Package 'gglyph'

July 2, 2021

Title Multivariate Data Visualization using Glyphs

Version 0.0.0.9000

Description Provides geoms for visualizing multivariate data as glyphs using 'ggplot2'.

License GPL-2 | GPL-3

Encoding UTF-8

RoxygenNote 7.1.1

LazyData true

RdMacros Rdpack

Depends R (>= 3.5.0)

Imports dplyr, ggplot2, grid, Rdpack, rlang, scales

Suggests RColorBrewer

URL https://github.com/aravind-j/gglyph, https://aravind-j.github.io/gglyph/

BugReports https://github.com/aravind-j/gglyph/issues

Language en-GB

R topics documented:

`dotglyphGrob` *Draw a Dot Profile Glyph*

Description

Uses `Grid` graphics to draw a dot profile glyph (Chambers et al. 1983; DuToit et al. 1986).

Usage

```
dotglyphGrob(
  x = 0.5,
  y = 0.5,
  z,
  radius = 1,
  col = "black",
  fill = NA,
  lwd = 1,
  alpha = 1,
  mirror = FALSE,
  flip.axes = FALSE
)
```

Arguments

`x`	A numeric vector or unit object specifying x-locations.
`y`	A numeric vector or unit object specifying y-locations.
`z`	A numeric vector specifying the values to be plotted as dimensions of the dot glyph (number of stacked dots).
`radius`	The radius of the glyphs.
`col`	The line colour.
`fill`	The fill colour.
`lwd`	The line width.
`alpha`	The alpha transparency value.
`mirror`	logical. If `TRUE`, mirror profile is plotted.
`flip.axes`	logical. If `TRUE`, axes are flipped.

Value

A `grob` object.

References

Chambers JM, Cleveland WS, Kleiner B, Tukey PA (1983). *Graphical Methods for Data Analysis*. Chapman and Hall/CRC, Boca Raton. ISBN 978-1-351-07230-4.

DuToit SHC, Steyn AGW, Stumpf RH (1986). *Graphical Exploratory Data Analysis*, Springer Texts in Statistics. Springer-Verlag, New York. ISBN 978-1-4612-9371-2.

See Also

geom_dotglyph

Other grobs: metroglyphGrob(), pieglyphGrob(), profileglyphGrob(), starglyphGrob(), tileglyphGrob()

Examples

```
dg1 <- dotglyphGrob(x = 150, y = 300,
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2)

dg2 <- dotglyphGrob(x = 550, y = 300,
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, mirror = TRUE)

dg3 <- dotglyphGrob(x = 100, y = 550,
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, flip.axes = TRUE)

dg4 <- dotglyphGrob(x = 550, y = 550,
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, mirror = TRUE,
                    flip.axes = TRUE)

grid::grid.newpage()
grid::grid.draw(dg1)
grid::grid.draw(dg2)
grid::grid.draw(dg3)
grid::grid.draw(dg4)

dg1 <- dotglyphGrob(x = 150, y = 300,
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, fill = "black", col = "white")

dg2 <- dotglyphGrob(x = 550, y = 300,
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, mirror = TRUE,
                    fill = "salmon", col = "black")

dg3 <- dotglyphGrob(x = 100, y = 550,
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, flip.axes = TRUE,
                    fill = "cyan", col = "grey")

dg4 <- dotglyphGrob(x = 550, y = 550,
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, mirror = TRUE,
                    flip.axes = TRUE,
                    fill = "green", col = "grey")

grid::grid.newpage()
grid::grid.draw(dg1)
grid::grid.draw(dg2)
grid::grid.draw(dg3)
grid::grid.draw(dg4)
```

```
clrs <- mapply(function(a, b) rep(a, b),
               RColorBrewer::brewer.pal(6, "Dark2"),
               round(c(4, 3.5, 2.7, 6.8, 3.4, 5.7)))
clrs <- unlist(clrs)

dg1 <- dotglyphGrob(x = 150, y = 300,
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, fill = clrs, col = "white")

dg2 <- dotglyphGrob(x = 550, y = 300,
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, mirror = TRUE,
                    fill = clrs, col = "black")

dg3 <- dotglyphGrob(x = 100, y = 550,
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, flip.axes = TRUE,
                    fill = "black", col = clrs, lwd = 5)

dg4 <- dotglyphGrob(x = 550, y = 550,
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, mirror = TRUE,
                    flip.axes = TRUE,
                    col = clrs)

grid::grid.newpage()
grid::grid.draw(dg1)
grid::grid.draw(dg2)
grid::grid.draw(dg3)
grid::grid.draw(dg4)
```

`geom_dotglyph` *Add Dot Profile Glyphs as a Scatterplot*

Description

The dotglyph geom is used to plot multivariate data as dot profile glyphs (Chambers et al. 1983; DuToit et al. 1986) in a scatterplot.

Usage

```
geom_dotglyph(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
  cols = character(0L),
  radius = 1,
  fill.dot = NULL,
  fill.gradient = NULL,
  linewidth = 1,
```

```
  mirror = TRUE,
  flip.axes = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes() or aes_(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x,10)).
stat	The statistical transformation to use on the data for this layer, as a string.
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
...	Other arguments passed on to layer(). These are often aesthetics, used to set an aesthetic to a fixed value, like colour = "green" or size = 3. They may also be parameters to the paired geom/stat.
cols	Name of columns specifying the variables to be plotted in the glyphs as a character vector.
radius	The radius of the glyphs.
fill.dot	The fill colour of the stacked dots.
fill.gradient	The palette for gradient fill of the segments. See **Details** section of col_numeric() function in the scales package for available options.
linewidth	The line width of the dot glyphs.
mirror	logical. If TRUE, mirror profile is plotted.
flip.axes	logical. If TRUE, axes are flipped.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders().

Value

A geom layer.

Aesthetics

geom_pieglyph() understands the following aesthetics (required aesthetics are in bold):

- **x**
- **y**
- alpha
- colour
- fill
- group
- shape
- size
- stroke
- linetype

References

Chambers JM, Cleveland WS, Kleiner B, Tukey PA (1983). *Graphical Methods for Data Analysis*. Chapman and Hall/CRC, Boca Raton. ISBN 978-1-351-07230-4.

DuToit SHC, Steyn AGW, Stumpf RH (1986). *Graphical Exploratory Data Analysis*, Springer Texts in Statistics. Springer-Verlag, New York. ISBN 978-1-4612-9371-2.

See Also

dotglyphGrob

Other geoms: geom_metroglyph(), geom_pieglyph(), geom_profileglyph(), geom_starglyph(), geom_tileglyph()

Examples

```
# Convert data to classes
zs <- c("hp", "drat", "wt", "qsec", "vs", "am", "gear", "carb")

mtcars[ , zs] <- lapply(mtcars[, zs],
                        function(x) cut(x, breaks = 5,
                                        labels = c(1, 2, 3, 4, 5)))
mtcars[ , zs] <- lapply(mtcars[ , zs], as.factor)

mtcars$cyl <- as.factor(mtcars$cyl)
mtcars$lab <- row.names(mtcars)

library(ggplot2)
theme_set(theme_bw())
options(ggplot2.discrete.colour = RColorBrewer::brewer.pal(8, "Dark2"))
options(ggplot2.discrete.fill = RColorBrewer::brewer.pal(8, "Dark2"))

# Mapped fill
ggplot(data = mtcars) +
  geom_dotglyph(aes(x = mpg, y = disp, fill = cyl),
                cols = zs, radius = 0.5,
                alpha =  0.8) +
  ylim(c(-0, 550))
```

```
ggplot(data = mtcars) +
  geom_dotglyph(aes(x = mpg, y = disp, fill = cyl),
                cols = zs, radius = 0.5,
                mirror = FALSE,
                alpha =  0.8) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_dotglyph(aes(x = mpg, y = disp, fill = cyl),
                cols = zs, radius = 0.5,
                flip.axes = TRUE,
                alpha =  0.8) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_dotglyph(aes(x = mpg, y = disp, fill = cyl),
                cols = zs, radius = 0.5,
                mirror = FALSE, flip.axes = TRUE,
                alpha =  0.8) +
  ylim(c(-0, 550))

# Mapped colour
ggplot(data = mtcars) +
  geom_dotglyph(aes(x = mpg, y = disp, colour = cyl),
                cols = zs, radius = 0.5,
                alpha =  0.8) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_dotglyph(aes(x = mpg, y = disp, colour = cyl),
                cols = zs, radius = 0.5,
                mirror = FALSE,
                alpha =  0.8) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_dotglyph(aes(x = mpg, y = disp, colour = cyl),
                cols = zs, radius = 0.5,
                flip.axes = TRUE,
                alpha =  0.8) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_dotglyph(aes(x = mpg, y = disp, colour = cyl),
                cols = zs, radius = 0.5,
                mirror = FALSE, flip.axes = TRUE,
                alpha =  0.8) +
  ylim(c(-0, 550))

# Different fill colours
ggplot(data = mtcars) +
  geom_dotglyph(aes(x = mpg, y = disp),
                cols = zs, radius = 0.5,
                fill.dot = RColorBrewer::brewer.pal(8, "Dark2"),
                alpha =  0.8) +
  ylim(c(-0, 550))
```

```
ggplot(data = mtcars) +
  geom_dotglyph(aes(x = mpg, y = disp),
                cols = zs, radius = 0.5,
                mirror = FALSE,
                fill.dot = RColorBrewer::brewer.pal(8, "Dark2"),
                alpha =  0.8) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_dotglyph(aes(x = mpg, y = disp),
                cols = zs, radius = 0.5,
                flip.axes = TRUE,
                fill.dot = RColorBrewer::brewer.pal(8, "Dark2"),
                alpha =  0.8) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_dotglyph(aes(x = mpg, y = disp),
                cols = zs, radius = 0.5,
                mirror = FALSE, flip.axes = TRUE,
                fill.dot = RColorBrewer::brewer.pal(8, "Dark2"),
                alpha =  0.8) +
  ylim(c(-0, 550))

# Gradient fill
ggplot(data = mtcars) +
  geom_dotglyph(aes(x = mpg, y = disp),
                cols = zs, radius = 0.5,
                fill.gradient = "Greens",
                alpha =  0.8) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_dotglyph(aes(x = mpg, y = disp),
                cols = zs, radius = 0.5,
                fill.gradient = "Blues",
                mirror = FALSE,
                alpha =  0.8) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_dotglyph(aes(x = mpg, y = disp),
                cols = zs, radius = 0.5,
                flip.axes = TRUE,
                fill.gradient = "RdYlBu",
                alpha =  0.8) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_dotglyph(aes(x = mpg, y = disp),
                cols = zs, radius = 0.5,
                mirror = FALSE, flip.axes = TRUE,
                fill.gradient = "viridis",
                alpha =  0.8) +
  ylim(c(-0, 550))
```

```
# Faceted
ggplot(data = mtcars) +
  geom_dotglyph(aes(x = mpg, y = disp, fill = cyl),
                cols = zs, radius = 0.5,
                alpha =  0.8) +
  ylim(c(-0, 550)) +
  facet_grid(. ~ cyl)

ggplot(data = mtcars) +
  geom_dotglyph(aes(x = mpg, y = disp, colour = cyl),
                cols = zs, radius = 0.5,
                alpha =  0.8) +
  ylim(c(-0, 550)) +
  facet_grid(. ~ cyl)

ggplot(data = mtcars) +
  geom_dotglyph(aes(x = mpg, y = disp),
                cols = zs, radius = 0.5,
                fill.dot = RColorBrewer::brewer.pal(8, "Dark2"),
                alpha =  0.8) +
  ylim(c(-0, 550)) +
  facet_grid(. ~ cyl)

ggplot(data = mtcars) +
  geom_dotglyph(aes(x = mpg, y = disp),
                cols = zs, radius = 0.5,
                fill.gradient = "viridis",
                alpha =  0.8) +
  ylim(c(-0, 550)) +
  facet_grid(. ~ cyl)
```

`geom_metroglyph` *Add Metroglyphs as a Scatterplot*

Description

The metroglyph geom is used to plot multivariate data as metroglyphs (Anderson 1957; DuToit et al. 1986) in a scatterplot.

Usage

```
geom_metroglyph(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
  cols = character(0L),
  circle.size = 1,
  colour.circle = NULL,
  colour.ray = NULL,
  colour.points = NULL,
```

```
  linewidth.circle = 1,
  linewidth.ray = 1,
  full = TRUE,
  draw.grid = FALSE,
  point.size = 1,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes() or aes_(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x,10)).
stat	The statistical transformation to use on the data for this layer, as a string.
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
...	Other arguments passed on to layer(). These are often aesthetics, used to set an aesthetic to a fixed value, like colour = "green" or size = 3. They may also be parameters to the paired geom/stat.
cols	Name of columns specifying the variables to be plotted in the glyphs as a character vector.
circle.size	The size of the central circle.
colour.circle	The colour of circles.
colour.ray	The colour of rays.
colour.points	The colour of grid points.
linewidth.circle	The circle line width.
linewidth.ray	The ray line width.
full	logical. If TRUE, full star glyphs (360°) are plotted, otherwise half star glyphs (180°) are plotted.
draw.grid	logical. If TRUE, grid points are plotted along the whiskers if all the variables specified in cols are of type factor. Default is FALSE.
point.size	The size of the grid points in native units.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders().

Value

A geom layer.

Aesthetics

`geom_metroglyph()` understands the following aesthetics (required aesthetics are in bold):

- **x**
- **y**
- alpha
- colour
- fill
- group
- shape
- size
- stroke
- linetype

References

Anderson E (1957). "A semigraphical method for the analysis of complex problems." *Proceedings of the National Academy of Sciences of the United States of America*, **43**(10), 923.

DuToit SHC, Steyn AGW, Stumpf RH (1986). *Graphical Exploratory Data Analysis*, Springer Texts in Statistics. Springer-Verlag, New York. ISBN 978-1-4612-9371-2.

See Also

`metroglyphGrob`

Other geoms: `geom_dotglyph()`, `geom_pieglyph()`, `geom_profileglyph()`, `geom_starglyph()`, `geom_tileglyph()`

Examples

```
# Scale the data
zs <- c("hp", "drat", "wt", "qsec", "vs", "am", "gear", "carb")
mtcars[ , zs] <- lapply(mtcars[ , zs], scales::rescale)

mtcars$cyl <- as.factor(mtcars$cyl)
mtcars$lab <- row.names(mtcars)

library(ggplot2)
theme_set(theme_bw())
options(ggplot2.discrete.colour = RColorBrewer::brewer.pal(8, "Dark2"))
options(ggplot2.discrete.fill = RColorBrewer::brewer.pal(8, "Dark2"))

# Mapped colour
ggplot(data = mtcars) +
  geom_metroglyph(aes(x = mpg, y = disp, colour = cyl),
                  cols = zs, circle.size = 3, colour.ray = NULL,
                  linewidth.circle = 2, linewidth.ray = 2,
                  size = 10, alpha =  0.8, lineend = "butt") +
```

```
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_metroglyph(aes(x = mpg, y = disp, colour = cyl),
                  cols = zs, circle.size = 3, colour.ray = NULL,
                  linewidth.circle = 2, linewidth.ray = 2, fill = "gray30",
                  size = 10, alpha =  0.8, lineend = "butt") +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_metroglyph(aes(x = mpg, y = disp, colour = cyl),
                  cols = zs, circle.size = 3, colour.ray = NULL,
                  linewidth.circle = 2, linewidth.ray = 2,
                  full = FALSE,
                  size = 10, alpha =  0.8, lineend = "butt") +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_metroglyph(aes(x = mpg, y = disp, colour = cyl),
                  cols = zs, circle.size = 3, colour.ray = NULL,
                  full = FALSE,
                  linewidth.circle = 2, linewidth.ray = 2, fill = "gray30",
                  size = 10, alpha =  0.8, lineend = "butt") +
  ylim(c(-0, 550))

# Mapped colour + fill
ggplot(data = mtcars) +
  geom_metroglyph(aes(x = mpg, y = disp, colour = cyl, fill = cyl),
                  cols = zs, circle.size = 3, colour.ray = NULL,
                  linewidth.circle = 2, linewidth.ray = 2,
                  size = 10, alpha =  0.8, lineend = "butt") +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_metroglyph(aes(x = mpg, y = disp, colour = cyl, fill = cyl),
                  cols = zs, circle.size = 3, colour.ray = NULL,
                  full = FALSE,
                  linewidth.circle = 2, linewidth.ray = 2,
                  size = 10, alpha =  0.8, lineend = "butt") +
  ylim(c(-0, 550))

# Mapped fill
ggplot(data = mtcars) +
  geom_metroglyph(aes(x = mpg, y = disp, fill = cyl),
                  cols = zs, circle.size = 3, colour.ray = NULL,
                  linewidth.circle = 2, linewidth.ray = 2,
                  size = 10, alpha =  0.8, lineend = "butt") +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_metroglyph(aes(x = mpg, y = disp, fill = cyl),
                  cols = zs, circle.size = 3, colour.ray = NULL,
                  linewidth.circle = 2, linewidth.ray = 2,
                  colour.circle = "transparent",
                  size = 10, alpha =  0.8, lineend = "butt") +
  ylim(c(-0, 550))
```

```
ggplot(data = mtcars) +
  geom_metroglyph(aes(x = mpg, y = disp, fill = cyl),
                  cols = zs, circle.size = 3, colour.ray = NULL,
                  full = FALSE,
                  linewidth.circle = 2, linewidth.ray = 2,
                  size = 10, alpha =  0.8, lineend = "butt") +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_metroglyph(aes(x = mpg, y = disp, fill = cyl),
                  cols = zs, circle.size = 3, colour.ray = NULL,
                  full = FALSE, colour.circle = "transparent",
                  linewidth.circle = 2, linewidth.ray = 2,
                  size = 10, alpha =  0.8, lineend = "butt") +
  ylim(c(-0, 550))

# Rays with colours
ggplot(data = mtcars) +
  geom_metroglyph(aes(x = mpg, y = disp),
                  cols = zs, circle.size = 3,
                  linewidth.circle = 0, linewidth.ray = 2,
                  colour.circle = "transparent", fill = "gray",
                  colour.ray = RColorBrewer::brewer.pal(8, "Dark2"),
                  size = 10, alpha =  0.8) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_metroglyph(aes(x = mpg, y = disp),
                  cols = zs, circle.size = 3,
                  linewidth.circle = 0, linewidth.ray = 2,
                  colour.circle = "transparent", fill = "gray",
                  colour.ray = RColorBrewer::brewer.pal(8, "Dark2"),
                  size = 10, alpha =  0.8, full = FALSE) +
  ylim(c(-0, 550))

# Faceted
ggplot(data = mtcars) +
  geom_metroglyph(aes(x = mpg, y = disp, colour = cyl),
                  cols = zs, circle.size = 3, colour.ray = NULL,
                  linewidth.circle = 2, linewidth.ray = 2,
                  size = 10, alpha =  0.8, lineend = "butt") +
  ylim(c(-0, 550)) +
  facet_grid(. ~ cyl)

ggplot(data = mtcars) +
  geom_metroglyph(aes(x = mpg, y = disp, fill = cyl),
                  cols = zs, circle.size = 3, colour.ray = NULL,
                  linewidth.circle = 2, linewidth.ray = 2,
                  size = 10, alpha =  0.8, lineend = "butt") +
  ylim(c(-0, 550)) +
  facet_grid(. ~ cyl)


rm(mtcars)
mtcars[ , zs] <- lapply(mtcars[ , zs], scales::rescale)

mtcars[ , zs] <- lapply(mtcars[, zs],
```

```
                        function(x) cut(x, breaks = 3,
                                        labels = c(1, 2, 3)))
mtcars[ , zs] <- lapply(mtcars[ , zs], as.factor)

mtcars$cyl <- as.factor(mtcars$cyl)
mtcars$lab <- row.names(mtcars)

# Grid points
ggplot(data = mtcars) +
  geom_metroglyph(aes(x = mpg, y = disp, colour = cyl),
                  cols = zs, circle.size = 3, colour.ray = NULL,
                  linewidth.circle = 2, linewidth.ray = 2,
                  size = 2.5, alpha =  0.8, lineend = "butt",
                  draw.grid = TRUE, point.size = 5) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_metroglyph(aes(x = mpg, y = disp, fill = cyl),
                  cols = zs, circle.size = 3, colour.ray = NULL,
                  linewidth.circle = 2, linewidth.ray = 2,
                  size = 2.5, alpha =  0.8, lineend = "butt",
                  draw.grid = TRUE, point.size = 5) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_metroglyph(aes(x = mpg, y = disp),
                  cols = zs, circle.size = 3,
                  linewidth.circle = 0, linewidth.ray = 2,
                  colour.circle = "transparent", fill = "gray",
                  colour.ray = RColorBrewer::brewer.pal(8, "Dark2"),
                  size = 2.5, alpha =  0.8,
                  draw.grid = TRUE, point.size = 5) +
  ylim(c(-0, 550))
```

`geom_pieglyph` *Add Pie Glyphs as a Scatterplot*

Description

The pieglyph geom is used to plot multivariate data as pie glyphs (Ward and Lipchak 2000; Fuchs et al. 2013) in a scatterplot.

Usage

```
geom_pieglyph(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
  cols = character(0L),
  edges = 200,
  fill.segment = NULL,
```

```
  fill.gradient = NULL,
  colour.grid = NULL,
  linewidth = 1,
  linewidth.grid = linewidth,
  scale.segment = FALSE,
  scale.radius = TRUE,
  full = TRUE,
  draw.grid = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes() or aes_(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x,10)).
stat	The statistical transformation to use on the data for this layer, as a string.
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
...	Other arguments passed on to layer(). These are often aesthetics, used to set an aesthetic to a fixed value, like colour = "green" or size = 3. They may also be parameters to the paired geom/stat.
cols	Name of columns specifying the variables to be plotted in the glyphs as a character vector.
edges	The number of edges of the polygon to depict the circular glyph outline.
fill.segment	The fill colour of the segments.
fill.gradient	The palette for gradient fill of the segments. See **Details** section of col_numeric() function in the scales package for available options.
colour.grid	The colour of grid lines.
linewidth	The line width of the segments.
linewidth.grid	The line width for the grid lines.
scale.segment	logical. If TRUE, the segments (pie slices) are scaled according to value of cols.
scale.radius	logical. If TRUE, the radius of segments (pie slices) are scaled according to value of cols.
full	logical. If TRUE, full star glyphs (360°) are plotted, otherwise half star glyphs (180°) are plotted.
draw.grid	logical. If TRUE, grid points are plotted along the whiskers if all the variables specified in cols are of type factor. Default is FALSE.

show.legend — logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.

inherit.aes — If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders().

Value

A geom layer.

Aesthetics

geom_pieglyph() understands the following aesthetics (required aesthetics are in bold):

- **x**
- **y**
- alpha
- colour
- fill
- group
- shape
- size
- stroke
- linetype

References

Fuchs J, Fischer F, Mansmann F, Bertini E, Isenberg P (2013). "Evaluation of alternative glyph designs for time series data in a small multiple setting." In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 3237–3246. ISBN 978-1-4503-1899-0.

Ward MO, Lipchak BN (2000). "A visualization tool for exploratory analysis of cyclic multivariate data." *Metrika*, **51**(1), 27–37.

See Also

pieglyphGrob

Other geoms: geom_dotglyph(), geom_metroglyph(), geom_profileglyph(), geom_starglyph(), geom_tileglyph()

Examples

```
zs <- c("hp", "drat", "wt", "qsec", "vs", "am", "gear", "carb")
mtcars[ , zs] <- lapply(mtcars[ , zs], scales::rescale)

mtcars$cyl <- as.factor(mtcars$cyl)
mtcars$lab <- row.names(mtcars)

library(ggplot2)
theme_set(theme_bw())
```

```
options(ggplot2.discrete.colour = RColorBrewer::brewer.pal(8, "Dark2"))
options(ggplot2.discrete.fill = RColorBrewer::brewer.pal(8, "Dark2"))

# Mapped fill + scaled radius
ggplot(data = mtcars) +
  geom_pieglyph(aes(x = mpg, y = disp, fill = cyl),
                cols = zs, size = 10,
                alpha =  0.8) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_pieglyph(aes(x = mpg, y = disp, fill = cyl),
                cols = zs, size = 10,
                alpha =  0.8, full = FALSE) +
  ylim(c(-0, 550))

# Mapped fill + scaled segment
ggplot(data = mtcars) +
  geom_pieglyph(aes(x = mpg, y = disp, fill = cyl),
                cols = zs, size = 5,
                scale.radius = FALSE, scale.segment = TRUE,
                alpha =  0.8) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_pieglyph(aes(x = mpg, y = disp, fill = cyl),
                cols = zs, size = 5,
                scale.radius = FALSE, scale.segment = TRUE,
                alpha =  0.8, full = FALSE) +
  ylim(c(-0, 550))

# Mapped colour + scaled radius
ggplot(data = mtcars) +
  geom_pieglyph(aes(x = mpg, y = disp, colour = cyl),
                cols = zs, size = 10,
                alpha =  0.8) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_pieglyph(aes(x = mpg, y = disp, colour = cyl),
                cols = zs, size = 10, fill = "white",
                alpha =  0.8,  linewidth = 2) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_pieglyph(aes(x = mpg, y = disp, colour = cyl),
                cols = zs, size = 10,
                alpha =  0.8, full = FALSE) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_pieglyph(aes(x = mpg, y = disp, colour = cyl),
                cols = zs, size = 10, fill = "white",
                alpha =  0.8,  linewidth = 2, full = FALSE) +
  ylim(c(-0, 550))

# Mapped colour + scaled segment
```

```
ggplot(data = mtcars) +
  geom_pieglyph(aes(x = mpg, y = disp, colour = cyl),
                cols = zs, size = 5,
                scale.radius = FALSE, scale.segment = TRUE,
                alpha =  0.8) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_pieglyph(aes(x = mpg, y = disp, colour = cyl),
                cols = zs, size = 5, fill = "white",
                scale.radius = FALSE, scale.segment = TRUE,
                alpha =  0.8, linewidth = 2) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_pieglyph(aes(x = mpg, y = disp, colour = cyl),
                cols = zs, size = 5,
                scale.radius = FALSE, scale.segment = TRUE,
                alpha =  0.8, full = FALSE) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_pieglyph(aes(x = mpg, y = disp, colour = cyl),
                cols = zs, size = 5, fill = "white",
                scale.radius = FALSE, scale.segment = TRUE,
                alpha =  0.8, linewidth = 2, full = FALSE) +
  ylim(c(-0, 550))

# Segments with colours + scaled radius
ggplot(data = mtcars) +
  geom_pieglyph(aes(x = mpg, y = disp),
                cols = zs, size = 10,
                fill.segment = RColorBrewer::brewer.pal(8, "Dark2"),
                alpha =  0.8) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_pieglyph(aes(x = mpg, y = disp),
                cols = zs, size = 10,
                fill.segment = RColorBrewer::brewer.pal(8, "Dark2"),
                alpha =  0.8, full = FALSE) +
  ylim(c(-0, 550))

# Segments with colours + scaled segment (scatterpie)
ggplot(data = mtcars) +
  geom_pieglyph(aes(x = mpg, y = disp),
                cols = zs, size = 5,
                scale.radius = FALSE, scale.segment = TRUE,
                fill.segment = RColorBrewer::brewer.pal(8, "Dark2"),
                alpha =  0.8) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_pieglyph(aes(x = mpg, y = disp),
                cols = zs, size = 5,
                scale.radius = FALSE, scale.segment = TRUE,
                fill.segment = RColorBrewer::brewer.pal(8, "Dark2"),
```

```
                alpha =  0.8, full = FALSE) +
  ylim(c(-0, 550))

# Gradient fill
ggplot(data = mtcars) +
  geom_pieglyph(aes(x = mpg, y = disp),
                cols = zs, size = 5,
                scale.radius = FALSE, scale.segment = FALSE,
                fill.gradient = "Greens",
                alpha =  0.8) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_pieglyph(aes(x = mpg, y = disp),
                cols = zs, size = 5,
                scale.radius = FALSE, scale.segment = FALSE,
                fill.gradient = "Blues",
                alpha =  0.8) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_pieglyph(aes(x = mpg, y = disp),
                cols = zs, size = 5,
                scale.radius = FALSE, scale.segment = FALSE,
                fill.gradient = "RdYlBu",
                alpha =  0.8) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_pieglyph(aes(x = mpg, y = disp),
                cols = zs, size = 5,
                scale.radius = FALSE, scale.segment = FALSE,
                fill.gradient = "viridis",
                alpha =  0.8) +
  ylim(c(-0, 550))

# Faceted
ggplot(data = mtcars) +
  geom_pieglyph(aes(x = mpg, y = disp, fill = cyl),
                cols = zs, size = 10,
                alpha =  0.8) +
  ylim(c(-0, 550)) +
  facet_grid(. ~ cyl)

ggplot(data = mtcars) +
  geom_pieglyph(aes(x = mpg, y = disp, colour = cyl),
                cols = zs, size = 10,
                alpha =  0.8) +
  ylim(c(-0, 550)) +
  facet_grid(. ~ cyl)

ggplot(data = mtcars) +
  geom_pieglyph(aes(x = mpg, y = disp),
                cols = zs, size = 10,
                fill.segment = RColorBrewer::brewer.pal(8, "Dark2"),
                alpha =  0.8) +
  ylim(c(-0, 550)) +
```

```
  facet_grid(. ~ cyl)

rm(mtcars)
mtcars[ , zs] <- lapply(mtcars[ , zs], scales::rescale)

mtcars[ , zs] <- lapply(mtcars[, zs],
                        function(x) cut(x, breaks = 3,
                                        labels = c(1, 2, 3)))
mtcars[ , zs] <- lapply(mtcars[ , zs], as.factor)

mtcars$cyl <- as.factor(mtcars$cyl)
mtcars$lab <- row.names(mtcars)

# Grid lines (when scale.radius = TRUE)
ggplot(data = mtcars) +
  geom_pieglyph(aes(x = mpg, y = disp, fill = cyl),
                cols = zs, size = 2,
                alpha =  0.8, draw.grid = TRUE) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_pieglyph(aes(x = mpg, y = disp, colour = cyl),
                cols = zs, size = 2,
                alpha =  0.8, draw.grid = TRUE) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_pieglyph(aes(x = mpg, y = disp),
                cols = zs, size = 2,
                scale.radius = TRUE, scale.segment = FALSE,
                fill.gradient = "Blues",
                alpha =  0.8, draw.grid = TRUE) +
  ylim(c(-0, 550))
```

`geom_profileglyph` *Add Profile Glyphs as a Scatterplot*

Description

The profileglyph geom is used to plot multivariate data as profile glyphs (Chambers et al. 1983; DuToit et al. 1986) in a scatterplot.

Usage

```
geom_profileglyph(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
  cols = character(0L),
  width = 10,
  colour.bar = NULL,
```

```
  colour.line = NULL,
  colour.grid = NULL,
  linewidth.line = 1,
  linewidth.bar = 1,
  linewidth.grid = 1,
  fill.bar = NULL,
  fill.gradient = NULL,
  flip.axes = FALSE,
  bar = TRUE,
  line = TRUE,
  mirror = TRUE,
  draw.grid = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes() or aes_(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x,10)).
stat	The statistical transformation to use on the data for this layer, as a string.
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
...	Other arguments passed on to layer(). These are often aesthetics, used to set an aesthetic to a fixed value, like colour = "green" or size = 3. They may also be parameters to the paired geom/stat.
cols	Name of columns specifying the variables to be plotted in the glyphs as a character vector.
width	The width of the bars.
colour.bar	The colour of bars.
colour.line	The colour of profile line(s).
colour.grid	The colour of the grid lines.
linewidth.line	The line width of the profile line(s)
linewidth.bar	The line width of the bars.
linewidth.grid	The line width of the grid lines.
fill.bar	The fill colour of the bars.
fill.gradient	The palette for gradient fill of the segments. See **Details** section of col_numeric() function in the scales package for available options.

flip.axes	logical. If TRUE, axes are flipped.
bar	logical. If TRUE, profile bars are plotted.
line	logical. If TRUE, profile line is plotted.
mirror	logical. If TRUE, mirror profile is plotted.
draw.grid	logical. If TRUE, grid points are plotted along the whiskers if all the variables specified in cols are of type factor. Default is FALSE.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders().

Value

A geom layer.

Aesthetics

geom_pieglyph() understands the following aesthetics (required aesthetics are in bold):

- **x**
- **y**
- alpha
- colour
- fill
- group
- shape
- size
- stroke
- linetype

References

Chambers JM, Cleveland WS, Kleiner B, Tukey PA (1983). *Graphical Methods for Data Analysis*. Chapman and Hall/CRC, Boca Raton. ISBN 978-1-351-07230-4.

DuToit SHC, Steyn AGW, Stumpf RH (1986). *Graphical Exploratory Data Analysis*, Springer Texts in Statistics. Springer-Verlag, New York. ISBN 978-1-4612-9371-2.

See Also

profileglyphGrob

Other geoms: geom_dotglyph(), geom_metroglyph(), geom_pieglyph(), geom_starglyph(), geom_tileglyph()

Examples

```
# Scale the data
zs <- c("hp", "drat", "wt", "qsec", "vs", "am", "gear", "carb")
mtcars[ , zs] <- lapply(mtcars[ , zs], scales::rescale)

mtcars$cyl <- as.factor(mtcars$cyl)
mtcars$lab <- row.names(mtcars)

library(ggplot2)
theme_set(theme_bw())
options(ggplot2.discrete.colour = RColorBrewer::brewer.pal(8, "Dark2"))
options(ggplot2.discrete.fill = RColorBrewer::brewer.pal(8, "Dark2"))

ggplot(data = mtcars) +
  geom_profileglyph(aes(x = mpg, y = disp, fill = cyl),
                    cols = zs, size = 5, width = 1,
                    alpha =  0.8) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_profileglyph(aes(x = mpg, y = disp, fill = cyl),
                    cols = zs, size = 5, width = 1,
                    mirror = FALSE,
                    alpha =  0.8) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_profileglyph(aes(x = mpg, y = disp, fill = cyl),
                    cols = zs, size = 5, width = 1,
                    flip.axes = TRUE,
                    alpha =  0.8) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_profileglyph(aes(x = mpg, y = disp, fill = cyl),
                    cols = zs, size = 5, width = 1,
                    mirror = FALSE, flip.axes = TRUE,
                    alpha =  0.8) +
  ylim(c(-0, 550))

# Mapped fill + line
ggplot(data = mtcars) +
  geom_profileglyph(aes(x = mpg, y = disp, fill = cyl),
                    cols = zs, size = 5, width = 1,
                    line = FALSE,
                    alpha =  0.8) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_profileglyph(aes(x = mpg, y = disp, fill = cyl),
                    cols = zs, size = 5, width = 1,
                    line = FALSE, mirror = FALSE,
                    alpha =  0.8) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
```

```
  geom_profileglyph(aes(x = mpg, y = disp, fill = cyl),
                    cols = zs, size = 5, width = 1,
                    line = FALSE, flip.axes = TRUE,
                    alpha =  0.8) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_profileglyph(aes(x = mpg, y = disp, fill = cyl),
                    cols = zs, size = 5, width = 1,
                    line = FALSE, mirror = FALSE, flip.axes = TRUE,
                    alpha =  0.8) +
  ylim(c(-0, 550))

# Mapped fill + bar
ggplot(data = mtcars) +
  geom_profileglyph(aes(x = mpg, y = disp, fill = cyl),
                    cols = zs, size = 5, width = 1,
                    bar = FALSE,
                    alpha =  0.8) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_profileglyph(aes(x = mpg, y = disp, fill = cyl),
                    cols = zs, size = 5, width = 1,
                    bar = FALSE, mirror = FALSE,
                    alpha =  0.8) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_profileglyph(aes(x = mpg, y = disp, fill = cyl),
                    cols = zs, size = 5, width = 1,
                    bar = FALSE, flip.axes = TRUE,
                    alpha =  0.8) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_profileglyph(aes(x = mpg, y = disp, fill = cyl),
                    cols = zs, size = 5, width = 1,
                    bar = FALSE, mirror = FALSE, flip.axes = TRUE,
                    alpha =  0.8) +
  ylim(c(-0, 550))

# Mapped colour + bar and line
ggplot(data = mtcars) +
  geom_profileglyph(aes(x = mpg, y = disp, colour = cyl),
                    cols = zs, size = 5, width = 1,
                    alpha =  0.8) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_profileglyph(aes(x = mpg, y = disp, colour = cyl),
                    cols = zs, size = 5, width = 1,
                    mirror = FALSE,
                    alpha =  0.8) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
```

```
  geom_profileglyph(aes(x = mpg, y = disp, colour = cyl),
                    cols = zs, size = 5, width = 1,
                    flip.axes = TRUE,
                    alpha =  0.8) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_profileglyph(aes(x = mpg, y = disp, colour = cyl),
                    cols = zs, size = 5, width = 1,
                    mirror = FALSE, flip.axes = TRUE,
                    alpha =  0.8) +
  ylim(c(-0, 550))

# Mapped colour + line
ggplot(data = mtcars) +
  geom_profileglyph(aes(x = mpg, y = disp, colour = cyl),
                    cols = zs, size = 5, width = 1,
                    line = FALSE,
                    alpha =  0.8) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_profileglyph(aes(x = mpg, y = disp, colour = cyl),
                    cols = zs, size = 5, width = 1,
                    line = FALSE, mirror = FALSE,
                    alpha =  0.8) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_profileglyph(aes(x = mpg, y = disp, colour = cyl),
                    cols = zs, size = 5, width = 1,
                    line = FALSE, flip.axes = TRUE,
                    alpha =  0.8) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_profileglyph(aes(x = mpg, y = disp, colour = cyl),
                    cols = zs, size = 5, width = 1,
                    line = FALSE, mirror = FALSE, flip.axes = TRUE,
                    alpha =  0.8) +
  ylim(c(-0, 550))

# Mapped colour + bar
ggplot(data = mtcars) +
  geom_profileglyph(aes(x = mpg, y = disp, colour = cyl),
                    cols = zs, size = 5, width = 1,
                    bar = FALSE,
                    alpha =  0.8) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_profileglyph(aes(x = mpg, y = disp, colour = cyl),
                    cols = zs, size = 5, width = 1,
                    bar = FALSE, mirror = FALSE,
                    alpha =  0.8) +
  ylim(c(-0, 550))
```

```
ggplot(data = mtcars) +
  geom_profileglyph(aes(x = mpg, y = disp, colour = cyl),
                    cols = zs, size = 5, width = 1,
                    bar = FALSE, flip.axes = TRUE,
                    alpha =  0.8) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_profileglyph(aes(x = mpg, y = disp, colour = cyl),
                    cols = zs, size = 5, width = 1,
                    bar = FALSE, mirror = FALSE, flip.axes = TRUE,
                    alpha =  0.8) +
  ylim(c(-0, 550))

# Bars with different fill + bar and line
ggplot(data = mtcars) +
  geom_profileglyph(aes(x = mpg, y = disp),
                    cols = zs, size = 5, width = 1,
                    fill.bar = RColorBrewer::brewer.pal(8, "Dark2"),
                    alpha =  0.8) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_profileglyph(aes(x = mpg, y = disp),
                    cols = zs, size = 5, width = 1,
                    fill.bar = RColorBrewer::brewer.pal(8, "Dark2"),
                    mirror = FALSE,
                    alpha =  0.8) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_profileglyph(aes(x = mpg, y = disp),
                    cols = zs, size = 5, width = 1,
                    fill.bar = RColorBrewer::brewer.pal(8, "Dark2"),
                    flip.axes = TRUE,
                    alpha =  0.8) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_profileglyph(aes(x = mpg, y = disp),
                    cols = zs, size = 5, width = 1,
                    fill.bar = RColorBrewer::brewer.pal(8, "Dark2"),
                    mirror = FALSE, flip.axes = TRUE,
                    alpha =  0.8) +
  ylim(c(-0, 550))

# Bars with different fill + bar
ggplot(data = mtcars) +
  geom_profileglyph(aes(x = mpg, y = disp),
                    cols = zs, size = 5, width = 1,
                    line = FALSE,
                    fill.bar = RColorBrewer::brewer.pal(8, "Dark2"),
                    alpha =  0.8) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_profileglyph(aes(x = mpg, y = disp),
```

```
                    cols = zs, size = 5, width = 1,
                    line = FALSE,
                    fill.bar = RColorBrewer::brewer.pal(8, "Dark2"),
                    mirror = FALSE,
                    alpha =  0.8) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_profileglyph(aes(x = mpg, y = disp),
                    cols = zs, size = 5, width = 1,
                    line = FALSE,
                    fill.bar = RColorBrewer::brewer.pal(8, "Dark2"),
                    flip.axes = TRUE,
                    alpha =  0.8) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_profileglyph(aes(x = mpg, y = disp),
                    cols = zs, size = 5, width = 1,
                    line = FALSE,
                    fill.bar = RColorBrewer::brewer.pal(8, "Dark2"),
                    mirror = FALSE, flip.axes = TRUE,
                    alpha =  0.8) +
  ylim(c(-0, 550))

# Bars with gradient fill + bar and line
ggplot(data = mtcars) +
  geom_profileglyph(aes(x = mpg, y = disp),
                    cols = zs, size = 5, width = 1,
                    fill.gradient = "Greens",
                    alpha =  0.8) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_profileglyph(aes(x = mpg, y = disp),
                    cols = zs, size = 5, width = 1,
                    fill.gradient = "Blues",
                    mirror = FALSE,
                    alpha =  0.8) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_profileglyph(aes(x = mpg, y = disp),
                    cols = zs, size = 5, width = 1,
                    fill.gradient = "RdYlBu",
                    flip.axes = TRUE,
                    alpha =  0.8) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_profileglyph(aes(x = mpg, y = disp),
                    cols = zs, size = 5, width = 1,
                    fill.gradient = "viridis",
                    mirror = FALSE, flip.axes = TRUE,
                    alpha =  0.8) +
  ylim(c(-0, 550))
```

```
# Bars with gradient fill + bar
ggplot(data = mtcars) +
  geom_profileglyph(aes(x = mpg, y = disp),
                    cols = zs, size = 5, width = 1,
                    line = FALSE,
                    fill.gradient = "Greens",
                    alpha =  0.8) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_profileglyph(aes(x = mpg, y = disp),
                    cols = zs, size = 5, width = 1,
                    line = FALSE,
                    fill.gradient = "Blues",
                    mirror = FALSE,
                    alpha =  0.8) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_profileglyph(aes(x = mpg, y = disp),
                    cols = zs, size = 5, width = 1,
                    line = FALSE,
                    fill.gradient = "RdYlBu",
                    flip.axes = TRUE,
                    alpha =  0.8) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_profileglyph(aes(x = mpg, y = disp),
                    cols = zs, size = 5, width = 1,
                    line = FALSE,
                    fill.gradient = "viridis",
                    mirror = FALSE, flip.axes = TRUE,
                    alpha =  0.8) +
  ylim(c(-0, 550))

# Faceted
ggplot(data = mtcars) +
  geom_profileglyph(aes(x = mpg, y = disp, fill = cyl),
                    cols = zs, size = 5, width = 1,
                    alpha =  0.8) +
  ylim(c(-0, 550)) +
  facet_grid(. ~ cyl)

ggplot(data = mtcars) +
  geom_profileglyph(aes(x = mpg, y = disp, colour = cyl),
                    cols = zs, size = 5, width = 1,
                    alpha =  0.8) +
  ylim(c(-0, 550)) +
  facet_grid(. ~ cyl)

ggplot(data = mtcars) +
  geom_profileglyph(aes(x = mpg, y = disp),
                    cols = zs, size = 5, width = 1,
                    fill.bar = RColorBrewer::brewer.pal(8, "Dark2"),
                    alpha =  0.8) +
  ylim(c(-0, 550)) +
```

```
  facet_grid(. ~ cyl)

rm(mtcars)
mtcars[ , zs] <- lapply(mtcars[ , zs], scales::rescale)

mtcars[ , zs] <- lapply(mtcars[, zs],
                        function(x) cut(x, breaks = 3,
                                        labels = c(1, 2, 3)))
mtcars[ , zs] <- lapply(mtcars[ , zs], as.factor)

mtcars$cyl <- as.factor(mtcars$cyl)
mtcars$lab <- row.names(mtcars)

# Grid lines (when bar = TRUE)
ggplot(data = mtcars) +
  geom_profileglyph(aes(x = mpg, y = disp, fill = cyl),
                    cols = zs, size = 3, width = 1,
                    alpha =  0.8, draw.grid = TRUE) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_profileglyph(aes(x = mpg, y = disp, col = cyl),
                    cols = zs, size = 3, width = 1,
                    alpha =  0.8, draw.grid = TRUE) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_profileglyph(aes(x = mpg, y = disp, fill = cyl),
                    cols = zs, size = 3, width = 1,
                    fill.gradient = "Blues",
                    alpha =  0.8, draw.grid = TRUE) +
  ylim(c(-0, 550))
```

`geom_starglyph` *Add Star Glyphs as a Scatterplot*

Description

The starglyph geom is used to plot multivariate data as star glyphs (Siegel et al. 1972; Chambers et al. 1983; DuToit et al. 1986) in a scatterplot.

Usage

```
geom_starglyph(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
  cols = character(0L),
  whisker = TRUE,
  contour = TRUE,
  colour.whisker = NULL,
```

```
  colour.contour = NULL,
  colour.points = NULL,
  linewidth.whisker = 1,
  linewidth.contour = 1,
  full = TRUE,
  draw.grid = FALSE,
  point.size = 1,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes() or aes_(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x,10)).
stat	The statistical transformation to use on the data for this layer, as a string.
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
...	Other arguments passed on to layer(). These are often aesthetics, used to set an aesthetic to a fixed value, like colour = "green" or size = 3. They may also be parameters to the paired geom/stat.
cols	Name of columns specifying the variables to be plotted in the glyphs as a character vector.
whisker	logical. If TRUE, plots the star glyph whiskers.
contour	logical. If TRUE, plots the star glyph contours.
colour.whisker	The colour of whiskers.
colour.contour	The colour of contours.
colour.points	The colour of grid points.
linewidth.whisker	The whisker line width.
linewidth.contour	The contour line width.
full	logical. If TRUE, full star glyphs (360°) are plotted, otherwise half star glyphs (180°) are plotted.
draw.grid	logical. If TRUE, grid points are plotted along the whiskers if all the variables specified in cols are of type factor. Default is FALSE.
point.size	The size of the grid points in native units.

show.legend — logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.

inherit.aes — If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders().

Value

A geom layer.

Aesthetics

geom_starglyph() understands the following aesthetics (required aesthetics are in bold):

- **x**
- **y**
- alpha
- colour
- fill
- group
- shape
- size
- stroke
- linetype

References

Chambers JM, Cleveland WS, Kleiner B, Tukey PA (1983). *Graphical Methods for Data Analysis*. Chapman and Hall/CRC, Boca Raton. ISBN 978-1-351-07230-4.

DuToit SHC, Steyn AGW, Stumpf RH (1986). *Graphical Exploratory Data Analysis*, Springer Texts in Statistics. Springer-Verlag, New York. ISBN 978-1-4612-9371-2.

Siegel JH, Farrell EJ, Goldwyn RM, Friedman HP (1972). "The surgical implications of physiologic patterns in myocardial infarction shock." *Surgery*, **72**(1), 126–141.

See Also

starglyphGrob

Other geoms: geom_dotglyph(), geom_metroglyph(), geom_pieglyph(), geom_profileglyph(), geom_tileglyph()

Examples

```
# Scale the data
zs <- c("hp", "drat", "wt", "qsec", "vs", "am", "gear", "carb")
mtcars[ , zs] <- lapply(mtcars[ , zs], scales::rescale)

mtcars$cyl <- as.factor(mtcars$cyl)
mtcars$lab <- row.names(mtcars)
```

```
library(ggplot2)
theme_set(theme_bw())
options(ggplot2.discrete.colour = RColorBrewer::brewer.pal(8, "Dark2"))
options(ggplot2.discrete.fill = RColorBrewer::brewer.pal(8, "Dark2"))

# Both whiskers and contour
ggplot(data = mtcars) +
  geom_starglyph(aes(x = mpg, y = disp, fill = cyl),
                 cols = zs, whisker = TRUE, contour = TRUE,
                 size = 10, alpha =  0.5) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_starglyph(aes(x = mpg, y = disp, fill = cyl),
                 cols = zs, whisker = TRUE, contour = TRUE,
                 size = 10, alpha =  0.5, full = FALSE) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_starglyph(aes(x = mpg, y = disp, fill = cyl),
                 cols = zs, whisker = TRUE, contour = TRUE,
                 size = 10, alpha =  0.5,
                 linewidth.whisker = 3, linewidth.contour = 0.1) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_starglyph(aes(x = mpg, y = disp, fill = cyl),
                 cols = zs, whisker = TRUE, contour = TRUE,
                 size = 10, alpha =  0.5,
                 linewidth.whisker = 1, linewidth.contour = 3) +
  ylim(c(-0, 550))


# Only contours (polygon)
ggplot(data = mtcars) +
  geom_starglyph(aes(x = mpg, y = disp, fill = cyl),
                 cols = zs, whisker = FALSE, contour = TRUE,
                 size = 10, alpha =  0.5) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_starglyph(aes(x = mpg, y = disp, fill = cyl),
                 cols = zs, whisker = FALSE, contour = TRUE,
                 size = 10, alpha =  0.5, linewidth.contour = 3) +
  ylim(c(-0, 550))

# Only whiskers
ggplot(data = mtcars) +
  geom_starglyph(aes(x = mpg, y = disp, colour = cyl),
                 cols = zs, whisker = TRUE, contour = FALSE,
                 size = 10) +
  geom_point(data = mtcars, aes(x = mpg, y = disp, colour = cyl)) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_starglyph(aes(x = mpg, y = disp, colour = cyl),
```

```
                cols = zs, whisker = TRUE, contour = FALSE,
                size = 10, full = FALSE) +
  geom_point(data = mtcars, aes(x = mpg, y = disp, colour = cyl)) +
  ylim(c(-0, 550))

# Whiskers with colours
ggplot(data = mtcars) +
  geom_starglyph(aes(x = mpg, y = disp),
                cols = zs, whisker = TRUE, contour = FALSE,
                size = 10,
                colour.whisker = RColorBrewer::brewer.pal(8, "Dark2")) +
  geom_point(data = mtcars, aes(x = mpg, y = disp)) +
  ylim(c(-0, 550))

# With text annotations
ggplot(data = mtcars) +
  geom_starglyph(aes(x = mpg, y = disp, colour = cyl),
                cols = zs, whisker = TRUE, contour = FALSE,
                size = 10) +
  geom_point(data = mtcars, aes(x = mpg, y = disp, colour = cyl)) +
  geom_text(data = mtcars, aes(x = mpg, y = disp, label = lab), cex = 2) +
  ylim(c(-0, 550))

# Faceted
ggplot(data = mtcars) +
  geom_starglyph(aes(x = mpg, y = disp, fill = cyl),
                cols = zs, whisker = TRUE, contour = TRUE,
                size = 10, alpha =  0.5) +
  ylim(c(-0, 550)) +
  facet_grid(. ~ cyl)

ggplot(data = mtcars) +
  geom_starglyph(aes(x = mpg, y = disp, colour = cyl),
                cols = zs, whisker = TRUE, contour = TRUE,
                size = 10) +
  ylim(c(-0, 550)) +
  facet_grid(. ~ cyl)

rm(mtcars)
mtcars[ , zs] <- lapply(mtcars[ , zs], scales::rescale)

mtcars[ , zs] <- lapply(mtcars[, zs],
                        function(x) cut(x, breaks = 3,
                                        labels = c(1, 2, 3)))
mtcars[ , zs] <- lapply(mtcars[ , zs], as.factor)

mtcars$cyl <- as.factor(mtcars$cyl)
mtcars$lab <- row.names(mtcars)

# Grid points
ggplot(data = mtcars) +
  geom_starglyph(aes(x = mpg, y = disp, fill = cyl),
                cols = zs, whisker = TRUE, contour = TRUE,
                size = 3, alpha =  0.5, draw.grid = TRUE,
                point.size = 5) +
  ylim(c(-0, 550))
```

```
ggplot(data = mtcars) +
  geom_starglyph(aes(x = mpg, y = disp, colour = cyl),
                 cols = zs, whisker = TRUE, contour = FALSE,
                 size = 3, draw.grid = TRUE, point.size = 7,
                 linewidth.whisker = 2, alpha = 0.7) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_starglyph(aes(x = mpg, y = disp),
                 cols = zs, whisker = TRUE, contour = FALSE,
                 size = 3, draw.grid = TRUE,
                 point.size = 5, alpha =  0.8,
                 colour.whisker = RColorBrewer::brewer.pal(8, "Dark2")) +
  geom_point(data = mtcars, aes(x = mpg, y = disp)) +
  ylim(c(-0, 550))
```

`geom_tileglyph` *Add Tile Glyphs as a Scatterplot*

Description

The tileglyph geom is used to plot multivariate data as tile glyphs similar to 'autoglyph' (Beddow 1990) or 'stripe glyph' (Fuchs et al. 2013) in a scatterplot.

Usage

```
geom_tileglyph(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
  cols = character(0L),
  ratio = 1,
  nrow = 1,
  linewidth = 1,
  fill.gradient = NULL,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

`mapping`	Set of aesthetic mappings created by `aes()` or `aes_()`. If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.
`data`	The data to be displayed in this layer. There are three options: If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`.

	A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x,10)).
stat	The statistical transformation to use on the data for this layer, as a string.
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
...	Other arguments passed on to layer(). These are often aesthetics, used to set an aesthetic to a fixed value, like colour = "green" or size = 3. They may also be parameters to the paired geom/stat.
cols	Name of columns specifying the variables to be plotted in the glyphs as a character vector.
ratio	The aspect ratio (height / width).
nrow	The number of rows.
linewidth	The line width of the tile glyphs.
fill.gradient	The palette for gradient fill of the segments. See **Details** section of col_numeric() function in the scales package for available options.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders().

Value

A geom layer.

Aesthetics

geom_pieglyph() understands the following aesthetics (required aesthetics are in bold):

- **x**
- **y**
- alpha
- colour
- fill
- group
- shape
- size
- stroke
- linetype

References

Beddow J (1990). "Shape coding of multidimensional data on a microcomputer display." In *Proceedings of the First IEEE Conference on Visualization: Visualization '90*, 238–246. ISBN 978-0-8186-2083-6.

Fuchs J, Fischer F, Mansmann F, Bertini E, Isenberg P (2013). "Evaluation of alternative glyph designs for time series data in a small multiple setting." In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 3237–3246. ISBN 978-1-4503-1899-0.

See Also

tileglyphGrob

Other geoms: geom_dotglyph(), geom_metroglyph(), geom_pieglyph(), geom_profileglyph(), geom_starglyph()

Examples

```
# Scale the data
zs <- c("hp", "drat", "wt", "qsec", "vs", "am", "gear", "carb")
mtcars[ , zs] <- lapply(mtcars[ , zs], scales::rescale)

mtcars$cyl <- as.factor(mtcars$cyl)
mtcars$lab <- row.names(mtcars)

library(ggplot2)
theme_set(theme_bw())
options(ggplot2.discrete.colour = RColorBrewer::brewer.pal(8, "Dark2"))
options(ggplot2.discrete.fill = RColorBrewer::brewer.pal(8, "Dark2"))

ggplot(data = mtcars) +
  geom_tileglyph(aes(x = mpg, y = disp),
                 cols = zs, size = 2,
                 fill.gradient = "Blues",
                 alpha =  0.5) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_tileglyph(aes(x = mpg, y = disp),
                 cols = zs, size = 2,
                 nrow = 2,
                 fill.gradient = "Greens",
                 alpha =  0.5) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_tileglyph(aes(x = mpg, y = disp),
                 cols = zs, size = 1,
                 ratio = 4,
                 fill.gradient = "RdYlBu",
                 alpha =  0.5) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_tileglyph(aes(x = mpg, y = disp),
                 cols = zs, size = 1,
```

```
                ratio = 4, nrow = 2,
                fill.gradient = "viridis",
                alpha =  0.5) +
  ylim(c(-0, 550))
```

`metroglyphGrob` *Draw a Metroglyph*

Description

Uses `Grid` graphics to draw a metroglyph (Anderson 1957; DuToit et al. 1986).

Usage

```
metroglyphGrob(
  x = 0.5,
  y = 0.5,
  z,
  size = 1,
  circle.size = 5,
  col.circle = "black",
  col.ray = "black",
  col.points = "black",
  fill = NA,
  lwd.circle = 1,
  lwd.ray = 1,
  alpha = 1,
  angle.start = 0,
  angle.stop = 2 * base::pi,
  lineend = c("round", "butt", "square"),
  grid.levels = NULL,
  draw.grid = FALSE,
  point.size = 10
)
```

Arguments

`x`	A numeric vector or unit object specifying x-locations.
`y`	A numeric vector or unit object specifying y-locations.
`z`	A numeric vector specifying the length of rays.
`size`	The size of rays.
`circle.size`	The size of the central circle.
`col.circle`	The circle colour.
`col.ray`	The colour of rays.
`col.points`	The colour of grid points.
`fill`	The circle fill colour.
`lwd.circle`	The circle line width.

lwd.ray	The ray line width.
alpha	The alpha transparency value.
angle.start	The start angle for the glyph rays in radians. Default is zero.
angle.stop	The stop angle for the glyph rays in radians. Default is 2π.
lineend	The line end style for the rays. Either "round", "butt" or "square".
grid.levels	A list of grid levels (as vectors) corresponding to the values in z at which points are to be plotted. The values in z should be present in the list specified.
draw.grid	logical. If TRUE, grid points are plotted along the whiskers. Default is FALSE.
point.size	The size of the grid points in native units.

Value

A gTree object.

References

Anderson E (1957). "A semigraphical method for the analysis of complex problems." *Proceedings of the National Academy of Sciences of the United States of America*, **43**(10), 923.

DuToit SHC, Steyn AGW, Stumpf RH (1986). *Graphical Exploratory Data Analysis*, Springer Texts in Statistics. Springer-Verlag, New York. ISBN 978-1-4612-9371-2.

See Also

geom_metroglyph

Other grobs: dotglyphGrob(), pieglyphGrob(), profileglyphGrob(), starglyphGrob(), tileglyphGrob()

Examples

```
mglyph1 <- metroglyphGrob(x = 300, y = 200,
                          z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                          size = 25, circle.size = 2)

mglyph2 <- metroglyphGrob(x = 800, y = 200,
                          z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                          size = 25, circle.size = 5)

mglyph3 <- metroglyphGrob(x = 300, y = 600,
                          z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                          size = 25, circle.size = 0,
                          angle.start = base::pi, angle.stop = -base::pi)

mglyph4 <- metroglyphGrob(x = 800, y = 600,
                          z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                          size = 25, circle.size = 10,
                          angle.start = base::pi, angle.stop = -base::pi)

grid::grid.newpage()
grid::grid.draw(mglyph1)
grid::grid.draw(mglyph2)
grid::grid.draw(mglyph3)
grid::grid.draw(mglyph4)
```

```
mglyph1 <- metroglyphGrob(x = 200, y = 100,
                          z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                          size = 25, circle.size = 2,
                          angle.start = -base::pi, angle.stop = 0)

mglyph2 <- metroglyphGrob(x = 800, y = 100,
                          z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                          size = 25, circle.size = 5,
                          angle.start = -base::pi, angle.stop = 0)

mglyph3 <- metroglyphGrob(x = 200, y = 700,
                          z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                          size = 25, circle.size = 0,
                          angle.start = 0, angle.stop = base::pi)

mglyph4 <- metroglyphGrob(x = 800, y = 700,
                          z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                          size = 25, circle.size = 10,
                          angle.start = 0, angle.stop = base::pi)

grid::grid.newpage()
grid::grid.draw(mglyph1)
grid::grid.draw(mglyph2)
grid::grid.draw(mglyph3)
grid::grid.draw(mglyph4)

mglyph1 <- metroglyphGrob(x = 300, y = 200,
                          z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                          size = 25, circle.size = 2, lwd.circle = 3)

mglyph2 <- metroglyphGrob(x = 900, y = 200,
                          z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                          size = 25, circle.size = 5, lwd.circle = 3)

mglyph3 <- metroglyphGrob(x = 300, y = 600,
                          z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                          size = 25, circle.size = 0,
                          angle.start = base::pi, angle.stop = -base::pi,
                          lwd.ray = 3)

mglyph4 <- metroglyphGrob(x = 900, y = 600,
                          z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                          size = 25, circle.size = 10,
                          angle.start = base::pi, angle.stop = -base::pi,
                          lwd.ray = 3)

grid::grid.newpage()
grid::grid.draw(mglyph1)
grid::grid.draw(mglyph2)
grid::grid.draw(mglyph3)
grid::grid.draw(mglyph4)

mglyph1 <- metroglyphGrob(x = 300, y = 200,
                          z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                          size = 25, circle.size = 2, lwd.circle = 3,
                          col.ray = RColorBrewer::brewer.pal(6, "Dark2"),
                          col.circle = "gray")
```

```
mglyph2 <- metroglyphGrob(x = 900, y = 200,
                          z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                          size = 25, circle.size = 5, lwd.circle = 3,
                          col.ray = RColorBrewer::brewer.pal(6, "Dark2"),
                          col.circle = "white", fill = "black")

mglyph3 <- metroglyphGrob(x = 300, y = 600,
                          z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                          size = 25, circle.size = 0,
                          angle.start = base::pi, angle.stop = -base::pi,
                          lwd.ray = 3,
                          col.ray = RColorBrewer::brewer.pal(6, "Dark2"))

mglyph4 <- metroglyphGrob(x = 900, y = 600,
                          z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                          size = 25, circle.size = 10,
                          angle.start = base::pi, angle.stop = -base::pi,
                          lwd.ray = 5, lwd.circle = 15,
                          col.ray = RColorBrewer::brewer.pal(6, "Dark2"),
                          col.circle = "white", fill = "gray")

grid::grid.newpage()
grid::grid.draw(mglyph1)
grid::grid.draw(mglyph2)
grid::grid.draw(mglyph3)
grid::grid.draw(mglyph4)

mg1 <- metroglyphGrob(x = 300, y = 200,
                      z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                      size = 15, circle.size = 5,
                      lwd.ray = 5)

mg2 <- metroglyphGrob(x = 500, y = 400,
                      z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                      size = 15, circle.size = 5,
                      lwd.ray = 5, lineend = "butt")

mg3 <- metroglyphGrob(x = 700, y = 600,
                      z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                      size = 15, circle.size = 5,
                      lwd.ray = 5, lineend = "square")

grid::grid.newpage()
grid::grid.draw(mg1)
grid::grid.draw(mg2)
grid::grid.draw(mg3)

gl <- split(x = rep(c(1, 2, 3), 6),
            f = rep(1:6, each = 3))

mglyph1 <- metroglyphGrob(x = 200, y = 200,
                          z = c(1, 3, 2, 1, 2, 3),
                          size = 6, circle.size = 2, lwd.circle = 3,
                          draw.grid = TRUE, grid.levels = gl)

mglyph2 <- metroglyphGrob(x = 800, y = 200,
```

```
                          z = c(1, 3, 2, 1, 2, 3),
                          size = 6, circle.size = 5, lwd.circle = 3,
                          draw.grid = TRUE, grid.levels = gl)

mglyph3 <- metroglyphGrob(x = 250, y = 600,
                          z = c(1, 3, 2, 1, 2, 3),
                          size = 6, circle.size = 0,
                          angle.start = base::pi, angle.stop = -base::pi,
                          lwd.ray = 3,
                          draw.grid = TRUE, grid.levels = gl)

mglyph4 <- metroglyphGrob(x = 850, y = 600,
                          z = c(1, 3, 2, 1, 2, 3),
                          size = 6, circle.size = 10,
                          angle.start = base::pi, angle.stop = -base::pi,
                          lwd.ray = 3,
                          draw.grid = TRUE, grid.levels = gl)

grid::grid.newpage()
grid::grid.draw(mglyph1)
grid::grid.draw(mglyph2)
grid::grid.draw(mglyph3)
grid::grid.draw(mglyph4)

gl <- split(x = rep(c(0, 1, 2), 6),
            f = rep(1:6, each = 3))

mglyph1 <- metroglyphGrob(x = 200, y = 200,
                          z = c(0, 2, 1, 0, 1, 2),
                          size = 10, circle.size = 2, lwd.circle = 3,
                          draw.grid = TRUE, grid.levels = gl,
                          col.ray = RColorBrewer::brewer.pal(6, "Dark2"),
                          col.points = NA)

mglyph2 <- metroglyphGrob(x = 800, y = 200,
                          z = c(0, 2, 1, 0, 1, 2),
                          size = 10, circle.size = 5, lwd.circle = 3,
                          draw.grid = TRUE, grid.levels = gl,
                          col.ray = RColorBrewer::brewer.pal(6, "Dark2"))

mglyph3 <- metroglyphGrob(x = 250, y = 600,
                          z = c(0, 2, 1, 0, 1, 2),
                          size = 10, circle.size = 0,
                          angle.start = base::pi, angle.stop = -base::pi,
                          lwd.ray = 3,
                          draw.grid = TRUE, grid.levels = gl,
                          col.ray = RColorBrewer::brewer.pal(6, "Dark2"),
                          col.points = "white")

mglyph4 <- metroglyphGrob(x = 850, y = 600,
                          z = c(0, 2, 1, 0, 1, 2),
                          size = 10, circle.size = 10,
                          angle.start = base::pi, angle.stop = -base::pi,
                          lwd.ray = 3,
                          draw.grid = TRUE, grid.levels = gl,
                          col.ray = RColorBrewer::brewer.pal(6, "Dark2"),
                          col.points = NA, point.size = 20)
```

```
grid::grid.newpage()
grid::grid.draw(mglyph1)
grid::grid.draw(mglyph2)
grid::grid.draw(mglyph3)
grid::grid.draw(mglyph4)
```

`pieglyphGrob` *Draw a Pie Glyph*

Description

Uses Grid graphics to draw a circular pie or clock glyph (Ward and Lipchak 2000; Fuchs et al. 2013).

Usage

```
pieglyphGrob(
  x = 0.5,
  y = 0.5,
  z,
  size = 1,
  edges = 200,
  col = "black",
  fill = NA,
  lwd = 1,
  alpha = 1,
  angle.start = 0,
  angle.stop = 2 * base::pi,
  linejoin = c("mitre", "round", "bevel"),
  scale.segment = FALSE,
  scale.radius = TRUE,
  grid.levels = NULL,
  draw.grid = FALSE,
  col.grid = "grey",
  lwd.grid = lwd
)
```

Arguments

`x`	A numeric vector or unit object specifying x-locations.
`y`	A numeric vector or unit object specifying y-locations.
`z`	A numeric vector specifying the values to be plotted as dimensions of the pie glyph according to the arguments `scale.segment` or `scale.radius`.
`size`	The size of glyphs.
`edges`	The number of edges of the polygon to depict the circular glyph outline.
`col`	The line colour.
`fill`	The fill colour.
`lwd`	The line width.

alpha	The alpha transparency value.
angle.start	The start angle for the glyph in radians. Default is zero.
angle.stop	The stop angle for the glyph in radians. Default is 2π.
linejoin	The line join style for the pie segment polygons. Either "mitre", "round" or "bevel".
scale.segment	logical. If TRUE, the segments (pie slices) are scaled according to value of z.
scale.radius	logical. If TRUE, the radius of segments (pie slices) are scaled according to value of z.
grid.levels	A list of grid levels (as vectors) corresponding to the values in z at which grid lines are to be plotted. The values in z should be present in the list specified.
draw.grid	logical. If TRUE, grid lines are plotted along the segments when scale.radius = TRUE. Default is FALSE.
col.grid	The colour of the grid lines.
lwd.grid	The line width of the grid lines.

Value

A gTree object.

References

Fuchs J, Fischer F, Mansmann F, Bertini E, Isenberg P (2013). "Evaluation of alternative glyph designs for time series data in a small multiple setting." In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 3237–3246. ISBN 978-1-4503-1899-0.

Ward MO, Lipchak BN (2000). "A visualization tool for exploratory analysis of cyclic multivariate data." *Metrika*, **51**(1), 27–37.

See Also

geom_pieglyph

Other grobs: dotglyphGrob(), metroglyphGrob(), profileglyphGrob(), starglyphGrob(), tileglyphGrob()

Examples

```
p1 <- pieglyphGrob(x = 250, y = 200,
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 20)

p2 <- pieglyphGrob(x = 500, y = 200,
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 20, scale.radius = FALSE)

p3 <- pieglyphGrob(x = 900, y = 200,
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 20, scale.segment = TRUE, scale.radius = FALSE)

p4 <- pieglyphGrob(x = 250, y = 650,
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 20, angle.start = 0, angle.stop = base::pi)
```

```
p5 <- pieglyphGrob(x = 500, y = 650,
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 20, scale.radius = FALSE,
                   angle.start = 0, angle.stop = base::pi)

p6 <- pieglyphGrob(x = 900, y = 650,
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 20, scale.segment = TRUE, scale.radius = FALSE,
                   angle.start = 0, angle.stop = base::pi)

grid::grid.newpage()
grid::grid.draw(p1)
grid::grid.draw(p2)
grid::grid.draw(p3)
grid::grid.draw(p4)
grid::grid.draw(p5)
grid::grid.draw(p6)

p1 <- pieglyphGrob(x = 250, y = 200,
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 20, fill = RColorBrewer::brewer.pal(6, "Dark2"))

p2 <- pieglyphGrob(x = 500, y = 200,
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 20, scale.radius = FALSE,
                   fill = RColorBrewer::brewer.pal(6, "Dark2"))

p3 <- pieglyphGrob(x = 900, y = 200,
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 20, scale.segment = TRUE, scale.radius = FALSE,
                   fill = RColorBrewer::brewer.pal(6, "Dark2"))

p4 <- pieglyphGrob(x = 250, y = 650,
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 20, angle.start = 0, angle.stop = base::pi,
                   fill = RColorBrewer::brewer.pal(6, "Dark2"))

p5 <- pieglyphGrob(x = 500, y = 650,
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 20, scale.radius = FALSE,
                   angle.start = 0, angle.stop = base::pi,
                   fill = RColorBrewer::brewer.pal(6, "Dark2"))

p6 <- pieglyphGrob(x = 900, y = 650,
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 20, scale.segment = TRUE, scale.radius = FALSE,
                   angle.start = 0, angle.stop = base::pi,
                   fill = RColorBrewer::brewer.pal(6, "Dark2"))

grid::grid.newpage()
grid::grid.draw(p1)
grid::grid.draw(p2)
grid::grid.draw(p3)
grid::grid.draw(p4)
grid::grid.draw(p5)
grid::grid.draw(p6)
```

```
p1 <- pieglyphGrob(x = 300, y = 250,
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 20, lwd = 5)

p2 <- pieglyphGrob(x = 500, y = 450,
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 20, lwd = 5, linejoin = "round")

p3 <- pieglyphGrob(x = 700, y = 650,
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 20, lwd = 5, linejoin = "bevel")

grid::grid.newpage()
grid::grid.draw(p1)
grid::grid.draw(p2)
grid::grid.draw(p3)

dims = c(1, 3, 2, 1, 2, 3)
gl <- split(x = rep(c(1, 2, 3), 6),
            f = rep(1:6, each = 3))

p1 <- pieglyphGrob(x = 200, y = 250,
                   z = dims, size = 8,
                   draw.grid = TRUE, grid.levels = gl,
                   lwd = 2, col.grid = "black")

p2 <- pieglyphGrob(x = 700, y = 250,
                   angle.start = 0, angle.stop = base::pi,
                   z = dims, size = 8,
                   draw.grid = TRUE, grid.levels = gl,
                   lwd = 2, col.grid = "black")

p3 <- pieglyphGrob(x = 200, y = 600,
                   z = dims, size = 8, scale.segment = TRUE,
                   draw.grid = TRUE, grid.levels = gl,
                   lwd = 2, col.grid = "black")

p4 <- pieglyphGrob(x = 700, y = 600,
                   angle.start = 0, angle.stop = base::pi,
                   z = dims, size = 8, scale.segment = TRUE,
                   draw.grid = TRUE, grid.levels = gl,
                   lwd = 2, col.grid = "black")

grid::grid.newpage()
grid::grid.draw(p1)
grid::grid.draw(p2)
grid::grid.draw(p3)
grid::grid.draw(p4)

dims = c(1, 3, 2, 1, 2, 3)
gl <- split(x = rep(c(1, 2, 3), 6),
            f = rep(1:6, each = 3))

p1 <- pieglyphGrob(x = 200, y = 250,
                   z = dims, size = 8, col = "white",
                   draw.grid = TRUE, grid.levels = gl,
                   lwd = 3, col.grid = "white",
```

```
                    fill = RColorBrewer::brewer.pal(6, "Dark2"))

p2 <- pieglyphGrob(x = 700, y = 250,
                    angle.start = 0, angle.stop = base::pi,
                    z = dims, size = 8, col = "white",
                    draw.grid = TRUE, grid.levels = gl,
                    lwd = 3, col.grid = "white",
                    fill = RColorBrewer::brewer.pal(6, "Dark2"))

p3 <- pieglyphGrob(x = 200, y = 600,
                    z = dims, size = 8,
                    col = "white", scale.segment = TRUE,
                    draw.grid = TRUE, grid.levels = gl,
                    lwd = 3, col.grid = "white",
                    fill = RColorBrewer::brewer.pal(6, "Dark2"))

p4 <- pieglyphGrob(x = 700, y = 600,
                    angle.start = 0, angle.stop = base::pi,
                    z = dims, size = 8,
                    col = "white", scale.segment = TRUE,
                    draw.grid = TRUE, grid.levels = gl,
                    lwd = 3, col.grid = "white",
                    fill = RColorBrewer::brewer.pal(6, "Dark2"))


grid::grid.newpage()
grid::grid.draw(p1)
grid::grid.draw(p2)
grid::grid.draw(p3)
grid::grid.draw(p4)
```

`profileglyphGrob` *Draw a Profile Glyph*

Description

Uses `Grid` graphics to draw a profile glyph (Chambers et al. 1983; DuToit et al. 1986).

Usage

```
profileglyphGrob(
  x = 0.5,
  y = 0.5,
  z,
  size = 1,
  col.bar = "black",
  col.line = "black",
  fill = NA,
  lwd.bar = 1,
  lwd.line = 1,
  alpha = 1,
  width = 5,
```

```
  flip.axes = FALSE,
  bar = TRUE,
  line = TRUE,
  mirror = TRUE,
  linejoin = c("mitre", "round", "bevel"),
  lineend = c("round", "butt", "square"),
  grid.levels = NULL,
  draw.grid = FALSE,
  col.grid = "grey",
  lwd.grid = 1
)
```

Arguments

x	A numeric vector or unit object specifying x-locations.
y	A numeric vector or unit object specifying y-locations.
z	A numeric vector specifying the values to be plotted as dimensions of the profile (length of the bars).
size	The size of glyphs.
col.bar	The colour of bars.
col.line	The colour of profile line(s).
fill	The fill colour.
lwd.bar	The line width of the bars.
lwd.line	The line width of the profile line(s)
alpha	The alpha transparency value.
width	The width of the bars.
flip.axes	logical. If TRUE, axes are flipped.
bar	logical. If TRUE, profile bars are plotted.
line	logical. If TRUE, profile line is plotted.
mirror	logical. If TRUE, mirror profile is plotted.
linejoin	The line join style for the profile line(s) and bars. Either "mitre", "round" or "bevel".
lineend	The line end style for the whisker lines. Either "round", "butt" or "square".
grid.levels	A list of grid levels (as vectors) corresponding to the values in z at which grid lines are to be plotted. The values in z should be present in the list specified.
draw.grid	logical. If TRUE, grid lines are plotted along the bars. Default is FALSE.
col.grid	The colour of the grid lines.
lwd.grid	The line width of the grid lines.

Value

A gTree object.

References

Chambers JM, Cleveland WS, Kleiner B, Tukey PA (1983). *Graphical Methods for Data Analysis*. Chapman and Hall/CRC, Boca Raton. ISBN 978-1-351-07230-4.

DuToit SHC, Steyn AGW, Stumpf RH (1986). *Graphical Exploratory Data Analysis*, Springer Texts in Statistics. Springer-Verlag, New York. ISBN 978-1-4612-9371-2.

See Also

geom_profileglyph

Other grobs: dotglyphGrob(), metroglyphGrob(), pieglyphGrob(), starglyphGrob(), tileglyphGrob()

Examples

```
# mirror = TRUE
dims = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33)
barglyph <- profileglyphGrob(x = 200, y = 200, z = dims,
                             size = 20)

barprofileglyph <- profileglyphGrob(x = 450, y = 200, z = dims,
                                    size = 20, line = FALSE)

profileglyph <- profileglyphGrob(x = 700, y = 200, z = dims,
                                 size = 20, line = TRUE, bar = FALSE)
grid::grid.newpage()
grid::grid.draw(barglyph)
grid::grid.draw(barprofileglyph)
grid::grid.draw(profileglyph)

barglyph <- profileglyphGrob(x = 200, y = 450, z = dims,
                             size = 20,
                             col.bar = "salmon", col.line = "salmon")

barprofileglyph <- profileglyphGrob(x = 450, y = 450, z = dims,
                                    size = 20, line = FALSE,
                                    col.bar = "cyan")

profileglyph <- profileglyphGrob(x = 700, y = 450, z = dims,
                                 size = 20, line = TRUE, bar = FALSE,
                                 col.line = "green")

grid::grid.draw(barglyph)
grid::grid.draw(barprofileglyph)
grid::grid.draw(profileglyph)

barglyph <- profileglyphGrob(x = 200, y = 700, z = dims, size = 20,
                             fill = "salmon")

barprofileglyph <- profileglyphGrob(x = 450, y = 700, z = dims,
                                    size = 20, line = FALSE,
                                    fill = "cyan")

profileglyph <- profileglyphGrob(x = 700, y = 700, z = dims, size = 20,
                                 line = TRUE, bar = FALSE,
                                 fill = "green")

grid::grid.draw(barglyph)
grid::grid.draw(barprofileglyph)
grid::grid.draw(profileglyph)


# mirror = FALSE
dims = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33)
barglyph <- profileglyphGrob(x = 200, y = 300, z = dims,
```

```
                              size = 20,
                              mirror = FALSE)

barprofileglyph <- profileglyphGrob(x = 450, y = 300, z = dims,
                                    size = 20, line = FALSE,
                                    mirror = FALSE)

profileglyph <- profileglyphGrob(x = 700, y = 300, z = dims,
                                 size = 20, line = TRUE, bar = FALSE,
                                 mirror = FALSE)
grid::grid.newpage()
grid::grid.draw(barglyph)
grid::grid.draw(barprofileglyph)
grid::grid.draw(profileglyph)

barglyph <- profileglyphGrob(x = 200, y = 550, z = dims,
                              size = 20, mirror = FALSE,
                              col.bar = "salmon", col.line = "salmon")

barprofileglyph <- profileglyphGrob(x = 450, y = 550, z = dims,
                                    size = 20, line = FALSE, mirror = FALSE,
                                    col.bar = "cyan")

profileglyph <- profileglyphGrob(x = 700, y = 550, z = dims,
                                 size = 20, line = TRUE, bar = FALSE,
                                 mirror = FALSE, col.line = "green")

grid::grid.draw(barglyph)
grid::grid.draw(barprofileglyph)
grid::grid.draw(profileglyph)#'

barglyph <- profileglyphGrob(x = 200, y = 800, z = dims, size = 20,
                              fill = "salmon", mirror = FALSE)

barprofileglyph <- profileglyphGrob(x = 450, y = 800, z = dims,
                                    size = 20, line = FALSE, mirror = FALSE,
                                    fill = "cyan")

profileglyph <- profileglyphGrob(x = 700, y = 800, z = dims, size = 20,
                                 line = TRUE, bar = FALSE,
                                 mirror = FALSE, fill = "green")

grid::grid.draw(barglyph)
grid::grid.draw(barprofileglyph)
grid::grid.draw(profileglyph)

# mirror = TRUE, flip.axes = TRUE
dims = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33)
barglyph <- profileglyphGrob(x = 200, y = 200, z = dims,
                              size = 20, flip.axes = TRUE)

barprofileglyph <- profileglyphGrob(x = 450, y = 200, z = dims,
                                    size = 20, line = FALSE,
                                    flip.axes = TRUE)

profileglyph <- profileglyphGrob(x = 700, y = 200, z = dims,
                                 size = 20, line = TRUE, bar = FALSE,
```

```
                                  flip.axes = TRUE)
grid::grid.newpage()
grid::grid.draw(barglyph)
grid::grid.draw(barprofileglyph)
grid::grid.draw(profileglyph)

barglyph <- profileglyphGrob(x = 200, y = 450, z = dims,
                             size = 20, flip.axes = TRUE,
                             col.bar = "salmon", col.line = "salmon")

barprofileglyph <- profileglyphGrob(x = 450, y = 450, z = dims,
                                    size = 20, line = FALSE,
                                    flip.axes = TRUE,
                                    col.bar = "cyan")

profileglyph <- profileglyphGrob(x = 700, y = 450, z = dims,
                                 size = 20, line = TRUE, bar = FALSE,
                                 flip.axes = TRUE,
                                 col.line = "green")

grid::grid.draw(barglyph)
grid::grid.draw(barprofileglyph)
grid::grid.draw(profileglyph)

barglyph <- profileglyphGrob(x = 200, y = 700, z = dims, size = 20,
                             flip.axes = TRUE,
                             fill = "salmon")

barprofileglyph <- profileglyphGrob(x = 450, y = 700, z = dims,
                                    size = 20, line = FALSE,
                                    flip.axes = TRUE,
                                    fill = "cyan")

profileglyph <- profileglyphGrob(x = 700, y = 700, z = dims, size = 20,
                                 line = TRUE, bar = FALSE,
                                 flip.axes = TRUE,
                                 fill = "green")

grid::grid.draw(barglyph)
grid::grid.draw(barprofileglyph)
grid::grid.draw(profileglyph)


# mirror = FALSE, flip.axes = TRUE
dims = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33)
barglyph <- profileglyphGrob(x = 200, y = 200, z = dims,
                             size = 20, flip.axes = TRUE,
                             mirror = FALSE)

barprofileglyph <- profileglyphGrob(x = 450, y = 200, z = dims,
                                    size = 20, line = FALSE,
                                    flip.axes = TRUE,
                                    mirror = FALSE)

profileglyph <- profileglyphGrob(x = 700, y = 200, z = dims,
                                 size = 20, line = TRUE, bar = FALSE,
                                 flip.axes = TRUE,
```

```
                                  mirror = FALSE)
grid::grid.newpage()
grid::grid.draw(barglyph)
grid::grid.draw(barprofileglyph)
grid::grid.draw(profileglyph)

barglyph <- profileglyphGrob(x = 200, y = 450, z = dims,
                             size = 20, mirror = FALSE,
                             flip.axes = TRUE,
                             col.bar = "salmon", col.line = "salmon")

barprofileglyph <- profileglyphGrob(x = 450, y = 450, z = dims,
                                    size = 20, line = FALSE, mirror = FALSE,
                                    flip.axes = TRUE,
                                    col.bar = "cyan")

profileglyph <- profileglyphGrob(x = 700, y = 450, z = dims,
                                 size = 20, line = TRUE, bar = FALSE,
                                 flip.axes = TRUE,
                                 mirror = FALSE, col.line = "green")

grid::grid.draw(barglyph)
grid::grid.draw(barprofileglyph)
grid::grid.draw(profileglyph)

barglyph <- profileglyphGrob(x = 200, y = 700, z = dims, size = 20,
                             flip.axes = TRUE,
                             fill = "salmon", mirror = FALSE)

barprofileglyph <- profileglyphGrob(x = 450, y = 700, z = dims,
                                    size = 20, line = FALSE, mirror = FALSE,
                                    flip.axes = TRUE,
                                    fill = "cyan")

profileglyph <- profileglyphGrob(x = 700, y = 700, z = dims, size = 20,
                                 line = TRUE, bar = FALSE,
                                 flip.axes = TRUE,
                                 mirror = FALSE, fill = "green")

grid::grid.draw(barglyph)
grid::grid.draw(barprofileglyph)
grid::grid.draw(profileglyph)

# linejoin variants
dims = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33)
pg1 <- profileglyphGrob(x = 200, y = 150, z = dims,
                        size = 25, lwd.bar = 5, width = 8)

pg2 <- profileglyphGrob(x = 500, y = 400, z = dims,
                        size = 25, lwd.bar = 5, width = 8,
                        linejoin = "round")

pg3 <- profileglyphGrob(x = 800, y = 650, z = dims,
                        size = 25, lwd.bar = 5, width = 8,
                        linejoin = "bevel")

grid::grid.newpage()
```

```
grid::grid.draw(pg1)
grid::grid.draw(pg2)
grid::grid.draw(pg3)

dims = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33)
pg1 <- profileglyphGrob(x = 200, y = 150, z = dims,
                        size = 25, lwd.line = 5, width = 8,
                        bar = FALSE)

pg2 <- profileglyphGrob(x = 500, y = 400, z = dims,
                        size = 25, lwd.line = 5, width = 8,
                        linejoin = "round", bar = FALSE)

pg3 <- profileglyphGrob(x = 800, y = 650, z = dims,
                        size = 25, lwd.line = 5, width = 8,
                        linejoin = "bevel", bar = FALSE)

grid::grid.newpage()
grid::grid.draw(pg1)
grid::grid.draw(pg2)
grid::grid.draw(pg3)

dims = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33)
pg1 <- profileglyphGrob(x = 200, y = 150, z = dims,
                        size = 25, lwd.bar = 5, width = 8,
                        line = FALSE)

pg2 <- profileglyphGrob(x = 500, y = 400, z = dims,
                        size = 25, lwd.bar = 5, width = 8,
                        linejoin = "round", line = FALSE)

pg3 <- profileglyphGrob(x = 800, y = 650, z = dims,
                        size = 25, lwd.bar = 5, width = 8,
                        linejoin = "bevel", line = FALSE)

grid::grid.newpage()
grid::grid.draw(pg1)
grid::grid.draw(pg2)
grid::grid.draw(pg3)

# lineend variants
dims = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33)
pg1 <- profileglyphGrob(x = 200, y = 150, z = dims,
                        size = 25, lwd.line = 5, width = 8)

pg2 <- profileglyphGrob(x = 500, y = 400, z = dims,
                        size = 25, lwd.line = 5, width = 8,
                        lineend = "butt")

pg3 <- profileglyphGrob(x = 800, y = 650, z = dims,
                        size = 25, lwd.line = 5, width = 8,
                        lineend = "square")

grid::grid.newpage()
grid::grid.draw(pg1)
grid::grid.draw(pg2)
grid::grid.draw(pg3)
```

```
# Bars with multiple fill colours
dims = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33)
bg1 <- profileglyphGrob(x = 200, y = 200, z = dims,
                        size = 20,
                        fill = RColorBrewer::brewer.pal(6, "Dark2"))

bpg1 <- profileglyphGrob(x = 700, y = 200, z = dims,
                         size = 20, line = FALSE,
                         fill = RColorBrewer::brewer.pal(6, "Dark2"))

bg2 <- profileglyphGrob(x = 350, y = 450, z = dims,
                        size = 20, mirror = FALSE,
                        fill = RColorBrewer::brewer.pal(6, "Dark2"))

bpg2 <- profileglyphGrob(x = 850, y = 450, z = dims,
                         size = 20, line = FALSE, mirror = FALSE,
                         fill = RColorBrewer::brewer.pal(6, "Dark2"))

bg3 <- profileglyphGrob(x = 200, y = 650, z = dims,
                        size = 20, flip.axes = TRUE,
                        fill = RColorBrewer::brewer.pal(6, "Dark2"))

bpg3 <- profileglyphGrob(x = 700, y = 650, z = dims,
                         size = 20, line = FALSE, flip.axes = TRUE,
                         fill = RColorBrewer::brewer.pal(6, "Dark2"))

bg4 <- profileglyphGrob(x = 350, y = 700, z = dims,
                        size = 20, mirror = FALSE, flip.axes = TRUE,
                        fill = RColorBrewer::brewer.pal(6, "Dark2"))

bpg4 <- profileglyphGrob(x = 850, y = 700, z = dims,
                         size = 20, line = FALSE, mirror = FALSE,
                         flip.axes = TRUE,
                         fill = RColorBrewer::brewer.pal(6, "Dark2"))


grid::grid.newpage()
grid::grid.draw(bg1)
grid::grid.draw(bpg1)
grid::grid.draw(bg2)
grid::grid.draw(bpg2)
grid::grid.draw(bg3)
grid::grid.draw(bpg3)
grid::grid.draw(bg4)
grid::grid.draw(bpg4)

# Grid lines
dims = c(1, 3, 2, 1, 2, 3)
gl <- split(x = rep(c(1, 2, 3), 6),
            f = rep(1:6, each = 3))

bg1 <- profileglyphGrob(x = 150, y = 200, z = dims,
                        size = 10, width = 5,
                        draw.grid = TRUE, lwd.bar = 5,
                        grid.levels = gl, col.grid = "black")
```

```
bg2 <- profileglyphGrob(x = 400, y = 250, z = dims,
                        size = 10, width = 5, lwd.bar = 5,
                        draw.grid = TRUE, mirror = FALSE,
                        grid.levels = gl, col.grid = "black")

bg3 <- profileglyphGrob(x = 650, y = 200, z = dims,
                        size = 10, width = 5, flip.axes = TRUE,
                        draw.grid = TRUE, lwd.bar = 5,
                        grid.levels = gl, col.grid = "black")

bg4 <- profileglyphGrob(x = 800, y = 200, z = dims,
                        size = 10, width = 5, flip.axes = TRUE,
                        draw.grid = TRUE, mirror = FALSE,
                        grid.levels = gl, col.grid = "black",
                        lwd.bar = 5)

bg5 <- profileglyphGrob(x = 150, y = 500, z = dims,
                        size = 10, width = 5,
                        draw.grid = TRUE, lwd.bar = 5,
                        grid.levels = gl, col.grid = "white",
                        col.bar = "white", line = FALSE,
                        fill = RColorBrewer::brewer.pal(6, "Dark2"))

bg6 <- profileglyphGrob(x = 400, y = 550, z = dims,
                        size = 10, width = 5, lwd.bar = 5,
                        draw.grid = TRUE, mirror = FALSE,
                        grid.levels = gl, col.grid = "white",
                        col.bar = "white", line = FALSE,
                        fill = RColorBrewer::brewer.pal(6, "Dark2"))

bg7 <- profileglyphGrob(x = 650, y = 500, z = dims,
                        size = 10, width = 5, flip.axes = TRUE,
                        draw.grid = TRUE, lwd.bar = 5,
                        grid.levels = gl, col.grid = "white",
                        col.bar = "white", line = FALSE,
                        fill = RColorBrewer::brewer.pal(6, "Dark2"))

bg8 <- profileglyphGrob(x = 800, y = 500, z = dims,
                        size = 10, width = 5, flip.axes = TRUE,
                        draw.grid = TRUE, mirror = FALSE,
                        grid.levels = gl, col.grid = "white",
                        col.bar = "white", lwd.bar = 5, line = FALSE,
                        fill = RColorBrewer::brewer.pal(6, "Dark2"))


grid::grid.newpage()
grid::grid.draw(bg1)
grid::grid.draw(bg2)
grid::grid.draw(bg3)
grid::grid.draw(bg4)
grid::grid.draw(bg5)
grid::grid.draw(bg6)
grid::grid.draw(bg7)
grid::grid.draw(bg8)
```

starglyphGrob *Draw a Star Glyph*

Description

Uses Grid graphics to draw a star glyph (Siegel et al. 1972; Chambers et al. 1983; DuToit et al. 1986).

Usage

```
starglyphGrob(
  x = 0.5,
  y = 0.5,
  z,
  size = 1,
  col.whisker = "black",
  col.contour = "black",
  col.points = "black",
  fill = NA,
  lwd.whisker = 1,
  lwd.contour = 1,
  alpha = 1,
  angle.start = 0,
  angle.stop = 2 * base::pi,
  whisker = TRUE,
  contour = TRUE,
  linejoin = c("mitre", "round", "bevel"),
  lineend = c("round", "butt", "square"),
  grid.levels = NULL,
  draw.grid = FALSE,
  point.size = 10
)
```

Arguments

x	A numeric vector or unit object specifying x-locations.
y	A numeric vector or unit object specifying y-locations.
z	A numeric vector specifying the distance of star glyph points from the centre.
size	The size of glyphs.
col.whisker	The colour of whiskers.
col.contour	The colour of contours.
col.points	The colour of grid points.
fill	The fill colour.
lwd.whisker	The whisker line width.
lwd.contour	The contour line width.
alpha	The alpha transparency value.
angle.start	The start angle for the glyph in radians. Default is zero.

angle.stop	The stop angle for the glyph in radians. Default is 2π.
whisker	logical. If TRUE, plots the star glyph whiskers.
contour	logical. If TRUE, plots the star glyph contours.
linejoin	The line join style for the contour polygon. Either "mitre", "round" or "bevel".
lineend	The line end style for the whisker lines. Either "round", "butt" or "square".
grid.levels	A list of grid levels (as vectors) corresponding to the values in z at which points are to be plotted. The values in z should be present in the list specified.
draw.grid	logical. If TRUE, grid points are plotted along the whiskers. Default is FALSE.
point.size	The size of the grid points in native units.

Value

A gTree object.

References

Chambers JM, Cleveland WS, Kleiner B, Tukey PA (1983). *Graphical Methods for Data Analysis*. Chapman and Hall/CRC, Boca Raton. ISBN 978-1-351-07230-4.

DuToit SHC, Steyn AGW, Stumpf RH (1986). *Graphical Exploratory Data Analysis*, Springer Texts in Statistics. Springer-Verlag, New York. ISBN 978-1-4612-9371-2.

Siegel JH, Farrell EJ, Goldwyn RM, Friedman HP (1972). "The surgical implications of physiologic patterns in myocardial infarction shock." *Surgery*, **72**(1), 126–141.

See Also

geom_starglyph

Other grobs: dotglyphGrob(), metroglyphGrob(), pieglyphGrob(), profileglyphGrob(), tileglyphGrob()

Examples

```
sg1 <- starglyphGrob(x = 400, y = 150,
                     z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 25)

sg2 <- starglyphGrob(x = 400, y = 400,
                     z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 25,
                     lwd.whisker = 3,
                     lwd.contour = 0.1)

sg3 <- starglyphGrob(x = 400, y = 650,
                     z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 25,
                     lwd.whisker = 0.1,
                     lwd.contour = 3)

sg4 <- starglyphGrob(x = 800, y = 300,
                     z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 25,
                     angle.start = 0, angle.stop = base::pi)

sg5 <- starglyphGrob(x = 800, y = 550,
                     z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 25,
                     lwd.whisker = 3,
                     lwd.contour = 0.1,
```

```
                    angle.start = 0, angle.stop = base::pi)

sg6 <- starglyphGrob(x = 800, y = 800,
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 25,
                    lwd.whisker = 0.1,
                    lwd.contour = 3,
                    angle.start = 0, angle.stop = base::pi)

grid::grid.newpage()
grid::grid.draw(sg1)
grid::grid.draw(sg2)
grid::grid.draw(sg3)
grid::grid.draw(sg4)
grid::grid.draw(sg5)
grid::grid.draw(sg6)

sg1 <- starglyphGrob(x = 400, y = 150,
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 25,
                    fill = "salmon")

sg2 <- starglyphGrob(x = 400, y = 400,
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 25,
                    lwd.whisker = 3,
                    lwd.contour = 0.1,
                    fill = "cyan")

sg3 <- starglyphGrob(x = 400, y = 650,
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 25,
                    lwd.whisker = 0.1,
                    lwd.contour = 3,
                    fill = "green")

sg4 <- starglyphGrob(x = 800, y = 300,
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 25,
                    angle.start = 0, angle.stop = base::pi,
                    fill = "salmon")

sg5 <- starglyphGrob(x = 800, y = 550,
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 25,
                    lwd.whisker = 3,
                    lwd.contour = 0.1,
                    angle.start = 0, angle.stop = base::pi,
                    fill = "cyan")

sg6 <- starglyphGrob(x = 800, y = 800,
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 25,
                    lwd.whisker = 0.1,
                    lwd.contour = 3,
                    angle.start = 0, angle.stop = base::pi,
                    fill = "green")

grid::grid.newpage()
grid::grid.draw(sg1)
grid::grid.draw(sg2)
grid::grid.draw(sg3)
grid::grid.draw(sg4)
grid::grid.draw(sg5)
```

```
grid::grid.draw(sg6)

sg1 <- starglyphGrob(x = 400, y = 150,
                     z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 25,
                     col.whisker = RColorBrewer::brewer.pal(6, "Dark2"),
                     col.contour = "gray")

sg2 <- starglyphGrob(x = 400, y = 400,
                     z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 25,
                     lwd.whisker = 3,
                     lwd.contour = 0.1,
                     col.whisker = RColorBrewer::brewer.pal(6, "Dark2"),
                     col.contour = "gray")

sg3 <- starglyphGrob(x = 400, y = 650,
                     z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 25,
                     lwd.whisker = 0.1,
                     lwd.contour = 3,
                     col.whisker = RColorBrewer::brewer.pal(6, "Dark2"),
                     col.contour = "gray")

sg4 <- starglyphGrob(x = 800, y = 300,
                     z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 25,
                     angle.start = 0, angle.stop = base::pi,
                     col.whisker = RColorBrewer::brewer.pal(6, "Dark2"),
                     col.contour = "gray")

sg5 <- starglyphGrob(x = 800, y = 550,
                     z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 25,
                     lwd.whisker = 3,
                     lwd.contour = 0.1,
                     angle.start = 0, angle.stop = base::pi,
                     col.whisker = RColorBrewer::brewer.pal(6, "Dark2"),
                     col.contour = "gray")

sg6 <- starglyphGrob(x = 800, y = 800,
                     z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 25,
                     lwd.whisker = 0.1,
                     lwd.contour = 3,
                     angle.start = 0, angle.stop = base::pi,
                     col.whisker = RColorBrewer::brewer.pal(6, "Dark2"),
                     col.contour = "gray")

grid::grid.newpage()
grid::grid.draw(sg1)
grid::grid.draw(sg2)
grid::grid.draw(sg3)
grid::grid.draw(sg4)
grid::grid.draw(sg5)
grid::grid.draw(sg6)

sg1 <- starglyphGrob(x = 300, y = 250,
                     z = c(0.28, 0.33, 0.8, 1.2, 0.6, 0.5, 0.7), size = 25,
                     lwd.contour = 10)

sg2 <- starglyphGrob(x = 600, y = 300,
                     z = c(0.28, 0.33, 0.8, 1.2, 0.6, 0.5, 0.7), size = 25,
```

```
                    lwd.contour = 10, linejoin = "bevel")

sg3 <- starglyphGrob(x = 900, y = 350,
                    z = c(0.28, 0.33, 0.8, 1.2, 0.6, 0.5, 0.7), size = 25,
                    lwd.contour = 10, linejoin = "round")

grid::grid.newpage()
grid::grid.draw(sg1)
grid::grid.draw(sg2)
grid::grid.draw(sg3)

sg1 <- starglyphGrob(x = 300, y = 250,
                    z = c(0.28, 0.33, 0.8, 1.2, 0.6, 0.5, 0.7), size = 25,
                    lwd.whisker = 10, contour = FALSE)

sg2 <- starglyphGrob(x = 600, y = 300,
                    z = c(0.28, 0.33, 0.8, 1.2, 0.6, 0.5, 0.7), size = 25,
                    lwd.whisker = 10, lineend = "butt", contour = FALSE)

sg3 <- starglyphGrob(x = 900, y = 350,
                    z = c(0.28, 0.33, 0.8, 1.2, 0.6, 0.5, 0.7), size = 25,
                    lwd.whisker = 10, lineend = "square", contour = FALSE)

grid::grid.newpage()
grid::grid.draw(sg1)
grid::grid.draw(sg2)
grid::grid.draw(sg3)

gl <- split(x = rep(c(1, 2, 3), 6),
            f = rep(1:6, each = 3))

sg1 <- starglyphGrob(x = 150, y = 150,
                    z = c(1, 3, 2, 1, 2, 3), size = 5,
                    draw.grid = TRUE, grid.levels = gl)

sg2 <- starglyphGrob(x = 150, y = 400,
                    z = c(1, 3, 2, 1, 2, 3), size = 5,
                    lwd.whisker = 3, col.points = "white",
                    draw.grid = TRUE, grid.levels = gl,
                    contour = FALSE)

sg3 <- starglyphGrob(x = 150, y = 650,
                    z = c(1, 3, 2, 1, 2, 3), size = 5,
                    lwd.contour = 3,
                    draw.grid = FALSE, grid.levels = gl,
                    whisker = FALSE)

sg4 <- starglyphGrob(x = 600, y = 150,
                    z = c(1, 3, 2, 1, 2, 3), size = 5,
                    angle.start = 0, angle.stop = base::pi,
                    draw.grid = TRUE, grid.levels = gl)

sg5 <- starglyphGrob(x = 600, y = 400,
                    z = c(1, 3, 2, 1, 2, 3), size = 5,
                    lwd.whisker = 3,
                    angle.start = 0, angle.stop = base::pi,
                    draw.grid = TRUE, grid.levels = gl,
```

```
                    point.size = 20, contour = FALSE)

sg6 <- starglyphGrob(x = 600, y = 650,
                    z = c(1, 3, 2, 1, 2, 3), size = 5,
                    lwd.contour = 3,
                    angle.start = 0, angle.stop = base::pi,
                    draw.grid = FALSE, grid.levels = gl,
                    whisker = FALSE)

grid::grid.newpage()
grid::grid.draw(sg1)
grid::grid.draw(sg2)
grid::grid.draw(sg3)
grid::grid.draw(sg4)
grid::grid.draw(sg5)
grid::grid.draw(sg6)

gl <- split(x = rep(c(1, 2, 3), 6),
            f = rep(1:6, each = 3))

sg1 <- starglyphGrob(x = 150, y = 150,
                    z = c(1, 3, 2, 1, 2, 3), size = 5,
                    col.whisker = RColorBrewer::brewer.pal(6, "Dark2"),
                    draw.grid = TRUE, grid.levels = gl,
                    col.points = NA, fill = "black")

sg2 <- starglyphGrob(x = 150, y = 400,
                    z = c(1, 3, 2, 1, 2, 3), size = 5,
                    lwd.whisker = 3,
                    draw.grid = TRUE, grid.levels = gl,
                    col.whisker = RColorBrewer::brewer.pal(6, "Dark2"),
                    contour = FALSE)

sg3 <- starglyphGrob(x = 150, y = 650,
                    z = c(1, 3, 2, 1, 2, 3), size = 5,
                    lwd.contour = 3,
                    draw.grid = FALSE, grid.levels = gl,
                    col.whisker = RColorBrewer::brewer.pal(6, "Dark2"),
                    whisker = FALSE)

sg4 <- starglyphGrob(x = 600, y = 150,
                    z = c(1, 3, 2, 1, 2, 3), size = 5,
                    col.contour = "gray",
                    angle.start = 0, angle.stop = base::pi,
                    draw.grid = TRUE, grid.levels = gl,
                    col.whisker = RColorBrewer::brewer.pal(6, "Dark2"),
                    point.size = 10, col.points = "gray")

sg5 <- starglyphGrob(x = 600, y = 400,
                    z = c(1, 3, 2, 1, 2, 3), size = 5,
                    lwd.whisker = 3,
                    angle.start = 0, angle.stop = base::pi,
                    draw.grid = TRUE, grid.levels = gl,
                    col.whisker = RColorBrewer::brewer.pal(6, "Dark2"),
                    point.size = 20, col.points = NA,
                    contour = FALSE)
```

```
sg6 <- starglyphGrob(x = 600, y = 650,
                     z = c(1, 3, 2, 1, 2, 3), size = 5,
                     lwd.contour = 3,
                     angle.start = 0, angle.stop = base::pi,
                     draw.grid = FALSE, grid.levels = gl,
                     whisker = FALSE)

grid::grid.newpage()
grid::grid.draw(sg1)
grid::grid.draw(sg2)
grid::grid.draw(sg3)
grid::grid.draw(sg4)
grid::grid.draw(sg5)
grid::grid.draw(sg6)
```

`tileglyphGrob` *Draw a Tile Glyph*

Description

Uses Grid graphics to draw a tile glyph similar to 'autoglyph' (Beddow 1990) or 'stripe glyph' (Fuchs et al. 2013).

Usage

```
tileglyphGrob(
  x = 0.5,
  y = 0.5,
  z,
  size = 10,
  ratio = 1,
  nrow = 1,
  col = "black",
  fill = NA,
  lwd = 1,
  alpha = 1,
  linejoin = c("mitre", "round", "bevel")
)
```

Arguments

x	A numeric vector or unit object specifying x-locations.
y	A numeric vector or unit object specifying y-locations.
z	A numeric vector specifying the values to be plotted as dimensions in the tileglyph.
size	The size of glyphs.
ratio	The aspect ratio (height / width).
nrow	The number of rows.
col	The line colour.

`fill`	The fill colour.
`lwd`	The line width.
`alpha`	The alpha transparency value.
`linejoin`	The line join style for the tile polygon. Either `"mitre"`, `"round"` or `"bevel"`.

Value

A `grob` object.

References

Beddow J (1990). "Shape coding of multidimensional data on a microcomputer display." In *Proceedings of the First IEEE Conference on Visualization: Visualization '90*, 238–246. ISBN 978-0-8186-2083-6.

Fuchs J, Fischer F, Mansmann F, Bertini E, Isenberg P (2013). "Evaluation of alternative glyph designs for time series data in a small multiple setting." In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 3237–3246. ISBN 978-1-4503-1899-0.

See Also

`geom_tileglyph`

Other grobs: `dotglyphGrob()`, `metroglyphGrob()`, `pieglyphGrob()`, `profileglyphGrob()`, `starglyphGrob()`

Examples

```
tg1 <- tileglyphGrob(x = 150, y = 150,
                     z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7, 4.3),
                     size = 5)

tg2 <- tileglyphGrob(x = 450, y = 150,
                      z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                      size = 5)

tg3 <- tileglyphGrob(x = 150, y = 250,
                      z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7, 4.3),
                      size = 5, nrow = 2)

tg4 <- tileglyphGrob(x = 450, y = 250,
                      z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                      size = 5, nrow = 2)

tg5 <- tileglyphGrob(x = 150, y = 350,
                      z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7, 4.3),
                      size = 5,
                      fill = RColorBrewer::brewer.pal(7, "Dark2"))

tg6 <- tileglyphGrob(x = 450, y = 350,
                      z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                      size = 5,
                      fill = RColorBrewer::brewer.pal(7, "Dark2"))

tg7 <- tileglyphGrob(x = 150, y = 450,
                      z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7, 4.3),
                      size = 5, nrow = 2,
```

```
                    fill = RColorBrewer::brewer.pal(7, "Dark2"))

tg8 <- tileglyphGrob(x = 450, y = 450,
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    size = 5, nrow = 2,
                    fill = RColorBrewer::brewer.pal(7, "Dark2"))

grid::grid.newpage()
grid::grid.draw(tg1)
grid::grid.draw(tg2)
grid::grid.draw(tg3)
grid::grid.draw(tg4)
grid::grid.draw(tg5)
grid::grid.draw(tg6)
grid::grid.draw(tg7)
grid::grid.draw(tg8)

tg1 <- tileglyphGrob(x = 150, y = 150,
                   z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7, 4.3),
                   size = 2, ratio = 6)

tg2 <- tileglyphGrob(x = 450, y = 150,
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    size = 2, ratio = 6)

tg3 <- tileglyphGrob(x = 150, y = 300,
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7, 4.3),
                    size = 2, nrow = 2, ratio = 6)

tg4 <- tileglyphGrob(x = 450, y = 300,
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    size = 2, nrow = 2, ratio = 6)

tg5 <- tileglyphGrob(x = 150, y = 450,
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7, 4.3),
                    size = 2, ratio = 6,
                    fill = RColorBrewer::brewer.pal(7, "Dark2"))

tg6 <- tileglyphGrob(x = 450, y = 450,
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    size = 2, ratio = 6,
                    fill = RColorBrewer::brewer.pal(7, "Dark2"))

tg7 <- tileglyphGrob(x = 150, y = 600,
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7, 4.3),
                    size = 2, nrow = 2, ratio = 6,
                    fill = RColorBrewer::brewer.pal(7, "Dark2"))

tg8 <- tileglyphGrob(x = 450, y = 600,
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    size = 2, nrow = 2, ratio = 6,
                    fill = RColorBrewer::brewer.pal(7, "Dark2"))

grid::grid.newpage()
grid::grid.draw(tg1)
grid::grid.draw(tg2)
grid::grid.draw(tg3)
```

```
grid::grid.draw(tg4)
grid::grid.draw(tg5)
grid::grid.draw(tg6)
grid::grid.draw(tg7)
grid::grid.draw(tg8)

tg1 <- tileglyphGrob(x = 150, y = 150,
                     z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7, 4.3),
                     size = 5, nrow = 2, lwd = 5)

tg2 <- tileglyphGrob(x = 300, y = 300,
                     z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7, 4.3),
                     size = 5, nrow = 2, lwd = 5,
                     linejoin = "round")

tg3 <- tileglyphGrob(x = 450, y = 450,
                     z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7, 4.3),
                     size = 5, nrow = 2, lwd = 5,
                     linejoin = "bevel")


grid::grid.newpage()
grid::grid.draw(tg1)
grid::grid.draw(tg2)
grid::grid.draw(tg3)
```

Index