapping = NULL,
  data = NULL,
  stat = StatStreamField,
  position = "identity",
  ...,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  n = 11,
  xlim = NULL,
  ylim = NULL,
  normalize = TRUE,
  center = FALSE,
  type = "vector",
  formula = cbind(fx, fy) ~ x * y,
  eval_points = NULL,
  lineend = "butt",
  linejoin = "round",
  linemitre = 10,
  arrow = grid::arrow(angle = 20, length = unit(0.015, "npc"), type = "closed")
)

stat_stream_smooth(
  mapping = NULL,
  data = NULL,
  geom = GeomStream,
  position = "identity",
  ...,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  n = 11,
  xlim = NULL,
  ylim = NULL,
  normalize = TRUE,
  center = FALSE,
  type = "vector",
  formula = cbind(fx, fy) ~ x * y,
  eval_points = NULL,
  lineend = "butt",
  linejoin = "round",
  linemitre = 10,
  arrow = grid::arrow(angle = 20, length = unit(0.015, "npc"), type = "closed")
```

```
)
```

Arguments

mapping	A set of aesthetic mappings created by `ggplot2::aes()`. **Required:** Must include `x` and `y`; vector displacements are defined by `fx` and `fy`.
data	A data frame containing the raw vector data.
stat	The statistical transformation to use on the data. Defaults to `"vector_smooth"`.
position	Position adjustment, either as a string or the result of a position adjustment function.
...	Additional arguments passed to the layer. If a fixed parameter `color` is not provided, then `color = "blue"` is used.
na.rm	Logical. If `FALSE` (the default), missing values are removed with a warning.
show.legend	Logical. Should this layer be included in the legends?
inherit.aes	Logical. If `FALSE`, overrides the default aesthetics rather than combining with them.
n	An integer vector specifying the grid resolution for smoothing.
xlim	Numeric vector of length 2 specifying the domain limits in the x-direction. Defaults to $c(-1, 1)$.
ylim	Numeric vector of length 2 specifying the domain limits in the y-direction. Defaults to $c(-1, 1)$.
normalize	Logical. If `TRUE`, the vector endpoints are scaled to unit length before being scaled by `L` (default: `TRUE`).
center	Logical. If `TRUE`, the vector is recentered so that the original `(x, y)` becomes the midpoint (default is `TRUE` for `geom_vector()` and `FALSE` for `geom_vector2()`).
type	Character. Either `"stream"` (default) or `"vector"`. `"stream"` computes a full streamline by integrating in both directions (if `center = TRUE`), while `"vector"` computes a single vector.
formula	A formula specifying the multivariate linear model used for smoothing. Defaults to `cbind(fx, fy) ~ x * y`.
eval_points	A data frame of evaluation points, or `NULL`. When provided, it specifies the grid where the smoothing model is evaluated; if `NULL`, a grid is generated based on `n`.
lineend	Line end style (round, butt, square).
linejoin	Line join style (round, mitre, bevel).
linemitre	Line mitre limit (number greater than 1).
arrow	An optional `grid::arrow()` specification to add arrowheads to the vectors (default: `grid::arrow(angle = 25, length = unit(0.025, "npc"), type = "closed")`).
geom	The geometric object used to render the streamline (only used in `stat_stream()`; defaults to GeomStream).

Value

A ggplot2 layer that can be added to a ggplot object to display a smoothed vector field.

norm Computed as the Euclidean norm of the displacement, $\sqrt{fx^2 + fy^2}$, this variable is used to normalize and scale the vector lengths.

t The integration time or evaluation time at each computed point along the smoothed field (when applicable).

d The incremental distance between consecutive computed points.

l The cumulative arc length along the smoothed vector field, calculated as the cumulative sum of `d`.

Aesthetics

`geom_stream_smooth()` supports the following aesthetics (required aesthetics are in **bold**):

- `x`: The x-coordinate of the vector's starting point.
- `y`: The y-coordinate of the vector's starting point.
- `fx`: The displacement along the x-axis.
- `fy`: The displacement along the y-axis.
- `color`: The fixed color for the vector. Defaults to `"blue"`.
- `linewidth`: The thickness of the vector line.
- `linetype`: The type of the vector line (e.g., solid or dashed).
- `alpha`: The transparency level of the vector.
- `arrow`: Specifies arrowheads for the vectors.

Details

Data Conversion: If `xend`/`yend` are missing or all `NA`, the function computes them. It first checks for vector displacements (`fx` and `fy`); if present, it computes $xend = x + fx, \quad yend = y + fy$. Otherwise, it checks for polar coordinates (`angle` and `distance`) and computes $xend = x + distance \times \cos(angle \times 180/\pi), \quad yend = y + distance \times \sin(angle \times 180/\pi)$. An error is thrown if neither set is available.

Smoothing: The multivariate linear model is fitted using the provided `formula` and `data`. This model is then used to predict vector displacements at any specified grid point, generating a smooth approximation of the vector field.

Prediction Intervals: Two types of prediction intervals can be displayed:

- **Ellipse:** Depicts the joint uncertainty (covariance) in the predicted `fx` and `fy`.
- **Wedge:** Indicates the range of possible vector directions and magnitudes.

Examples

```
# Define a true vector field function
f <- function(u) {
  x <- u[1]; y <- u[2]
  c(x^2 - y^2, x^2 + y^2 - 2)
}

# Alternative example function
f <- function(u) c(-u[2], u[1])

# Visualize the vector field
ggplot() + geom_stream_field(fun = f, xlim = c(-2, 2), ylim = c(-2, 2))

# Generate design points
n <- 20
df <- data.frame(x = runif(n, -2, 2), y = runif(n, -2, 2))

# Sample function values at design points
fdf <- as.data.frame(t(apply(df, 1, f)))
colnames(fdf) <- c("fx", "fy")
df <- cbind(df, fdf)

# Visualize raw vector field data
ggplot(df) + geom_vector(aes(x, y, fx = fx, fy = fy))

# Add smoothed layer using default model
ggplot(df) +
  geom_vector(aes(x, y, fx = fx, fy = fy)) +
  geom_stream_smooth(formula = cbind(fx, fy) ~ x * y)

# Use a more complex polynomial model
ggplot(df) +
  geom_vector(aes(x, y, fx = fx, fy = fy)) +
  geom_stream_smooth(formula = cbind(fx, fy) ~ poly(x, 2) * poly(y, 2), data = df)

# Fit a linear model and use it for prediction
fhat <- function(u) {
  model <- lm(cbind(fx, fy) ~ x * y, data = df)
  predict(model, newdata = data.frame(x = u[1], y = u[2])) |> as.numeric()
}

# Visualize estimated field with the raw vector field
ggplot(df) +
  geom_stream_field(fun = fhat, normalize = FALSE, color = "#3366FF") +
  geom_vector(aes(x, y, fx = fx, fy = fy))

# Generate a hexagonal grid
hex_lattice <- grid_hex(xlim = c(-5, 5), ylim = c(-5, 5), d = 1)

# Use the hexagonal grid in geom_stream_field
ggplot(data = df) +
```

```
  geom_vector(aes(x, y, fx = fx, fy = fy), color = "black", normalize = FALSE) +
  geom_stream_smooth(eval_points = hex_lattice)

# user specified point

eval_pts <- data.frame(x = c(0, 1), y = c(2, -1))

ggplot(data = df) +
  geom_vector(aes(x, y, fx = fx, fy = fy), color = "black", normalize = FALSE) +
  geom_stream_smooth(eval_points = eval_pts)
```

---

geom_vector *Vector Layers for ggplot2*

Description

Create layers for drawing vectors on ggplot2 plots. These functions accept wide-format data with the required aesthetics x and y plus either xend and yend or one of the alternative specifications: fx and fy, or angle/ angle_deg and distance.

Usage

```
geom_vector(
  mapping = NULL,
  data = NULL,
  stat = StatVector,
  position = "identity",
  ...,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  center = TRUE,
  normalize = TRUE,
  tail_point = FALSE,
  eval_point = FALSE,
  L = NULL,
  arrow = grid::arrow(angle = 25, length = unit(0.025, "npc"), type = "closed")
)

stat_vector(
  mapping = NULL,
  data = NULL,
  geom = GeomStream,
  position = "identity",
  ...,
  na.rm = FALSE,
```

```
  show.legend = NA,
  inherit.aes = TRUE,
  center = TRUE,
  normalize = TRUE,
  tail_point = FALSE,
  eval_point = FALSE,
  L = NULL,
  arrow = grid::arrow(angle = 25, length = unit(0.025, "npc"), type = "closed")
)

geom_vector2(
  mapping = NULL,
  data = NULL,
  stat = StatVector,
  position = "identity",
  ...,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  center = FALSE,
  tail_point = TRUE,
  eval_point = FALSE,
  L = NULL,
  arrow = NULL
)

stat_vector2(
  mapping = NULL,
  data = NULL,
  geom = GeomStream,
  position = "identity",
  ...,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  center = FALSE,
  tail_point = TRUE,
  eval_point = FALSE,
  L = NULL,
  arrow = NULL
)
```

Arguments

mapping	A set of aesthetic mappings created by `ggplot2::aes()`. **Required:** Must include `x` and `y`; in addition, either `xend` and `yend` or one of the alternative specifications (`fx`/ `fy` or `angle`/ `angle_deg` and `distance`) must be provided.
data	A data frame containing the vector data in wide format.

stat	The statistical transformation to use on the data (default: StatVector).
position	Position adjustment, either as a string or the result of a position adjustment function.
...	Other arguments passed on to grid::layer().
na.rm	Logical. If FALSE (the default), missing values are removed with a warning.
show.legend	Logical. Should this layer be included in the legends?
inherit.aes	Logical. If FALSE, overrides the default aesthetics rather than combining with them.
center	Logical. If TRUE, the vector is recentered so that the original (x, y) becomes the midpoint (default is TRUE for geom_vector() and FALSE for geom_vector2()).
normalize	Logical. If TRUE, the vector endpoints are scaled to unit length before being scaled by L (default: TRUE).
tail_point	Logical. If TRUE, a point is drawn at the tail (the starting point) of each vector (default is FALSE for geom_vector() and TRUE for geom_vector2()).
eval_point	Logical. If TRUE, a point is drawn at the evaluation point corresponding to the original (untransformed) seed point before any centering or normalization (default: FALSE).
L	Numeric scalar. The desired length for the vectors in data units. If NULL (the default), a value is computed automatically based on the plot's x and y limits.
arrow	An optional grid::arrow() specification to add arrowheads to the vectors (default: grid::arrow(angle = 25, length = unit(0.025, "npc"), type = "closed")).
geom	The geometric object used to render the streamline (only used in stat_stream(); defaults to GeomStream).

Details

When specifying the vector direction using polar coordinates, you can provide either:

- angle: the vector direction in **radians**.
- angle_deg: the vector direction in **degrees** (which is automatically converted to radians).

The endpoints are computed by translating the starting point using these polar coordinates along with the supplied distance.

The data is converted to long format (two rows per vector) via StatVector and rendered with GeomStream. Optionally, arrowheads can be added to indicate direction.

There are two variants:

- geom_vector(): Uses the user-supplied aesthetic mapping.
- geom_vector2(): Uses the same underlying stat (StatVector) but adds a default mapping for length = after_stat(norm), making the computed vector norm available as an aesthetic.

Value

A ggplot2 layer that can be added to a plot.

Aesthetics

`geom_vector()` and `geom_vector2()` understand the following aesthetics (required aesthetics are in bold):

- **x**
- **y**
- xend
- yend
- fx (alternative specification)
- fy (alternative specification)
- angle (vector direction in radians; alternative specification)
- angle_deg (vector direction in degrees; alternative specification, converted to radians)
- distance (with angle/angle_deg, used to compute endpoints)
- alpha
- color
- fill
- group
- linetype
- size

Computed Variables

These are calculated by the 'stat' part of layers and can be accessed with delayed evaluation.

norm Calculated as the Euclidean distance between the starting point (`x`, `y`) and the computed endpoint (`xend`, `yend`). This value is used to normalize the vector length when the `normalize` parameter is set to `TRUE`.

Examples

```
set.seed(1234)
n <- 10

# Generate wind data in polar coordinates
data <- data.frame(
  x = rnorm(n),
  y = rnorm(n),
  dir = runif(n, -pi, pi), # angle in radians
  spd = rchisq(n, df = 2)  # speed
) |>
  transform(fx = spd * cos(dir), fy = spd * sin(dir))

# Using fx/fy to compute endpoints
ggplot(data, aes(x, y)) +
  geom_vector(aes(fx = fx, fy = fy))
```

```
# Using angle (in radians) and distance to compute endpoints
ggplot(data, aes(x, y)) +
  geom_vector(aes(angle = dir, distance = spd))

# Using angle_deg (in degrees) and distance to compute endpoints
vectors3 <- data.frame(
  x = c(0, 1, 2),
  y = c(0, 1, 2),
  angle_deg = c(0, 90, 45),
  angle = c(0, pi/2, pi/4),
  distance = c(3, 4, 5)
)
ggplot(vectors3, aes(x, y)) +
  geom_vector(aes(angle_deg = angle_deg, distance = distance))

# Basic usage with explicit start and end points:
vectors1 <- data.frame(
  x    = c(0, 1, 2),
  y    = c(0, 1, 2),
  xend = c(3, 1, 5),
  yend = c(0, 5, 6)
)
ggplot(vectors1, aes(x = x, y = y, xend = xend, yend = yend)) +
  geom_vector()

# Using center = TRUE to recenter vectors:
ggplot(vectors1, aes(x = x, y = y, xend = xend, yend = yend)) +
  geom_vector(center = TRUE)

# Using normalize = TRUE to adjust vectors to unit length:
ggplot(vectors3, aes(x = x, y = y, angle = angle, distance = distance)) +
  geom_vector(normalize = TRUE)

# Using geom_vector2, which adds a default mapping for `length`
ggplot(vectors1, aes(x = x, y = y, xend = xend, yend = yend)) +
  geom_vector2()
```

`geom_vector_field` *Vector Field Layers for ggplot2*

Description

These functions provide convenient ggplot2 layers for drawing vector fields using streamlines.

Usage

```
geom_vector_field(
  mapping = NULL,
  data = NULL,
```

```
  stat = StatStreamField,
  position = "identity",
  ...,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = FALSE,
  fun,
  xlim = NULL,
  ylim = NULL,
  n = 11,
  args = list(),
  center = TRUE,
  normalize = TRUE,
  tail_point = FALSE,
  eval_point = FALSE,
  grid = NULL,
  lineend = "butt",
  linejoin = "round",
  linemitre = 10,
  arrow = grid::arrow(angle = 30, length = unit(0.02, "npc"), type = "closed")
)

stat_vector_field(
  mapping = NULL,
  data = NULL,
  stat = StatStreamField,
  geom = GeomStream,
  position = "identity",
  ...,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = FALSE,
  fun,
  xlim = NULL,
  ylim = NULL,
  n = 11,
  args = list(),
  center = TRUE,
  normalize = TRUE,
  tail_point = FALSE,
  eval_point = FALSE,
  grid = NULL,
  lineend = "butt",
  linejoin = "round",
  linemitre = 10,
  arrow = grid::arrow(angle = 30, length = unit(0.02, "npc"), type = "closed")
)
```

```
geom_vector_field2(
  mapping = NULL,
  data = NULL,
  stat = StatStreamField,
  position = "identity",
  ...,
  na.rm = FALSE,
  show.legend = TRUE,
  inherit.aes = FALSE,
  fun,
  xlim = NULL,
  ylim = NULL,
  n = 11,
  args = list(),
  center = FALSE,
  tail_point = TRUE,
  eval_point = FALSE,
  grid = NULL,
  lineend = "butt",
  linejoin = "round",
  linemitre = 10,
  arrow = NULL
)

stat_vector_field2(
  mapping = NULL,
  data = NULL,
  geom = GeomStream,
  position = "identity",
  ...,
  na.rm = FALSE,
  show.legend = TRUE,
  inherit.aes = FALSE,
  fun,
  xlim = NULL,
  ylim = NULL,
  n = 11,
  args = list(),
  center = FALSE,
  tail_point = TRUE,
  eval_point = FALSE,
  grid = NULL,
  lineend = "butt",
  linejoin = "round",
  linemitre = 10,
  arrow = NULL
)
```

Arguments

mapping	A set of aesthetic mappings created by ggplot2::aes(). Additional aesthetics such as color, size, linetype, and alpha can be defined. In **geom_vector_field**, the default mapping includes color = after_stat(norm), whereas in **geom_vector_field2** the default mapping includes length = after_stat(norm).
data	A data frame containing the input data.
stat	The statistical transformation to use on the data for this layer. Defaults to StatStreamField.
position	Position adjustment, either as a string or the result of a call to a position adjustment function.
...	Other arguments passed on to grid::layer().
na.rm	Logical. If FALSE (the default), missing values are removed with a warning.
show.legend	Logical. Should this layer be included in the legends?
inherit.aes	Logical. If FALSE, overrides the default aesthetics rather than combining with them.
fun	A function that defines the vector field. It should take a numeric vector of length 2 (representing (x, y)) and return a numeric vector of length 2 (representing (dx, dy)). **(Required)**
xlim	Numeric vector of length two. Specifies the limits of the x-axis domain. Defaults to c(-1, 1).
ylim	Numeric vector of length two. Specifies the limits of the y-axis domain. Defaults to c(-1, 1).
n	Integer. Grid resolution specifying the number of seed points along each axis. Higher values produce a denser vector field. Defaults to 11.
args	List of additional arguments passed on to the function defined by fun.
center	Logical. If TRUE, centers the seed points or the vectors so that the original (x, y) becomes the midpoint. Defaults differ between the variants.
normalize	Logical. If TRUE, stream lengths are normalized based on grid spacing. If FALSE, a default arc length is used. (Default is TRUE; if TRUE, it is converted internally to "vector".)
tail_point	Logical. If TRUE, a point is drawn at the tail of each streamline.
eval_point	Logical. If TRUE, a point is drawn at the evaluation point, corresponding to the original (untransformed) seed point before any centering or normalization is applied.
grid	A data frame containing precomputed grid points for seed placement. If NULL (default), a regular Cartesian grid is generated based on xlim, ylim, and n.
lineend	Line end style (round, butt, square).
linejoin	Line join style (round, mitre, bevel).
linemitre	Line mitre limit (number greater than 1).
arrow	A grid::arrow() specification to add arrowheads to the streamlines. In **geom_vector_field**, the default is a closed arrow with a 30° angle and length 0.02 npc; in **geom_vector_field2** the default is NULL.
geom	The geometric object used to render the streamline (only used in stat_stream(); defaults to GeomStream).

Details

A user-defined function (`fun`) specifies the behavior of the vector field by taking a numeric vector of length 2 (representing (x, y)) and returning a numeric vector of length 2 (representing (dx, dy)). The underlying StatStreamField computes the streamlines based on the vector field function, and GeomStream renders them.

Two variants are provided:

- **geom_vector_field()** uses a default mapping that sets `color = after_stat(norm)`.
- **geom_vector_field2()** uses a default mapping that sets `length = after_stat(norm)` (with `color` unmapped by default).

Value

A ggplot2 layer that computes and plots a vector field using streamlines.

norm Calculated as the Euclidean distance between the starting point (`x`, `y`) and the computed endpoint. Used to normalize the vector.

Aesthetics

`geom_vector_field()` and `geom_vector_field2()` understand the following aesthetics (required aesthetics are in **bold**):

- `x`: The x-coordinate of the vector's starting point.
- `y`: The y-coordinate of the vector's starting point.
- `fx`: The horizontal component of the vector displacement.
- `fy`: The vertical component of the vector displacement.
- `color`: The color of the vector lines (default mapping in **geom_vector_field**).
- `length`: The computed vector norm (default mapping in **geom_vector_field2**).
- `linetype`: The type of the vector line (e.g., solid, dashed).
- `linewidth`: The thickness of the vector line.
- `alpha`: The transparency of the vector.

See Also

`geom_stream_field()`

Examples

```
f <- function(u) c(-u[2], u[1])
ggplot() + geom_vector_field(fun = f, xlim = c(-1,1), ylim = c(-1,1))

# xlim and ylim default to (-1,1), so for ease of illustration we remove them

ggplot() + geom_vector_field(fun = f)
ggplot() + geom_vector_field(fun = f, grid = "hex")
```

```
ggplot() + geom_vector_field2(fun = f)
ggplot() + geom_vector_field2(fun = f, grid = "hex")

f <- efield_maker()
ggplot() + geom_vector_field(fun = f, xlim = c(-2,2), ylim = c(-2,2))
ggplot() + geom_vector_field2(fun = f, xlim = c(-2,2), ylim = c(-2,2))
```

`ggvfields` *ggvfields: Vector Field Visualizations with ggplot2*

Description

A ggplot2 extension for visualizing vector fields in a two-dimensional space. Provides functions to create vector field layers using user-defined vector field functions.

See Also

Useful links:

- https://jamesotto852.github.io/ggdensity/
- https://github.com/dusty-turner/ggvfields/

`grid_hex` *Generate a Hexagonal Lattice*

Description

This function generates a hexagonal lattice of points within the given x and y limits, using a specified hexagon diameter. The diameter is 2 times the distance between adjacent x (and y) values, see examples.

Usage

```
grid_hex(xlim, ylim, d)
```

Arguments

`xlim`	A numeric vector of length 2 specifying the x-axis limits.
`ylim`	A numeric vector of length 2 specifying the y-axis limits.
`d`	A numeric value specifying the hexagon diameter.

Value

A data frame with two columns, `x` and `y`, containing the coordinates of the hexagonal grid points.

Examples

```
xlim <- c(-1, 1)
ylim <- c(-1, 0)

grid <- grid_hex(xlim, ylim, .25)

head( grid )
str( grid )
plot( grid, asp = 1 )

diff(sort(unique(grid$x)))
```

`scale_length_continuous`

Create a Continuous Scale for Vector Length

Description

`scale_length_continuous()` provides a continuous scale for controlling the length aesthetic in a ggplot. This is particularly useful when working with vector plots where vector lengths are mapped to a continuous scale.

Usage

```
scale_length_continuous(max_range = 0.5, ...)
```

Arguments

`max_range`	The maximum value to which the input is rescaled. Numeric scalar specifying the upper bound of the output range. Should be between 0 and 1.
`...`	Other arguments passed to `continuous_scale()`.

Value

If `max_range` is less than or equal to 0.5 (the default), a continuous scale object (typically of class `"ScaleContinuous"`) mapping the `length` aesthetic is returned. If `max_range` is greater than 0.5, a list is returned with two components:

- the continuous scale object, and
- a theme modification (a `theme` object) that adjusts the legend key width based on the value of `max_range`.

Index