Package 'ggstar'

September 19, 2025

Title Multiple Geometric Shape Point Layer for 'ggplot2'

Version 1.0.6

Description To create the multiple polygonal point layer for easily discernible shapes, we developed the package, it is like the 'geom_point' of 'ggplot2'. It can be used to draw the scatter plot.

Depends R (>= 3.5)

Imports grid, utils, ggplot2, scales, gridExtra, cli, rlang, ggiraph (>= 0.9.1)

Suggests knitr, markdown, rmarkdown, prettydoc, purrr

License Artistic-2.0

Encoding UTF-8

URL https://github.com/xiangpin/ggstar/

BugReports https://github.com/xiangpin/ggstar/issues

VignetteBuilder knitr

RoxygenNote 7.3.2

NeedsCompilation no

Author Shuangbin Xu [aut, cre] (ORCID: <https://orcid.org/0000-0003-3513-5362>)

Maintainer Shuangbin Xu <xshuangbin@163.com>

Repository CRAN

Date/Publication 2025-09-19 05:10:16 UTC

Contents

`draw_key` *Key drawing functions*

Description

Each Geom has an associated function that draws the key when the geom needs to be displayed in a legend. These are the options built into ggplot2.

Usage

```
draw_key_star(data, params, size)
```

Arguments

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

Value

A grid grob.

`GeomInteractiveStar` *ggproto classes for ggiraph*

Description

ggproto classes for ggiraph

`GeomStar` *GeomStar*

Description

GeomStar

Author(s)

Shuangbin Xu

geom_star *Star layer*

Description

geom_star provides the multiple geometric shape to create scatter plot or other point plot, it is like the 'geom_point' of 'ggplot2'. Note: the 'left-triangle' (17, 19) and 'right-triangle' (18, 20) are developed to plot the 'triangle-heatmap'. Their centers are not in their internal, but the center of hypotenuse.

Usage

```
geom_star(
  mapping = NULL,
  data = NULL,
  na.rm = FALSE,
  stat = "identity",
  position = "identity",
  show.legend = NA,
  inherit.aes = TRUE,
  ...
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
na.rm	If 'FALSE', the default, missing values are removed with a warning. If 'TRUE', missing values are silently removed.
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.

`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`.

Details

Aesthetics `geom_star` understands the following aesthetics. Required aesthetics are displayed in bold and defaults are displayed for optional aesthetics:

- `x`.
- `y`.
- `starshape` the shape of point, default is 1 (star shape).
- `starstroke` control the thickness of margin of point, default is 0.5.
- `size` the size of point, default is 1.5.
- `colour` the colour of margin, default is 'black'.
- `fill` the colour of fill, default is NA.
- `alpha` the transparency of fill, default is 1.
- `angle` control the angle of rotation of point, default is 0.
- `subset` subset the data frame which meet conditions to display.

Value

polygonal point layer

Author(s)

Shuangbin Xu

Examples

```
library(ggplot2)
p <- ggplot(iris, aes(x=Sepal.Length,
                      y=Sepal.Width,
                      starshape=Species)) +
     geom_star(size=4)
p
```

`geom_star_interactive` *Create interactive star points*

Description

The geometry is based on [geom_star()]. See the documentation for those functions for more details.

Usage

```
geom_star_interactive(...)
```

Arguments

`...`	see also the [geom_star()].

Examples

```
library(ggplot2)
library(ggiraph)
p <- ggplot(iris, aes(x=Sepal.Length,
                      y=Sepal.Width,
                      fill = Species,
                      starshape = Species,
                      tooltip = Species)
     ) +
     geom_star_interactive(size=3)
girafe(ggobj=p)
```

scale_manual *Create your own discrete scale*

Description

Create your own discrete scale

Usage

```
scale_starshape_manual(..., values, aesthetic = "starshape")

scale_angle_manual(..., values, aesthetic = "angle")
```

Arguments

`...` Arguments passed on to `ggplot2::discrete_scale`

`scale_name` **[Deprecated]** The name of the scale that should be used for error messages associated with this scale.

`palette` A palette function that when called with a single integer argument (the number of levels in the scale) returns the values that they should take (e.g., `scales::pal_hue()`).

`name` The name of the scale. Used as the axis or legend title. If `waiver()`, the default, the name of the scale is taken from the first mapping used for that aesthetic. If `NULL`, the legend title will be omitted.

`breaks` One of:

- `NULL` for no breaks
- `waiver()` for the default breaks (the scale limits)
- A character vector of breaks
- A function that takes the limits as input and returns breaks as output. Also accepts rlang lambda function notation.

`minor_breaks` One of:

- `NULL` for no minor breaks
- `waiver()` for the default breaks (none for discrete, one minor break between each major break for continuous)
- A numeric vector of positions
- A function that given the limits returns a vector of minor breaks. Also accepts rlang lambda function notation. When the function has two arguments, it will be given the limits and major break positions.

`labels` One of the options below. Please note that when `labels` is a vector, it is highly recommended to also set the `breaks` argument as a vector to protect against unintended mismatches.

- `NULL` for no labels
- `waiver()` for the default labels computed by the transformation object
- A character vector giving labels (must be same length as `breaks`)

- An expression vector (must be the same length as breaks). See ?plotmath for details.
- A function that takes the breaks as input and returns labels as output. Also accepts rlang lambda function notation.

`limits` One of:

- `NULL` to use the default scale values
- A character vector that defines possible values of the scale and their order
- A function that accepts the existing (automatic) values and returns new ones. Also accepts rlang lambda function notation.

`na.translate` Unlike continuous scales, discrete scales can easily show missing values, and do so by default. If you want to remove missing values from a discrete scale, specify `na.translate = FALSE`.

`na.value` If `na.translate = TRUE`, what aesthetic value should the missing values be displayed as? Does not apply to position scales where `NA` is always placed at the far right.

`drop` Should unused factor levels be omitted from the scale? The default, `TRUE`, uses the levels that appear in the data; `FALSE` includes the levels in the factor. Please note that to display every level in a legend, the layer should use `show.legend = TRUE`.

`guide` A function used to create a guide or its name. See `guides()` for more information.

`call` The `call` used to construct the scale for reporting messages.

`super` The super class to use for the constructed scale

`values` a set of aesthetic values to map data values to. If this is a named vector, then the values will be matched based on the names. If unnamed, values will be matched in order (usually alphabetical) with the limits of the scale. Any data values that don't match will be given 'na.value'.

`aesthetic` The names of the aesthetics that this scale works with.

Value

starshape scale constructor

`scale_starshape` *Scales for starshapes, aka glyphs*

Description

'scale_starshape' maps discrete variables to nine easily discernible shapes ('starshapes'). If you have more than 9 levels, you will get a warning message, and the seventh and subsequence levels will not appear on the plot. Use [scale_starshape_manual()] to supply your own values. You can not map a continuous variable to shape.

Usage

```
scale_starshape(..., default = TRUE)
```

Arguments

`...` Arguments passed on to `ggplot2::discrete_scale`

`aesthetics` The names of the aesthetics that this scale works with.

`palette` A palette function that when called with a single integer argument (the number of levels in the scale) returns the values that they should take (e.g., `scales::pal_hue()`).

`name` The name of the scale. Used as the axis or legend title. If `waiver()`, the default, the name of the scale is taken from the first mapping used for that aesthetic. If `NULL`, the legend title will be omitted.

`breaks` One of:

- `NULL` for no breaks
- `waiver()` for the default breaks (the scale limits)
- A character vector of breaks
- A function that takes the limits as input and returns breaks as output. Also accepts rlang lambda function notation.

`minor_breaks` One of:

- `NULL` for no minor breaks
- `waiver()` for the default breaks (none for discrete, one minor break between each major break for continuous)
- A numeric vector of positions
- A function that given the limits returns a vector of minor breaks. Also accepts rlang lambda function notation. When the function has two arguments, it will be given the limits and major break positions.

`labels` One of the options below. Please note that when `labels` is a vector, it is highly recommended to also set the `breaks` argument as a vector to protect against unintended mismatches.

- `NULL` for no labels
- `waiver()` for the default labels computed by the transformation object
- A character vector giving labels (must be same length as `breaks`)
- An expression vector (must be the same length as breaks). See ?plotmath for details.
- A function that takes the breaks as input and returns labels as output. Also accepts rlang lambda function notation.

`limits` One of:

- `NULL` to use the default scale values
- A character vector that defines possible values of the scale and their order
- A function that accepts the existing (automatic) values and returns new ones. Also accepts rlang lambda function notation.

`na.translate` Unlike continuous scales, discrete scales can easily show missing values, and do so by default. If you want to remove missing values from a discrete scale, specify `na.translate = FALSE`.

`na.value` If `na.translate = TRUE`, what aesthetic value should the missing values be displayed as? Does not apply to position scales where `NA` is always placed at the far right.

`drop` Should unused factor levels be omitted from the scale? The default, `TRUE`, uses the levels that appear in the data; `FALSE` includes the levels in the factor. Please note that to display every level in a legend, the layer should use `show.legend = TRUE`.

`guide` A function used to create a guide or its name. See `guides()` for more information.

`call` The `call` used to construct the scale for reporting messages.

`super` The super class to use for the constructed scale

`default` should the starshapes be default?

`scale_starshape_identity` *Use values without scaling for ggstar*

Description

Use values without scaling for ggstar

Usage

```
scale_starshape_identity(..., guide = "none")
```

Arguments

`...` Arguments passed on to `ggplot2::continuous_scale`

`aesthetics` The names of the aesthetics that this scale works with.

`scale_name` **[Deprecated]** The name of the scale that should be used for error messages associated with this scale.

`palette` A palette function that when called with a numeric vector with values between 0 and 1 returns the corresponding output values (e.g., `scales::pal_area()`).

`name` The name of the scale. Used as the axis or legend title. If `waiver()`, the default, the name of the scale is taken from the first mapping used for that aesthetic. If `NULL`, the legend title will be omitted.

`breaks` One of:

- `NULL` for no breaks
- `waiver()` for the default breaks computed by the transformation object
- A numeric vector of positions

- A function that takes the limits as input and returns breaks as output (e.g., a function returned by `scales::extended_breaks()`). Note that for position scales, limits are provided after scale expansion. Also accepts rlang lambda function notation.

`minor_breaks` One of:

- `NULL` for no minor breaks
- `waiver()` for the default breaks (none for discrete, one minor break between each major break for continuous)
- A numeric vector of positions
- A function that given the limits returns a vector of minor breaks. Also accepts rlang lambda function notation. When the function has two arguments, it will be given the limits and major break positions.

`n.breaks` An integer guiding the number of major breaks. The algorithm may choose a slightly different number to ensure nice break labels. Will only have an effect if `breaks = waiver()`. Use `NULL` to use the default number of breaks given by the transformation.

`labels` One of the options below. Please note that when `labels` is a vector, it is highly recommended to also set the `breaks` argument as a vector to protect against unintended mismatches.

- `NULL` for no labels
- `waiver()` for the default labels computed by the transformation object
- A character vector giving labels (must be same length as `breaks`)
- An expression vector (must be the same length as breaks). See ?plotmath for details.
- A function that takes the breaks as input and returns labels as output. Also accepts rlang lambda function notation.

`limits` One of:

- `NULL` to use the default scale range
- A numeric vector of length two providing limits of the scale. Use `NA` to refer to the existing minimum or maximum
- A function that accepts the existing (automatic) limits and returns new limits. Also accepts rlang lambda function notation. Note that setting limits on positional scales will **remove** data outside of the limits. If the purpose is to zoom, use the limit argument in the coordinate system (see `coord_cartesian()`).

`rescaler` A function used to scale the input values to the range [0, 1]. This is always `scales::rescale()`, except for diverging and n colour gradients (i.e., `scale_colour_gradient2()`, `scale_colour_gradientn()`). The `rescaler` is ignored by position scales, which always use `scales::rescale()`. Also accepts rlang lambda function notation.

`oob` One of:

- Function that handles limits outside of the scale limits (out of bounds). Also accepts rlang lambda function notation.
- The default (`scales::censor()`) replaces out of bounds values with `NA`.

- scales::squish() for squishing out of bounds values into range.
- scales::squish_infinite() for squishing infinite values into range.

expand For position scales, a vector of range expansion constants used to add some padding around the data to ensure that they are placed some distance away from the axes. Use the convenience function expansion() to generate the values for the expand argument. The defaults are to expand the scale by 5% on each side for continuous variables, and by 0.6 units on each side for discrete variables.

na.value Missing values will be replaced with this value.

transform For continuous scales, the name of a transformation object or the object itself. Built-in transformations include "asn", "atanh", "boxcox", "date", "exp", "hms", "identity", "log", "log10", "log1p", "log2", "logit", "modulus", "probability", "probit", "pseudo_log", "reciprocal", "reverse", "sqrt" and "time".

A transformation object bundles together a transform, its inverse, and methods for generating breaks and labels. Transformation objects are defined in the scales package, and are called transform_<name>. If transformations require arguments, you can call them from the scales package, e.g. scales::transform_boxcox(p = 2). You can create your own transformation with scales::new_transform().

trans **[Deprecated]** Deprecated in favour of transform.

position For position scales, The position of the axis. left or right for y axes, top or bottom for x axes.

call The call used to construct the scale for reporting messages.

super The super class to use for the constructed scale

guide Guide to use for this scale. Defaults to ‘"none"‘.

Value

identical (default) starshape scale constructor

See Also

scale_shape_identity

---

scale_starshape_interactive

Create interactive scales for ggstar shapes

Description

These scales are based on [scale_starshape], [scale_starshape_manual], [scale_starshape_discrete] see the document for those function for more details

Usage

```
scale_starshape_interactive(...)

scale_starshape_manual_interactive(...)

scale_starshape_discrete_interactive(...)
```

Arguments

...	arguments passed to base function, plus any of the [interactive_parameters].

Examples

```
library(ggplot2)
library(ggiraph)
iris$id <- seq(nrow(iris))
sps <- as.character(unique(iris$Species))
names(sps) <- sps
p <- ggplot(iris, aes(x=Sepal.Length,
                      y=Sepal.Width,
                      fill = Species,
                      starshape = Species,
                      tooltip = Species,
                      data_id = id
                     )
     ) +
     geom_star_interactive(size=2.5, alpha=.8) +
     scale_starshape_manual_interactive(
       values = c(1, 12, 15),
       tooltip = sps,
       data_id = sps,
     )
girafe(ggobj=p)
```

---

show_starshapes *Show the total star shapes*

Description

Show the total star shapes

Usage

```
show_starshapes(...)
```

Arguments

...	see also theme.

Value

gg object

Author(s)

Shuangbin Xu

Examples

```
p <- show_starshapes()
p
```

`starshape_pal` *starshape palette (discrete)*

Description

starshape palette (discrete)

Usage

```
starshape_pal(default = TRUE)
```

Arguments

`default`	should starshapes be reorder (1, 13, 15, 11, 12, 14, 29, 2, 27) or not?

Index