Package 'ggmultiglyph'

August 8, 2026

Title Multivariate Data Visualization using Glyphs

Version 0.1.0

Description Provides 'ggplot2' geoms for visualizing multivariate data using glyphs. The package implements several established glyph designs described in the information visualization literature, including the review by Borgo et al. (2013) <doi:10.2312/conf/EG2013/stars/039-063>.

Copyright 2021-2026, ICAR-NBPGR

License GPL-2 | GPL-3

Encoding UTF-8

RdMacros Rdpack

LinkingTo Rcpp

Depends R (>= 3.5.0)

Imports cli, ggplot2, grid, methods, packcircles, Rcpp, Rdpack, rlang, scales

Suggests ggrepel, gridExtra, RColorBrewer

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

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

Language en-GB

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

NeedsCompilation yes

Author J. Aravind [aut, cre] (ORCID: <https://orcid.org/0000-0002-4791-442X>), Kamil Slowikowski [ctb] (ORCID: <https://orcid.org/0000-0002-2843-6370>, Repel algorithm), ICAR-NBGPR [cph] (url: https://nbpgr.org.in/)

Maintainer J. Aravind <j.aravind@icar.org.in>

Repository CRAN

Date/Publication 2026-08-08 13:50:03 UTC

Contents

`addlabel.glyphGrob` *Add Labels to a* `glyphGrob`

Description

Add labels to a `glyphGrob` as a `textGrob`. Useful in creation of custom guides for glyphs using `guide_custom`.

Usage

```
addlabel.glyphGrob(
  grob,
  label,
  segment = TRUE,
  segment.control = ggmultiglyph.segment.control(),
  push = 10,
  angle = 0,
  hjust = 0.5,
  vjust = 0.5
)
```

Arguments

grob	A glyphGrob object.
label	The labels to be plotted as a character vector. Should be equal in length to the dimensions of grob glyphGrob object. Labels are plotted from 0 to 1 (left to right or bottom to top for linear labelling and right to left in clock-wise direction for radial labelling.)
segment	logical. If TRUE draws line segments connecting labels and the glyphGrob Default is TRUE.
segment.control	A list of control settings for the line segments. Ignored if segment = FALSE. See ggmultiglyph.segment.control for details on the various control parameters.
push	A numeric value indicating how far the labels have to be pushed out from the glyphGrob.
angle	Text angle (in $[0, 360]$).
hjust	Horizontal justification (in $[0, 1]$).
vjust	Vertical justification (in $[0, 1]$).

Value

A grob object.

Examples

```
library(ggmultiglyph)
library(grid)
library(gridExtra)

label <- c("hp", "drat", "wt", "qsec", "vs", "am")

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Add labels to metroglyphGrob
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

mglyph1 <- metroglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                          z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                          size = 20, circle.size = 2)

mglyph2 <- metroglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                          z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                          size = 20, circle.size = 5,
                          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")

mglyph3 <- metroglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                          z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                          size = 20, circle.size = 0,
                          angle.start = 0, angle.stop = base::pi)
```

```
mglyph1_lab <- addlabel.glyphGrob(grob = mglyph1, label = label,
                                  push = 1, segment = FALSE)
mglyph2_lab <- addlabel.glyphGrob(grob = mglyph2, label = label,
                                  push = 3)
mglyph3_lab <- addlabel.glyphGrob(grob = mglyph3, label = label,
                                  push = 1, segment = FALSE)

grid.arrange(mglyph1, mglyph1_lab)
grid.arrange(mglyph2, mglyph2_lab)
grid.arrange(mglyph3, mglyph3_lab, nrow = 1)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Add labels to starglyphGrob
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

sg1 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 25)

sg2 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     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 = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     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")

sg1_lab <- addlabel.glyphGrob(grob = sg1, label = label,
                                  push = 1, segment = FALSE)
sg2_lab <- addlabel.glyphGrob(grob = sg2, label = label,
                                  push = 3)
sg3_lab <- addlabel.glyphGrob(grob = sg3, label = label,
                                  push = 1, segment = FALSE)

grid.arrange(sg1, sg1_lab)
grid.arrange(sg2, sg2_lab)
grid.arrange(sg3, sg3_lab, nrow = 1)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Add labels to pieglyphGrob
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

pg1 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                    size = 20)

pg2 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
```

```
                    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)

pg3 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    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"))

pg1_lab <- addlabel.glyphGrob(grob = pg1, label = label,
                              push = 2, segment = FALSE)
pg2_lab <- addlabel.glyphGrob(grob = pg2, label = label,
                              push = 5)
pg3_lab <- addlabel.glyphGrob(grob = pg3, label = label,
                              push = 5, segment = FALSE)

grid.arrange(pg1, pg1_lab)
grid.arrange(pg2, pg2_lab)
grid.arrange(pg3, pg3_lab)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Add labels to profileglyphGrob
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

bg1 <- profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                        size = 20)

bg2 <- profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                        size = 20, line = FALSE,
                        col.bar = "cyan")

bg3 <- profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                        size = 20, line = TRUE, bar = FALSE,
                        fill = "green")

bg1_lab <- addlabel.glyphGrob(grob = bg1, label = label, segment = TRUE,
                              angle = 45, push = 1, hjust = 1)
bg2_lab <- addlabel.glyphGrob(grob = bg2, label = label, segment = TRUE,
                              angle = 45, push = 5, hjust = 1)
bg3_lab <- addlabel.glyphGrob(grob = bg3, label = label, segment = TRUE,
                              angle = 45, push = -35, hjust = 0)

grid.arrange(bg1, bg1_lab, nrow = 1)
grid.arrange(bg2, bg2_lab, nrow = 1)
grid.arrange(bg3, bg3_lab, nrow = 1)

bg1 <- profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                        size = 20,
                        mirror = FALSE)
```

```
bg2 <- profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                        size = 20, line = FALSE, mirror = FALSE,
                        col.bar = "cyan")

bg3 <- profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                        size = 20, line = TRUE, bar = FALSE,
                        mirror = FALSE, fill = "green")

bg1_lab <- addlabel.glyphGrob(grob = bg1, label = label, segment = TRUE,
                              angle = 45, push = 1, hjust = 1)
bg2_lab <- addlabel.glyphGrob(grob = bg2, label = label, segment = TRUE,
                              angle = 45, push = 5, hjust = 1)
bg3_lab <- addlabel.glyphGrob(grob = bg3, label = label, segment = TRUE,
                              angle = 45, push = -25, hjust = 0)

grid.arrange(bg1, bg1_lab, nrow = 1)
grid.arrange(bg2, bg2_lab, nrow = 1)
grid.arrange(bg3, bg3_lab, nrow = 1)

bg1 <- profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                        size = 20, flip.axes = TRUE,
                        fill = "salmon")

bg2 <- profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                        size = 20, line = FALSE,
                        flip.axes = TRUE,
                        col.bar = "cyan")

bg3 <- profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                        size = 20, line = TRUE, bar = FALSE,
                        flip.axes = TRUE)

bg1_lab <- addlabel.glyphGrob(grob = bg1, label = label, segment = TRUE,
                              angle = 45, push = 15, hjust = 1)
bg2_lab <- addlabel.glyphGrob(grob = bg2, label = label, segment = TRUE,
                              push = 20, hjust = 1)
bg3_lab <- addlabel.glyphGrob(grob = bg3, label = label, segment = TRUE,
                              angle = 45, push = -30, hjust = 0)

grid.arrange(bg1, bg1_lab, nrow = 1)
grid.arrange(bg2, bg2_lab, nrow = 1)
grid.arrange(bg3, bg3_lab, nrow = 1)

bg1 <- profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                        size = 20, flip.axes = TRUE,
                        fill = "salmon", mirror = FALSE)
```

```
bg2 <- profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                        size = 20, line = FALSE, mirror = FALSE,
                        flip.axes = TRUE,
                        col.bar = "cyan")

bg3 <- profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                        size = 20, line = TRUE, bar = FALSE,
                        flip.axes = TRUE,
                        mirror = FALSE)

bg1_lab <- addlabel.glyphGrob(grob = bg1, label = label, segment = TRUE,
                              angle = 45, push = 5, hjust = 1)
bg2_lab <- addlabel.glyphGrob(grob = bg2, label = label, segment = TRUE,
                              push = 5, hjust = 1)
bg3_lab <- addlabel.glyphGrob(grob = bg3, label = label, segment = TRUE,
                              angle = 45, push = -25, hjust = 0)

grid.arrange(bg1, bg1_lab, nrow = 1)
grid.arrange(bg2, bg2_lab, nrow = 1)
grid.arrange(bg3, bg3_lab, nrow = 1)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Add labels to dotglyphGrob
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

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 = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2)

dg2 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, mirror = TRUE,
                    fill = "salmon", col = "black")

dg3 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    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 = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, mirror = TRUE,
                    flip.axes = TRUE,
                    fill = "green", col = "grey")
```

```
dg1_lab <- addlabel.glyphGrob(grob = dg1, label = label, segment = FALSE,
                              push = 3, hjust = 1, angle = 45)
dg2_lab <- addlabel.glyphGrob(grob = dg2, label = label, segment = TRUE,
                              angle = 45, push = 20, hjust = 1)
dg3_lab <- addlabel.glyphGrob(grob = dg3, label = label, segment = FALSE,
                              push = 3, hjust = 1)
dg4_lab <- addlabel.glyphGrob(grob = dg4, label = label, segment = TRUE,
                              push = 20,  hjust = 1)

grid.arrange(dg1, dg1_lab, nrow = 1)
grid.arrange(dg2, dg2_lab, nrow = 1)
grid.arrange(dg3, dg3_lab, nrow = 1)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Add labels to tileglyphGrob
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

label = c("hp", "drat", "wt", "qsec", "vs", "am", "gear")

tg1 <- tileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7, 4.3),
                     size = 5,
                     fill = RColorBrewer::brewer.pal(7, "Dark2"))

tg2 <- tileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7, 4.3),
                     size = 5, nrow = 7,
                     fill = RColorBrewer::brewer.pal(7, "Dark2"))

tg3 <- tileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     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"))

tg4 <- tileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7, 4.3),
                     size = 5, nrow = 3,
                     fill = RColorBrewer::brewer.pal(7, "Dark2"))

tg1_lab <- addlabel.glyphGrob(grob = tg1, label = label, segment = FALSE,
                              angle = 45, push = -10, hjust = 1, vjust = 0.5)
tg2_lab <- addlabel.glyphGrob(grob = tg2, label = label, segment = FALSE,
                              push = -1, hjust = 0, vjust = 0.5)
tg3_lab <- addlabel.glyphGrob(grob = tg3, label = label)
tg4_lab <- addlabel.glyphGrob(grob = tg4, label = label, segment = TRUE,
                              push = 5, hjust = 0, vjust = 0.25)

grid.arrange(tg1, tg2, tg3, tg4,
             tg1_lab, tg2_lab, tg3_lab, tg4_lab,
             nrow = 2, ncol = 4)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Add labels to flowerglyphGrob
```

```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

label <- c("hp", "drat", "wt", "qsec", "vs", "am")

fg1 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z = c(0.54, 0.75, 0.8, 1.4, 0.85, 1.1),
                       petal.base.shape = 2, petal.width = 0.35,
                       size = 18)

fg2 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z = c(0.54, 0.75, 0.8, 1.4, 0.85, 1.1),
                       petal.base.shape = 2, petal.width = 0.35,
                       size = 18,
                       lwd = 2, col = RColorBrewer::brewer.pal(6, "Dark2"))

fg1_lab <- addlabel.glyphGrob(grob = fg1, label = label,
                              push = 2, segment = FALSE)

fg2_lab <- addlabel.glyphGrob(grob = fg2, label = label,
                              push = 3)

grid.arrange(fg1, fg1_lab)
grid.arrange(fg2, fg2_lab)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Add labels to bubbleglyphGrob
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

label <- c("hp", "drat", "wt", "qsec", "vs", "am")

bg1 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33) * 0.75,
                       size = 8, layout = "circle",
                       connector = "background")

bg2 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                       size = 8, layout = "pack",
                       fill = RColorBrewer::brewer.pal(6, "Dark2"))

bg1_lab <- addlabel.glyphGrob(grob = bg1, label = label,
                              push = 10, segment = FALSE)

bg2_lab <- addlabel.glyphGrob(grob = bg2, label = label,
                              push = 15)

grid.arrange(bg1, bg1_lab)
grid.arrange(bg2, bg2_lab)
```

---

bubbleglyphGrob *Draw a Bubble Glyph*

Description

Uses Grid graphics to draw a bubble glyph.

Usage

```
bubbleglyphGrob(
  x = 0.5,
  y = 0.5,
  z,
  size = 1,
  scale.radius = TRUE,
  scale.area = FALSE,
  bubble.layout = c("circle", "line", "chain", "hub", "pack", "annulus"),
  connector = c("none", "foreground", "background"),
  line.angle = 0,
  angle.start = 0,
  angle.stop = 2 * base::pi,
  col = "black",
  fill = NA,
  alpha = 1,
  lwd.connector = 1,
  col.connector = "black",
  connector.point = FALSE,
  connector.point.size = 0.01,
  col.connector.point = "black",
  preserve.zeros = TRUE,
  lwd = 1,
  linejoin = c("mitre", "round", "bevel"),
  draw.grid = FALSE,
  grid.levels = NULL,
  col.grid = "grey",
  lwd.grid = lwd,
  layout.eps = NULL
)
```

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 radius/area of bubbles.
size	The size of bubbles.
scale.radius	logical. If TRUE, the values in z are used directly as the bubble radii.
scale.area	logical. If TRUE, values in z are treated as bubble areas, and radii are derived as $\sqrt{z}$, so that bubble area (rather than radius) is proportional to z.
bubble.layout	The layout algorithm used to position the bubbles. One of "circle", "line", "chain", "hub", "pack" or "annulus" (See **Layouts** for more details).

`connector` The style used to connect bubbles to one another. One of `"none"` (no connectors are drawn), `"foreground"` (connectors are drawn on top of the bubbles) or `"background"` (connectors are drawn underneath the bubbles).

`line.angle` The angle (in radians) at which bubbles are arranged when `layout = "line"`.

`angle.start` The start angle for the bubbles in radians for `layout = "circle"` and `layout = "hub"`. Default is zero.

`angle.stop` The stop angle for the bubbles in radians for `layout = "circle"` and `layout = "hub"`. Default is 2π.

`col` The bubble outline colour.

`fill` The bubble fill colour.

`alpha` The alpha transparency value.

`lwd.connector` The line width of the connectors.

`col.connector` The colour of the connectors.

`connector.point`
logical. If `TRUE`, a small point is drawn at the centre of each bubble where connectors meet. Default is `FALSE`.

`connector.point.size`
The size of the connector points as a fraction of native coordinates. Default is `0.01`.

`col.connector.point`
The colour of the connector points. Default is `"black"`.

`preserve.zeros` logical. If `TRUE`, bubbles with zero values in `z` are still displayed with a minimum radius to ensure visibility in the layout computation. If `FALSE`, zero values result in no visible bubble. Default is `TRUE`.

`lwd` The bubble outline line width.

`linejoin` The line join style for the tile polygon. Either `"mitre"`, `"round"` or `"bevel"`.

`draw.grid` logical. If `TRUE`, grid levels are plotted as nested circles within each bubble. Default is `FALSE`.

`grid.levels` A list of grid levels (as vectors) corresponding to the values in `z` at which the nested circles are to be plotted. The values in `z` should be present in the list specified.

`col.grid` The colour of the grid circles.

`lwd.grid` The line width of the grid circles.

`layout.eps` A small numerical tolerance value used in layout computations to avoid numerical instability. If `NULL` (default), a value is automatically computed based on the largest bubble radius.

Value

A `gTree` object.

Layouts

The following layouts are available.

`"circle"` Bubbles are arranged around the circumference of an invisible circle, evenly spaced by angle between `angle.start` and `angle.stop`. The radius of this invisible circle is derived from the bubble radii (the largest bubble radius plus the mean bubble radius, with a small buffer added) so that bubbles do not overlap regardless of their individual sizes. This is the default layout.

`"line"` Bubbles are arranged side by side along a straight line passing through the glyph centre, each bubble touching its neighbours. The line may be rotated to any orientation using the `line.angle` argument (in radians).

`"chain"` Bubbles are linked together in a meandering tangent chain. Each new bubble is placed tangent to its immediate predecessor, alternating between opposite sides of the chain to produce a distinctive zig-zag arrangement. Where necessary, the placement is adjusted to minimise overlap with previously placed bubbles while preserving the ordering of the data.

`"hub"` One bubble (the one with the largest radius) is treated as a central "hub", and all other bubbles are placed around it, each touching the hub bubble, at angles spaced between `angle.start` and `angle.stop`. This is a quick, deterministic layout: because the surrounding bubbles are not tested against one another for overlap, bubbles can overlap when several of them are large relative to the hub.

`"pack"` Bubbles are arranged using a circle-packing algorithm (`circleProgressiveLayout`), which places bubbles as close together as possible without any overlap (Collins and Stephenson 2003; Wang et al. 2006; Bedward et al. 2024). This produces the most compact, space-efficient arrangement of the available layouts, at the cost of the arrangement having no inherent order or direction (bubbles are not ordered along an axis or around a hub).

`"annulus"` Bubbles are arranged, in the order supplied, around the circumference of a ring (annulus), with each bubble touching its immediate neighbours on either side (the last bubble touching the first, closing the ring). The radius of the ring is solved numerically so that all adjacent bubbles are mutually tangent.

References

Bedward M, Eppstein D, Menzel P (2024). "packcircles: Circle Packing. R package version 0.3.7."

Collins CR, Stephenson K (2003). "A circle packing algorithm." *Computational Geometry*, **25**(3), 233–256.

Wang W, Wang H, Dai G, Wang H (2006). "Visualization of large hierarchical data by circle packing." In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 517–520. ISBN 978-1-59593-372-0.

See Also

`geom_bubbleglyph`, `circleProgressiveLayout`

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

Examples

```
library(ggmultiglyph)
library(grid)
library(gridExtra)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust layout algorithm
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

bg1 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                       size = 5,
                       layout = "circle")

bg2 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                       size = 5,
                       layout = "line")

bg3 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                       size = 5,
                       layout = "chain")

bg4 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                       size = 5,
                       layout = "hub")

bg5 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                       size = 5,
                       layout = "annulus")

bg6 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                       size = 5,
                       layout = "pack")

grid.arrange(bg1, bg2, bg3, bg4, bg5, bg6, nrow = 2)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Add connectors to layout algorithms
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

bg1 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                       size = 5, connector = "foreground",
                       layout = "circle")

bg2 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
```

```
                        size = 5, connector = "foreground",
                        layout = "line")

bg3 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                        size = 5, connector = "foreground",
                        layout = "chain")

bg4 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                        size = 5, connector = "foreground",
                        layout = "hub")

bg5 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                        size = 5, connector = "foreground",
                        layout = "annulus")

bg6 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                        size = 5, connector = "foreground",
                        layout = "pack")

grid.arrange(bg1, bg2, bg3, bg4, bg5, bg6, nrow = 2)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust connector layer position
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

bg1 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                        size = 5,
                        layout = "circle", fill = "gray80")

bg2 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                        size = 5, connector = "foreground",
                        layout = "circle", fill = "gray80")

bg3 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                        size = 5, connector = "background",
                        layout = "circle", fill = "gray80")

bg4 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                        size = 5,
                        layout = "hub", fill = "gray80")

bg5 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                        size = 5, connector = "foreground",
                        layout = "hub", fill = "gray80")
```

```
bg6 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                       size = 5, connector = "background",
                       layout = "hub", fill = "gray80")

grid.arrange(bg1, bg2, bg3, bg4, bg5, bg6, nrow = 2)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust line width
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

bg1 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                       size = 5,
                       layout = "circle")

bg2 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                       size = 5,
                       layout = "circle", lwd = 3)

bg3 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                       size = 5,
                       layout = "circle", lwd = 5)

bg4 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                       size = 5, connector = "foreground",
                       layout = "circle")

bg5 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                       size = 5, connector = "foreground",
                       layout = "circle", lwd.connector = 3)

bg6 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                       size = 5, connector = "foreground",
                       layout = "circle", lwd.connector = 5)

grid.arrange(bg1, bg2, bg3, bg4, bg5, bg6, nrow = 2)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust line join style
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

bg1 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                       size = 5, connector = "foreground",
                       layout = "chain",
                       lwd.connector = 10)
```

```
bg2 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                       size = 5, connector = "foreground",
                       layout = "chain",
                       lwd.connector = 10, linejoin = "bevel")

bg3 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                       size = 5, connector = "foreground",
                       layout = "chain",
                       lwd.connector = 10, linejoin = "round")

bg4 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                       size = 5, connector = "foreground",
                       layout = "hub",
                       lwd.connector = 10)

bg5 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                       size = 5, connector = "foreground",
                       layout = "hub",
                       lwd.connector = 10, linejoin = "bevel")

bg6 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                       size = 5, connector = "foreground",
                       layout = "hub",
                       lwd.connector = 10, linejoin = "round")

grid.arrange(bg1, bg2, bg3, bg4, bg5, bg6, nrow = 2)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust angle for circle and hub layouts
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

bg1 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                       size = 5, connector = "foreground",
                       layout = "circle")

bg2 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                       size = 5, connector = "foreground",
                       layout = "circle",
                       angle.start = 2 * base::pi, angle.stop = 0)

bg3 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                       size = 5, connector = "foreground",
                       layout = "circle",
                       angle.start = 90 * (base::pi/180),
```

```
                        angle.stop = 450 * (base::pi/180))

bg4 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                        size = 5, connector = "foreground",
                        layout = "hub")

bg5 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                        size = 5, connector = "foreground",
                        layout = "hub",
                        angle.start = 2 * base::pi, angle.stop = 0)

bg6 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                        size = 5, connector = "foreground",
                        layout = "hub",
                        angle.start = 90 * (base::pi/180),
                        angle.stop = 450 * (base::pi/180))

grid.arrange(bg1, bg2, bg3, bg4, bg5, bg6, nrow = 2)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust angle for line layout
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

bg1 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                        size = 5, connector = "foreground",
                        layout = "line")

bg2 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                        size = 5, connector = "foreground",
                        layout = "line", line.angle = base::pi/4)

bg3 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                        size = 5, connector = "foreground",
                        layout = "line", line.angle = base::pi/2)

grid.arrange(bg1, bg2, bg3, nrow = 1)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust fill colour
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

bg1 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                        size = 5, fill = "salmon",
                        layout = "circle")

bg2 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
```

```
                        z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                        size = 5, fill = "cyan",
                        layout = "circle")

bg3 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                        size = 5, fill = "green",
                        layout = "circle")

bg4 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                        size = 5, fill = "salmon",
                        layout = "line")

bg5 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                        size = 5, fill = "cyan",
                        layout = "line")

bg6 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                        size = 5, fill = "green",
                        layout = "line")

grid.arrange(bg1, bg2, bg3, bg4, bg5, bg6, nrow = 2)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust line colour
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

bg1 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                        size = 5, col = "salmon", lwd = 3,
                        layout = "circle")

bg2 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                        size = 5, col = "cyan", lwd = 3,
                        layout = "circle")

bg3 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                        size = 5, col = "green", lwd = 3,
                        layout = "circle")

bg4 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                        size = 5, col = "salmon", lwd = 3,
                        layout = "line")

bg5 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                        size = 5, col = "cyan", lwd = 3,
```

```
                        layout = "line")

bg6 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                        size = 5, col = "green", lwd = 3,
                        layout = "line")

grid.arrange(bg1, bg2, bg3, bg4, bg5, bg6, nrow = 2)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust connector colour
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

bg1 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                        size = 5, connector = "foreground",
                        col.connector = "salmon", lwd.connector = 3,
                        layout = "circle")

bg2 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                        size = 5, connector = "foreground",
                        col.connector = "cyan", lwd.connector = 3,
                        layout = "circle")

bg3 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                        size = 5, connector = "foreground",
                        col.connector = "green", lwd.connector = 3,
                        layout = "circle")

bg4 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                        size = 5, connector = "foreground",
                        col.connector = "salmon", lwd.connector = 3,
                        layout = "annulus")

bg5 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                        size = 5, connector = "foreground",
                        col.connector = "cyan", lwd.connector = 3,
                        layout = "annulus")

bg6 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                        size = 5, connector = "foreground",
                        col.connector = "green", lwd.connector = 3,
                        layout = "annulus")

grid.arrange(bg1, bg2, bg3, bg4, bg5, bg6, nrow = 2)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Multivariate bubble fill and colour
```

```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

bg1 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                       size = 5,
                       fill = RColorBrewer::brewer.pal(6, "Dark2"),
                       layout = "circle")

bg2 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                       size = 5,
                       fill = RColorBrewer::brewer.pal(6, "Dark2"),
                       layout = "line")

bg3 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                       size = 5,
                       fill = RColorBrewer::brewer.pal(6, "Dark2"),
                       layout = "chain")

bg4 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                       size = 5,
                       fill = RColorBrewer::brewer.pal(6, "Dark2"),
                       layout = "hub")

bg5 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                       size = 5,
                       fill = RColorBrewer::brewer.pal(6, "Dark2"),
                       layout = "annulus")

bg6 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                       size = 5,
                       fill = RColorBrewer::brewer.pal(6, "Dark2"),
                       layout = "pack")

grid.arrange(bg1, bg2, bg3, bg4, bg5, bg6, nrow = 2)

bg1 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                       size = 5, lwd = 3,
                       col = RColorBrewer::brewer.pal(6, "Dark2"),
                       layout = "circle")

bg2 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                       size = 5, lwd = 3,
                       col = RColorBrewer::brewer.pal(6, "Dark2"),
                       layout = "line")

bg3 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
```

```
                        z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                        size = 5, lwd = 3,
                        col = RColorBrewer::brewer.pal(6, "Dark2"),
                        layout = "chain")

bg4 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                        size = 5, lwd = 3,
                        col = RColorBrewer::brewer.pal(6, "Dark2"),
                        layout = "hub")

bg5 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                        size = 5, lwd = 3,
                        col = RColorBrewer::brewer.pal(6, "Dark2"),
                        layout = "annulus")

bg6 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7),
                        size = 5, lwd = 3,
                        col = RColorBrewer::brewer.pal(6, "Dark2"),
                        layout = "pack")

grid.arrange(bg1, bg2, bg3, bg4, bg5, bg6, nrow = 2)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust grid levels
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

gl <- split(x = c(rep(c(1, 2, 3), 4),
                  rep(c(1, 2, 3, 4), 2)),
            f = c(rep(1:4, each = 3),
                  rep(5:6, each = 4)))

gl

bg1 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = c(1, 3, 2, 1, 2, 3),
                        size = 1,
                        layout = "circle",
                        draw.grid = TRUE, grid.levels = gl)

bg2 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = c(1, 3, 2, 1, 2, 3),
                        size = 1,
                        layout = "line",
                        draw.grid = TRUE, grid.levels = gl)

bg3 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = c(1, 3, 2, 1, 2, 3),
                        size = 1,
                        layout = "chain",
                        draw.grid = TRUE, grid.levels = gl)
```

```
bg4 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z = c(1, 3, 2, 1, 2, 3),
                       size = 1,
                       layout = "hub",
                       draw.grid = TRUE, grid.levels = gl)

bg5 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z = c(1, 3, 2, 1, 2, 3),
                       size = 1,
                       layout = "annulus",
                       draw.grid = TRUE, grid.levels = gl)

bg6 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z = c(1, 3, 2, 1, 2, 3),
                       size = 1,
                       layout = "pack",
                       draw.grid = TRUE, grid.levels = gl)

grid.arrange(bg1, bg2, bg3, bg4, bg5, bg6, nrow = 2)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust grid level colours
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

gl <- split(x = c(rep(c(1, 2, 3), 4),
                  rep(c(1, 2, 3, 4), 2)),
            f = c(rep(1:4, each = 3),
                  rep(5:6, each = 4)))

gl

bg1 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z = c(1, 3, 2, 1, 2, 3),
                       size = 1,
                       layout = "circle",
                       draw.grid = TRUE, grid.levels = gl,
                       col = "white", col.grid = "white",
                       fill = RColorBrewer::brewer.pal(6, "Dark2"))

bg2 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z = c(1, 3, 2, 1, 2, 3),
                       size = 1,
                       layout = "line",
                       draw.grid = TRUE, grid.levels = gl,
                       col = "white", col.grid = "white",
                       fill = RColorBrewer::brewer.pal(6, "Dark2"))

bg3 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z = c(1, 3, 2, 1, 2, 3),
                       size = 1,
                       layout = "chain",
                       draw.grid = TRUE, grid.levels = gl,
```

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

bg4 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = c(1, 3, 2, 1, 2, 3),
                        size = 1,
                        layout = "hub",
                        draw.grid = TRUE, grid.levels = gl,
                        col = "white", col.grid = "white",
                        fill = RColorBrewer::brewer.pal(6, "Dark2"))

bg5 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = c(1, 3, 2, 1, 2, 3),
                        size = 1,
                        layout = "annulus",
                        draw.grid = TRUE, grid.levels = gl,
                        col = "white", col.grid = "white",
                        fill = RColorBrewer::brewer.pal(6, "Dark2"))

bg6 <- bubbleglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = c(1, 3, 2, 1, 2, 3),
                        size = 1,
                        layout = "pack",
                        draw.grid = TRUE, grid.levels = gl,
                        col = "white", col.grid = "white",
                        fill = RColorBrewer::brewer.pal(6, "Dark2"))

grid.arrange(bg1, bg2, bg3, bg4, bg5, bg6, nrow = 2)
```

---

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: bubbleglyphGrob(), flowerglyphGrob(), metroglyphGrob(), pieglyphGrob(), profileglyphGrob(), starglyphGrob(), tileglyphGrob()

Examples

```
library(ggmultiglyph)
library(grid)
library(gridExtra)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Mirror and flip.axes combination
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

dg1 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2)

dg2 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
```

```
                    radius = 2, mirror = TRUE)

dg3 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, flip.axes = TRUE)

dg4 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, mirror = TRUE,
                    flip.axes = TRUE)

grid.arrange(dg1, dg2, nrow = 1)
grid.arrange(dg3, dg4, nrow = 1)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust radius
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

dg1 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 0.5)

dg2 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2)

dg3 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 3)

dg4 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 0.5, mirror = TRUE)

dg5 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, mirror = TRUE)

dg6 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 3, mirror = TRUE)

grid.arrange(dg1, dg2, dg3, nrow = 1)
grid.arrange(dg4, dg5, dg6, nrow = 1)

dg1 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 0.5, flip.axes = TRUE)

dg2 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, flip.axes = TRUE)
```

```
dg3 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 3, flip.axes = TRUE)

dg4 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 0.5, mirror = TRUE,
                    flip.axes = TRUE)

dg5 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, mirror = TRUE,
                    flip.axes = TRUE)

dg6 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 3, mirror = TRUE,
                    flip.axes = TRUE)

grid.arrange(dg1, dg2, dg3, nrow = 1, ncol = 4)
grid.arrange(dg4, dg5, dg6, nrow = 1)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust dot line width
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

dg1 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2)

dg2 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, lwd = 3)

dg3 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, lwd = 5)

dg4 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, mirror = TRUE)

dg5 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, mirror = TRUE, lwd = 3)

dg6 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, mirror = TRUE, lwd = 5)

grid.arrange(dg1, dg2, dg3, nrow = 1)
grid.arrange(dg4, dg5, dg6, nrow = 1)
```

```
dg1 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, flip.axes = TRUE)

dg2 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, flip.axes = TRUE, lwd = 3)

dg3 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, flip.axes = TRUE, lwd = 5)

dg4 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, mirror = TRUE,
                    flip.axes = TRUE)

dg5 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, mirror = TRUE,
                    flip.axes = TRUE, lwd = 3)

dg6 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, mirror = TRUE,
                    flip.axes = TRUE, lwd = 5)

grid.arrange(dg1, dg2, dg3, nrow = 1, ncol = 4)
grid.arrange(dg4, dg5, dg6, nrow = 1)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust dot fill
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

dg1 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, fill = "salmon")

dg2 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, fill = "cyan")

dg3 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, fill = "green")

dg4 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, mirror = TRUE, fill = "salmon")

dg5 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, mirror = TRUE, fill = "cyan")
```

```
dg6 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, mirror = TRUE, fill = "green")

grid.arrange(dg1, dg2, dg3, nrow = 1)
grid.arrange(dg4, dg5, dg6, nrow = 1)

dg1 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, flip.axes = TRUE, fill = "salmon")

dg2 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, flip.axes = TRUE, fill = "cyan")

dg3 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, flip.axes = TRUE, fill = "green")

dg4 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, mirror = TRUE,
                    flip.axes = TRUE, fill = "salmon")

dg5 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, mirror = TRUE,
                    flip.axes = TRUE, fill = "cyan")

dg6 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, mirror = TRUE,
                    flip.axes = TRUE, fill = "green")

grid.arrange(dg1, dg2, dg3, nrow = 1, ncol = 4)
grid.arrange(dg4, dg5, dg6, nrow = 1)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust dot line colour
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

dg1 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, col = "salmon")

dg2 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, col = "cyan")

dg3 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, col = "green")
```

```
dg4 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, mirror = TRUE, col = "salmon")

dg5 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, mirror = TRUE, col = "cyan")

dg6 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, mirror = TRUE, col = "green")

grid.arrange(dg1, dg2, dg3, nrow = 1)
grid.arrange(dg4, dg5, dg6, nrow = 1)

dg1 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, flip.axes = TRUE, col = "salmon")

dg2 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, flip.axes = TRUE, col = "cyan")

dg3 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, flip.axes = TRUE, col = "green")

dg4 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, mirror = TRUE,
                    flip.axes = TRUE, col = "salmon")

dg5 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, mirror = TRUE,
                    flip.axes = TRUE, col = "cyan")

dg6 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, mirror = TRUE,
                    flip.axes = TRUE, col = "green")

grid.arrange(dg1, dg2, dg3, nrow = 1, ncol = 4)
grid.arrange(dg4, dg5, dg6, nrow = 1)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Multivariate fill colour
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

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 = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, fill = clrs,
                    col = "black")

dg2 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, fill = clrs,
                    col = "white")

dg3 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, mirror = TRUE, fill = clrs,
                    col = "black")

dg4 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, mirror = TRUE, fill = clrs,
                    col = "white")

grid.arrange(dg1, dg2, nrow = 1)
grid.arrange(dg3, dg4, nrow = 1)

dg1 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, flip.axes = TRUE, fill = clrs,
                    col = "black")

dg2 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, flip.axes = TRUE, fill = clrs,
                    col = "white")

dg3 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, mirror = TRUE,
                    flip.axes = TRUE, fill = clrs,
                    col = "black")

dg4 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, mirror = TRUE,
                    flip.axes = TRUE, fill = clrs,
                    col = "white")

grid.arrange(dg1, dg2, nrow = 1)
grid.arrange(dg3, dg4, nrow = 1)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Multivariate dot line colour
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
dg1 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, fill = "black", lwd = 5,
                    col = clrs)

dg2 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, fill = "white", lwd = 5,
                    col = clrs)

dg3 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, mirror = TRUE,
                    fill = "black", lwd = 5,
                    col = clrs)

dg4 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, mirror = TRUE,
                    fill = "white", lwd = 5,
                    col = clrs)

grid.arrange(dg1, dg2, nrow = 1)
grid.arrange(dg3, dg4, nrow = 1)

dg1 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, flip.axes = TRUE,
                    fill = "black", lwd = 5,
                    col = clrs)

dg2 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, flip.axes = TRUE,
                    fill = "white", lwd = 5,
                    col = clrs)

dg3 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, mirror = TRUE,
                    flip.axes = TRUE,
                    fill = "black", lwd = 5,
                    col = clrs)

dg4 <- dotglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    radius = 2, mirror = TRUE,
                    flip.axes = TRUE,
                    fill = "white", lwd = 5,
                    col = clrs)

grid.arrange(dg1, dg2, nrow = 1)
```

```
grid.arrange(dg3, dg4, nrow = 1)
```

`flowerglyphGrob` *Draw a Flower Glyph*

Description

Uses Grid graphics to draw a flower glyph (Van Onzenoodt et al. 2023). Each variable in `z` is depicted as a "petal" radiating from a central point, with the petal length scaled according to the corresponding value in `z`. The silhouette of each petal is generated from a continuous parametric taper profile whose widening from the base and tapering towards the tip are controlled independently by `petal.base.shape` and `petal.tip.shape`. An optional notch at the tip, controlled by `petal.tip.notch`, can be used to produce cleft (bifid) or heart-shaped petals.

Usage

```
flowerglyphGrob(
  x = 0.5,
  y = 0.5,
  z,
  size = 1,
  petal.width = 0.6,
  petal.base.shape = 1,
  petal.tip.shape = 1,
  petal.tip.notch = 0,
  petal.tip.notch.width = 0.15,
  petal.base.notch = 0,
  petal.base.notch.width = 0.15,
  petal.waist = 0,
  petal.waist.position = 0.5,
  petal.waist.width = 0.15,
  petal.curvature = 0,
  petal.curvature.position = 0.7,
  petal.curvature.width = 0.2,
  edges = 30,
  col = "black",
  fill = NA,
  lwd = 1,
  alpha = 1,
  angle.start = 0,
  angle.stop = 2 * base::pi,
  linejoin = c("mitre", "round", "bevel"),
  scale.length = TRUE,
  scale.area = FALSE,
  centre = TRUE,
  centre.size = 1,
```

```
  col.centre = "black",
  fill.centre = "black",
  draw.grid = FALSE,
  grid.levels = NULL,
  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 the length of the flower glyph petals.
size	The size of the glyph (scales both petal length and width).
petal.width	The width of a petal (as a proportion of `size`) at its widest point.
petal.base.shape	A positive numeric value controlling how rapidly the petal widens from its base. Smaller values produce a broader attachment to the centre of the glyph, whereas larger values delay the expansion of the petal, producing a narrower, more clawed base.
petal.tip.shape	A positive numeric value controlling how rapidly the petal tapers towards its tip. Smaller values produce broader, more rounded tips, whereas larger values concentrate the petal width closer to its middle, producing progressively narrower, more pointed or lanceolate tips.
petal.tip.notch	A numeric value controlling deformation of the petal tip. A value of `0` produces a smooth, unmodified tip. Positive values (typically > 0) create an increasingly deep inward cleft with smooth shoulders, producing retuse, emarginate, obcordate, or bifid petal tips. Negative values extend the tip beyond its nominal length, producing progressively more acuminate, cuspidate, or aristate tips.
petal.tip.notch.width	A positive numeric value controlling the breadth of the tip notch. Smaller values produce a narrow, sharply incised cleft, or point, whereas larger values produce a broader, more rounded indentation or extension.
petal.base.notch	A numeric value controlling deformation of the petal base. A value of `0` produces a smooth, unmodified base. Positive values shift the basal shoulders backward relative to the central notch vertex (which remains anchored at the origin), producing cordate, sagittate, reniform, hastate, or auriculate petal shapes. Negative values extend the base into a basal protrusion or claw.
petal.base.notch.width	A positive numeric value controlling the breadth of the basal notch or protrusion. Smaller values produce a narrow, sharply defined sinus or point, whereas larger values produce a broader, more rounded basal indentation or extension.

`petal.waist` A numeric value controlling the depth of a constriction along the petal. A value of `0` produces a uniformly tapered petal, whereas larger values produce an increasingly pronounced narrowing, resulting in pandurate, fiddle-shaped, or otherwise constricted petals.

`petal.waist.position` A numeric value between `0` and `1` controlling the position of the waist along the petal axis, where `0` corresponds to the base and `1` to the tip.

`petal.waist.width` A positive numeric value controlling the breadth of the waist. Smaller values produce a sharp, localized constriction, whereas larger values produce a broader, more gradual narrowing.

`petal.curvature` A numeric value or vector controlling the curvature of the petal centreline. A value of `0` produces a straight petal, whereas single positive or negative values produce simple arcuate or falcate (C-shaped) curvature in opposite directions. Passing a vector specifies multiple curvature bends along the petal axis; for example, a vector of length 2 with opposing signs (e.g., `c(3.0, -3.0)`) produces a sigmoid (S-shaped) petal.

`petal.curvature.position` A numeric value or vector between `0` and `1` controlling the position of maximum curvature along the petal axis, where `0` corresponds to the base and `1` to the tip. When `petal.curvature` is a vector specifying multiple bends (e.g., for sigmoid shapes), this parameter specifies the location along the axis for each corresponding bend.

`petal.curvature.width` A positive numeric value controlling the breadth of the region over which the petal bends. Smaller values concentrate the curvature into a localized bend, whereas larger values distribute the curvature more evenly along the petal, producing a broader, more gradual arc.

`edges` The number of points used to approximate the taper profile of one side of a petal. Higher values give smoother petal outlines.

`col` The petal outline colour.

`fill` The petal fill colour.

`lwd` The petal outline line width.

`alpha` The alpha transparency value.

`angle.start` The start angle for the petals in radians. Default is zero.

`angle.stop` The stop angle for the petals in radians. Default is 2π.

`linejoin` The line join style for the petal outlines. Either `"mitre"`, `"round"` or `"bevel"`.

`scale.length` logical. If `TRUE`, the petal lengths are scaled according to value of `z`.

`scale.area` logical. If `TRUE`, the area of the petals are scaled according to value of `z`.

`centre` logical. If `TRUE`, a central point is drawn at the glyph origin. Default is `TRUE`.

`centre.size` The size (radius, in mm) of the central point.

`col.centre` The line colour of the central point.

`fill.centre`	The fill colour of the central point.
`draw.grid`	logical. If `TRUE`, grid levels are plotted as nested petals. Default is `FALSE`.
`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.
`col.grid`	The colour of the nested grid petal lines.
`lwd.grid`	The line width of the nested grid petal lines.

Value

A `gTree` object.

References

Van Onzenoodt C, Vazquez P, Ropinski T (2023). "Out of the plane: Flower versus star glyphs to support high-dimensional exploration in two-dimensional embeddings." *IEEE Transactions on Visualization and Computer Graphics*, **29**(12), 5468–5482.

See Also

`geom_flowerglyph`

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

Examples

```
library(ggmultiglyph)
library(grid)
library(gridExtra)

z <- c(0.5, 0.8, 1.2, 0.6, 1, 0.7)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust length and area scaling
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

fg1 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z = z,
                       size = 20,
                       petal.base.shape = 1.25, petal.tip.shape = 1,
                       petal.width = 0.25,
                       scale.area = TRUE, scale.length = FALSE)

fg2 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z = z,
                       size = 20,
                       petal.base.shape = 1.25, petal.tip.shape = 1,
                       petal.width = 0.25,
                       scale.area = FALSE, scale.length = TRUE)

fg3 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
```

```
                        z = z,
                        size = 20,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25,
                        scale.area = FALSE, scale.length = FALSE)

grid.arrange(fg1, fg2, fg3, nrow = 1)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust flower size
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

fg1 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = z,
                        size = 10,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25,
                        scale.area = TRUE, scale.length = FALSE)

fg2 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = z,
                        size = 15,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25,
                        scale.area = TRUE, scale.length = FALSE)

fg3 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = z,
                        size = 20,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25,
                        scale.area = TRUE, scale.length = FALSE)

grid.arrange(fg1, fg2, fg3, nrow = 1)

fg1 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = z,
                        size = 10,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25,
                        scale.area = FALSE, scale.length = TRUE)

fg2 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = z,
                        size = 15,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25,
                        scale.area = FALSE, scale.length = TRUE)

fg3 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = z,
                        size = 20,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
```

```
                        petal.width = 0.25,
                        scale.area = FALSE, scale.length = TRUE)

grid.arrange(fg1, fg2, fg3, nrow = 1)

fg1 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = z,
                        size = 10,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25,
                        scale.area = FALSE, scale.length = FALSE)

fg2 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = z,
                        size = 15,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25,
                        scale.area = FALSE, scale.length = FALSE)

fg3 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = z,
                        size = 20,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25,
                        scale.area = FALSE, scale.length = FALSE)

grid.arrange(fg1, fg2, fg3, nrow = 1)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust circle size
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

fg1 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = z,
                        size = 20, centre.size = 1,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25,
                        scale.area = TRUE, scale.length = FALSE)

fg2 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = z,
                        size = 20, centre.size = 2.5,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25,
                        scale.area = TRUE, scale.length = FALSE)

fg3 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = z,
                        size = 20, centre.size = 5,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25,
                        scale.area = TRUE, scale.length = FALSE)
```

```
grid.arrange(fg1, fg2, fg3, nrow = 1)

fg1 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z = z,
                       size = 20, centre.size = 1,
                       petal.base.shape = 1.25, petal.tip.shape = 1,
                       petal.width = 0.25,
                       scale.area = FALSE, scale.length = TRUE)

fg2 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z = z,
                       size = 20, centre.size = 2.5,
                       petal.base.shape = 1.25, petal.tip.shape = 1,
                       petal.width = 0.25,
                       scale.area = FALSE, scale.length = TRUE)

fg3 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z = z,
                       size = 20, centre.size = 5,
                       petal.base.shape = 1.25, petal.tip.shape = 1,
                       petal.width = 0.25,
                       scale.area = FALSE, scale.length = TRUE)

grid.arrange(fg1, fg2, fg3, nrow = 1)

fg1 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z = z,
                       size = 20, centre.size = 1,
                       petal.base.shape = 1.25, petal.tip.shape = 1,
                       petal.width = 0.25,
                       scale.area = FALSE, scale.length = FALSE)

fg2 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z = z,
                       size = 20, centre.size = 2.5,
                       petal.base.shape = 1.25, petal.tip.shape = 1,
                       petal.width = 0.25,
                       scale.area = FALSE, scale.length = FALSE)

fg3 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z = z,
                       size = 20, centre.size = 5,
                       petal.base.shape = 1.25, petal.tip.shape = 1,
                       petal.width = 0.25,
                       scale.area = FALSE, scale.length = FALSE)

grid.arrange(fg1, fg2, fg3, nrow = 1)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust angle
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

fg1 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
```

```
                        z = z,
                        size = 20,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25,
                        scale.area = TRUE, scale.length = FALSE)

fg2 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = z,
                        size = 20,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25,
                        scale.area = TRUE, scale.length = FALSE,
                        angle.start = 0, angle.stop = base::pi)

fg3 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = z,
                        size = 20,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25,
                        scale.area = TRUE, scale.length = FALSE,
                        angle.start = 90 * (base::pi/180),
                        angle.stop = 270 * (base::pi/180))

grid.arrange(fg1, fg2, fg3, nrow = 1)

fg1 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = z,
                        size = 20, centre.size = 1,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25,
                        scale.area = FALSE, scale.length = TRUE)

fg2 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = z,
                        size = 20, centre.size = 1,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25,
                        scale.area = FALSE, scale.length = TRUE,
                        angle.start = 0, angle.stop = base::pi)

fg3 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = z,
                        size = 20, centre.size = 1,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25,
                        scale.area = FALSE, scale.length = TRUE,
                        angle.start = 90 * (base::pi/180),
                        angle.stop = 270 * (base::pi/180))

grid.arrange(fg1, fg2, fg3, nrow = 1)

fg1 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = z,
```

```
                        size = 20,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25,
                        scale.area = FALSE, scale.length = FALSE)

fg2 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = z,
                        size = 20,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25,
                        scale.area = FALSE, scale.length = FALSE,
                        angle.start = 0, angle.stop = base::pi)

fg3 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = z,
                        size = 20,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25,
                        scale.area = FALSE, scale.length = FALSE,
                        angle.start = 90 * (base::pi/180),
                        angle.stop = 270 * (base::pi/180))

grid.arrange(fg1, fg2, fg3, nrow = 1)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust line widths
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

fg1 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = z,
                        size = 20,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25,
                        scale.area = TRUE, scale.length = FALSE)

fg2 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = z,
                        size = 20,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25,
                        scale.area = TRUE, scale.length = FALSE,
                        lwd = 3)

fg3 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = z,
                        size = 20,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25,
                        scale.area = TRUE, scale.length = FALSE,
                        lwd = 5)

grid.arrange(fg1, fg2, fg3, nrow = 1)
```

```
fg1 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z = z,
                       size = 20,
                       petal.base.shape = 1.25, petal.tip.shape = 1,
                       petal.width = 0.25,
                       scale.area = FALSE, scale.length = TRUE)

fg2 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z = z,
                       size = 20,
                       petal.base.shape = 1.25, petal.tip.shape = 1,
                       petal.width = 0.25,
                       scale.area = FALSE, scale.length = TRUE,
                       lwd = 3)

fg3 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z = z,
                       size = 20,
                       petal.base.shape = 1.25, petal.tip.shape = 1,
                       petal.width = 0.25,
                       scale.area = FALSE, scale.length = TRUE,
                       lwd = 5)

grid.arrange(fg1, fg2, fg3, nrow = 1)


fg1 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z = z,
                       size = 20,
                       petal.base.shape = 1.25, petal.tip.shape = 1,
                       petal.width = 0.25,
                       scale.area = FALSE, scale.length = FALSE)

fg2 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z = z,
                       size = 20,
                       petal.base.shape = 1.25, petal.tip.shape = 1,
                       petal.width = 0.25,
                       scale.area = FALSE, scale.length = FALSE,
                       lwd = 3)

fg3 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z = z,
                       size = 20,
                       petal.base.shape = 1.25, petal.tip.shape = 1,
                       petal.width = 0.25,
                       scale.area = FALSE, scale.length = FALSE,
                       lwd = 5)

grid.arrange(fg1, fg2, fg3, nrow = 1)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
# Adjust centre colour and fill
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

fg1 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z = z,
                       size = 20,
                       petal.base.shape = 1.25, petal.tip.shape = 1,
                       petal.width = 0.25,
                       scale.area = TRUE, scale.length = FALSE,
                       centre.size = 5,
                       col.centre = "salmon", fill.centre = "white")

fg2 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z = z,
                       size = 20,
                       petal.base.shape = 1.25, petal.tip.shape = 1,
                       petal.width = 0.25,
                       scale.area = TRUE, scale.length = FALSE,
                       centre.size = 5,
                       col.centre = "cyan", fill.centre = "white")

fg3 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z = z,
                       size = 20,
                       petal.base.shape = 1.25, petal.tip.shape = 1,
                       petal.width = 0.25,
                       scale.area = TRUE, scale.length = FALSE,
                       centre.size = 5,
                       col.centre = "green", fill.centre = "white")

grid.arrange(fg1, fg2, fg3, nrow = 1)

fg1 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z = z,
                       size = 20,
                       petal.base.shape = 1.25, petal.tip.shape = 1,
                       petal.width = 0.25,
                       scale.area = FALSE, scale.length = TRUE,
                       centre.size = 5,
                       col.centre = "salmon", fill.centre = "white")

fg2 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z = z,
                       size = 20,
                       petal.base.shape = 1.25, petal.tip.shape = 1,
                       petal.width = 0.25,
                       scale.area = FALSE, scale.length = TRUE,
                       centre.size = 5,
                       col.centre = "cyan", fill.centre = "white")

fg3 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z = z,
                       size = 20,
```

```
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25,
                        scale.area = FALSE, scale.length = TRUE,
                        centre.size = 5,
                        col.centre = "green", fill.centre = "white")

grid.arrange(fg1, fg2, fg3, nrow = 1)

fg1 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = z,
                        size = 20,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25,
                        scale.area = FALSE, scale.length = FALSE,
                        centre.size = 5,
                        col.centre = "salmon", fill.centre = "white")

fg2 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = z,
                        size = 20,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25,
                        scale.area = FALSE, scale.length = FALSE,
                        centre.size = 5,
                        col.centre = "cyan", fill.centre = "white")

fg3 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = z,
                        size = 20,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25,
                        scale.area = FALSE, scale.length = FALSE,
                        centre.size = 5,
                        col.centre = "green", fill.centre = "white")

grid.arrange(fg1, fg2, fg3, nrow = 1)

fg1 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = z,
                        size = 20,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25,
                        scale.area = TRUE, scale.length = FALSE,
                        centre.size = 5,
                        fill.centre = "salmon")

fg2 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = z,
                        size = 20,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25,
                        scale.area = TRUE, scale.length = FALSE,
                        centre.size = 5,
```

```
                        fill.centre = "cyan")

fg3 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = z,
                        size = 20,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25,
                        scale.area = TRUE, scale.length = FALSE,
                        centre.size = 5,
                        fill.centre = "green")

grid.arrange(fg1, fg2, fg3, nrow = 1)

fg1 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = z,
                        size = 20,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25,
                        scale.area = FALSE, scale.length = TRUE,
                        centre.size = 5,
                        fill.centre = "salmon")

fg2 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = z,
                        size = 20,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25,
                        scale.area = FALSE, scale.length = TRUE,
                        centre.size = 5,
                        fill.centre = "cyan")

fg3 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = z,
                        size = 20,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25,
                        scale.area = FALSE, scale.length = TRUE,
                        centre.size = 5,
                        fill.centre = "green")

grid.arrange(fg1, fg2, fg3, nrow = 1)

fg1 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = z,
                        size = 20,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25,
                        scale.area = FALSE, scale.length = FALSE,
                        centre.size = 5,
                        fill.centre = "salmon")

fg2 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = z,
```

```
                        size = 20,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25,
                        scale.area = FALSE, scale.length = FALSE,
                        centre.size = 5,
                        fill.centre = "cyan")

fg3 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = z,
                        size = 20,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25,
                        scale.area = FALSE, scale.length = FALSE,
                        centre.size = 5,
                        fill.centre = "green")

grid.arrange(fg1, fg2, fg3, nrow = 1)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust petal colour and fill
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

fg1 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = z,
                        size = 20,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25,
                        scale.area = TRUE, scale.length = FALSE,
                        col = "salmon")

fg2 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = z,
                        size = 20,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25,
                        scale.area = TRUE, scale.length = FALSE,
                        col = "cyan")

fg3 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = z,
                        size = 20,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25,
                        scale.area = TRUE, scale.length = FALSE,
                        col = "green")

grid.arrange(fg1, fg2, fg3, nrow = 1)

fg1 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = z,
                        size = 20,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25,
```

```
                        scale.area = FALSE, scale.length = TRUE,
                        col = "salmon")

fg2 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = z,
                        size = 20,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25,
                        scale.area = FALSE, scale.length = TRUE,
                        col = "cyan")

fg3 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = z,
                        size = 20,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25,
                        scale.area = FALSE, scale.length = TRUE,
                        col = "green")

grid.arrange(fg1, fg2, fg3, nrow = 1)

fg1 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = z,
                        size = 20,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25,
                        scale.area = FALSE, scale.length = FALSE,
                        col = "salmon")

fg2 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = z,
                        size = 20,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25,
                        scale.area = FALSE, scale.length = FALSE,
                        col = "cyan")

fg3 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = z,
                        size = 20,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25,
                        scale.area = FALSE, scale.length = FALSE,
                        col = "green")

grid.arrange(fg1, fg2, fg3, nrow = 1)

fg1 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = z,
                        size = 20,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25,
                        scale.area = TRUE, scale.length = FALSE,
```

```
                        fill = "salmon")

fg2 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = z,
                        size = 20,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25,
                        scale.area = TRUE, scale.length = FALSE,
                        fill = "cyan")

fg3 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = z,
                        size = 20,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25,
                        scale.area = TRUE, scale.length = FALSE,
                        fill = "green")

grid.arrange(fg1, fg2, fg3, nrow = 1)

fg1 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = z,
                        size = 20,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25,
                        scale.area = FALSE, scale.length = TRUE,
                        fill = "salmon")

fg2 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = z,
                        size = 20,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25,
                        scale.area = FALSE, scale.length = TRUE,
                        fill = "cyan")

fg3 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = z,
                        size = 20,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25,
                        scale.area = FALSE, scale.length = TRUE,
                        fill = "green")

grid.arrange(fg1, fg2, fg3, nrow = 1)

fg1 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = z,
                        size = 20,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25,
                        scale.area = FALSE, scale.length = FALSE,
                        fill = "salmon")
```

```
fg2 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z = z,
                       size = 20,
                       petal.base.shape = 1.25, petal.tip.shape = 1,
                       petal.width = 0.25,
                       scale.area = FALSE, scale.length = FALSE,
                       fill = "cyan")

fg3 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z = z,
                       size = 20,
                       petal.base.shape = 1.25, petal.tip.shape = 1,
                       petal.width = 0.25,
                       scale.area = FALSE, scale.length = FALSE,
                       fill = "green")

grid.arrange(fg1, fg2, fg3, nrow = 1)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Multivariate petal fill and colour
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

fg1 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z = z,
                       size = 20,
                       petal.base.shape = 1.25, petal.tip.shape = 1,
                       petal.width = 0.25,
                       scale.area = TRUE, scale.length = FALSE,
                       fill = RColorBrewer::brewer.pal(6, "Dark2"))

fg2 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z = z,
                       size = 20,
                       petal.base.shape = 1.25, petal.tip.shape = 1,
                       petal.width = 0.25,
                       scale.area = FALSE, scale.length = TRUE,
                       fill = RColorBrewer::brewer.pal(6, "Dark2"))

fg3 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z = z,
                       size = 20,
                       petal.base.shape = 1.25, petal.tip.shape = 1,
                       petal.width = 0.25,
                       scale.area = FALSE, scale.length = FALSE,
                       fill = RColorBrewer::brewer.pal(6, "Dark2"))

grid.arrange(fg1, fg2, fg3, nrow = 1)

fg1 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z = z,
                       size = 20,
                       petal.base.shape = 1.25, petal.tip.shape = 1,
```

```
                        petal.width = 0.25, lwd = 3,
                        scale.area = TRUE, scale.length = FALSE,
                        col = RColorBrewer::brewer.pal(6, "Dark2"))

fg2 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = z,
                        size = 20,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25, lwd = 3,
                        scale.area = FALSE, scale.length = TRUE,
                        col = RColorBrewer::brewer.pal(6, "Dark2"))

fg3 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = z,
                        size = 20,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25, lwd = 3,
                        scale.area = FALSE, scale.length = FALSE,
                        col = RColorBrewer::brewer.pal(6, "Dark2"))

grid.arrange(fg1, fg2, fg3, nrow = 1)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust grid levels
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

gl <- split(x = c(rep(c(1, 2, 3), 4),
                  rep(c(1, 2, 3, 4), 2)),
            f = c(rep(1:4, each = 3),
                  rep(5:6, each = 4)))

gl

fg1 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = c(1, 3, 2, 1, 2, 3),
                        size = 15,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25,
                        scale.area = TRUE, scale.length = FALSE,
                        draw.grid = TRUE, grid.levels = gl,
                        fill = "white", col.grid = "black")

fg2 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = c(1, 3, 2, 1, 2, 3),
                        size = 15,
                        petal.base.shape = 1.25, petal.tip.shape = 1,
                        petal.width = 0.25,
                        scale.area = FALSE, scale.length = TRUE,
                        draw.grid = TRUE, grid.levels = gl,
                        fill = "white", col.grid = "black")

grid.arrange(fg1, fg2, nrow = 1)
```

```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust fill and grid level colours
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

gl <- split(x = c(rep(c(1, 2, 3), 4),
                  rep(c(1, 2, 3, 4), 2)),
            f = c(rep(1:4, each = 3),
                  rep(5:6, each = 4)))

gl

fg1 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z = c(1, 3, 2, 1, 2, 3),
                       size = 15,
                       petal.base.shape = 1.25, petal.tip.shape = 1,
                       petal.width = 0.25,
                       scale.area = TRUE, scale.length = FALSE,
                       draw.grid = TRUE, grid.levels = gl,
                       col = "white", col.grid = "white",
                       fill = RColorBrewer::brewer.pal(6, "Dark2"))

fg2 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z = c(1, 3, 2, 1, 2, 3),
                       size = 15,
                       petal.base.shape = 1.25, petal.tip.shape = 1,
                       petal.width = 0.25,
                       scale.area = FALSE, scale.length = TRUE,
                       draw.grid = TRUE, grid.levels = gl,
                       col = "white", col.grid = "white",
                       fill = RColorBrewer::brewer.pal(6, "Dark2"))

grid.arrange(fg1, fg2, nrow = 1)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjusting petal shape
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Petal width
fg1 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z = c(0.5), size = 20,
                       petal.base.shape = 1, petal.tip.shape = 1,
                       petal.width = 0.5,
                       scale.area = FALSE, scale.length = FALSE)

fg2 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z = c(0.5), size = 20,
                       petal.base.shape = 1, petal.tip.shape = 1,
                       petal.width = 1,
                       scale.area = FALSE, scale.length = FALSE)

fg3 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z = c(0.5), size = 20,
                       petal.base.shape = 1, petal.tip.shape = 1,
```

```
                        petal.width = 2,
                        scale.area = FALSE, scale.length = FALSE)

grid.arrange(fg1, fg2, fg3, nrow = 1)

# Tip shape
fg1 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = c(0.5), size = 20,
                        petal.base.shape = 1, petal.tip.shape = 0.5,
                        petal.width = 0.5,
                        scale.area = FALSE, scale.length = FALSE)

fg2 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = c(0.5), size = 20,
                        petal.base.shape = 1, petal.tip.shape = 1,
                        petal.width = 0.5,
                        scale.area = FALSE, scale.length = FALSE)

fg3 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = c(0.5), size = 20,
                        petal.base.shape = 1, petal.tip.shape = 2,
                        petal.width = 0.5,
                        scale.area = FALSE, scale.length = FALSE)

grid.arrange(fg1, fg2, fg3, nrow = 1)

# Base shape
fg1 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = c(0.5), size = 20,
                        petal.base.shape = 0.5, petal.tip.shape = 1,
                        petal.width = 0.5,
                        scale.area = FALSE, scale.length = FALSE)

fg2 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = c(0.5), size = 20,
                        petal.base.shape = 1, petal.tip.shape = 1,
                        petal.width = 0.5,
                        scale.area = FALSE, scale.length = FALSE)

fg3 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = c(0.5), size = 20,
                        petal.base.shape = 2, petal.tip.shape = 1,
                        petal.width = 0.5,
                        scale.area = FALSE, scale.length = FALSE)

grid.arrange(fg1, fg2, fg3, nrow = 1)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjusting petal notch
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Base notch
fg1 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
```

```
                        z = c(0.5), size = 20,
                        petal.base.shape = 1, petal.tip.shape = 1,
                        petal.width = 0.5,
                        petal.base.notch = 0.9,
                        scale.area = FALSE, scale.length = FALSE)

fg2 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = c(0.5), size = 20,
                        petal.base.shape = 1, petal.tip.shape = 1,
                        petal.width = 0.5,
                        petal.base.notch = 0.5,
                        scale.area = FALSE, scale.length = FALSE)

fg3 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = c(0.5), size = 20,
                        petal.base.shape = 1, petal.tip.shape = 1,
                        petal.width = 0.5,
                        petal.base.notch = 0,
                        scale.area = FALSE, scale.length = FALSE)

fg4 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = c(0.5), size = 20,
                        petal.base.shape = 1, petal.tip.shape = 1,
                        petal.width = 0.5,
                        petal.base.notch = -0.9,
                        scale.area = FALSE, scale.length = FALSE)

fg5 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = c(0.5), size = 20,
                        petal.base.shape = 1, petal.tip.shape = 1,
                        petal.width = 0.5,
                        petal.base.notch = -0.5,
                        scale.area = FALSE, scale.length = FALSE)

grid.arrange(arrangeGrob(fg1, fg2, nrow = 1),
             arrangeGrob(fg3),
             arrangeGrob(fg4, fg4, nrow = 1),
             nrow = 3)

# Base notch width
fg1 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = c(0.5), size = 20,
                        petal.base.shape = 1, petal.tip.shape = 1,
                        petal.width = 0.5,
                        petal.base.notch = 0.5,
                        petal.base.notch.width = 0.05,
                        scale.area = FALSE, scale.length = FALSE)

fg2 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = c(0.5), size = 20,
                        petal.base.shape = 1, petal.tip.shape = 1,
                        petal.width = 0.5,
                        petal.base.notch = 0.5,
```

```
                        petal.base.notch.width = 0.1,
                        scale.area = FALSE, scale.length = FALSE)

fg3 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = c(0.5), size = 20,
                        petal.base.shape = 1, petal.tip.shape = 1,
                        petal.width = 0.5,
                        petal.base.notch = 0.5,
                        petal.base.notch.width = 0.15,
                        scale.area = FALSE, scale.length = FALSE)

grid.arrange(fg1, fg2, fg3, nrow = 1)

# Tip notch
fg1 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = c(0.5), size = 20,
                        petal.base.shape = 1, petal.tip.shape = 1,
                        petal.width = 0.5,
                        petal.tip.notch = 0.9,
                        scale.area = FALSE, scale.length = FALSE)

fg2 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = c(0.5), size = 20,
                        petal.base.shape = 1, petal.tip.shape = 1,
                        petal.width = 0.5,
                        petal.tip.notch = 0.5,
                        scale.area = FALSE, scale.length = FALSE)

fg3 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = c(0.5), size = 20,
                        petal.base.shape = 1, petal.tip.shape = 1,
                        petal.width = 0.5,
                        petal.tip.notch = 0,
                        scale.area = FALSE, scale.length = FALSE)

fg4 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = c(0.5), size = 20,
                        petal.base.shape = 1, petal.tip.shape = 1,
                        petal.width = 0.5,
                        petal.tip.notch = -0.9,
                        scale.area = FALSE, scale.length = FALSE)

fg5 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = c(0.5), size = 20,
                        petal.base.shape = 1, petal.tip.shape = 1,
                        petal.width = 0.5,
                        petal.tip.notch = -0.5,
                        scale.area = FALSE, scale.length = FALSE)

grid.arrange(arrangeGrob(fg1, fg2, nrow = 1),
             arrangeGrob(fg3),
             arrangeGrob(fg4, fg4, nrow = 1),
             nrow = 3)
```

```
# Base notch width
fg1 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z = c(0.5), size = 20,
                       petal.base.shape = 1, petal.tip.shape = 1,
                       petal.width = 0.5,
                       petal.tip.notch = 0.5,
                       petal.tip.notch.width = 0.05,
                       scale.area = FALSE, scale.length = FALSE)

fg2 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z = c(0.5), size = 20,
                       petal.base.shape = 1, petal.tip.shape = 1,
                       petal.width = 0.5,
                       petal.tip.notch = 0.5,
                       petal.tip.notch.width = 0.1,
                       scale.area = FALSE, scale.length = FALSE)

fg3 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z = c(0.5), size = 20,
                       petal.base.shape = 1, petal.tip.shape = 1,
                       petal.width = 0.5,
                       petal.tip.notch = 0.5,
                       petal.tip.notch.width = 0.15,
                       scale.area = FALSE, scale.length = FALSE)

grid.arrange(fg1, fg2, fg3, nrow = 1)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjusting petal curvature
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Curvature
fg1 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z = c(1), size = 20,
                       petal.base.shape = 0.5, petal.tip.shape = 1,
                       petal.width = 0.15,
                       petal.curvature = -2.5,
                       scale.area = FALSE, scale.length = TRUE)

fg2 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z = c(1), size = 20,
                       petal.base.shape = 0.5, petal.tip.shape = 1,
                       petal.width = 0.15,
                       petal.curvature = 0,
                       scale.area = FALSE, scale.length = TRUE)

fg3 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z = c(1), size = 20,
                       petal.base.shape = 0.5, petal.tip.shape = 1,
                       petal.width = 0.15,
                       petal.curvature = 2.5,
                       scale.area = FALSE, scale.length = TRUE)
```

```
grid.arrange(fg1, fg2, fg3, nrow = 1)

# Curvature position
fg1 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z = c(1), size = 20,
                       petal.base.shape = 0.5, petal.tip.shape = 1,
                       petal.width = 0.15,
                       petal.curvature = -2.5,
                       petal.curvature.position = 0.2,
                       scale.area = FALSE, scale.length = TRUE)

fg1 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z = c(1), size = 20,
                       petal.base.shape = 0.5, petal.tip.shape = 1,
                       petal.width = 0.15,
                       petal.curvature = -2.5,
                       petal.curvature.position = 0.5,
                       scale.area = FALSE, scale.length = TRUE)

fg1 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z = c(1), size = 20,
                       petal.base.shape = 0.5, petal.tip.shape = 1,
                       petal.width = 0.15,
                       petal.curvature = -2.5,
                       petal.curvature.position = 0.8,
                       scale.area = FALSE, scale.length = TRUE)

grid.arrange(fg1, fg2, fg3, nrow = 1)

# Curvature width
fg1 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z = c(1), size = 20,
                       petal.base.shape = 0.5, petal.tip.shape = 1,
                       petal.width = 0.15,
                       petal.curvature = -2.5,
                       petal.curvature.width = 0.05,
                       scale.area = FALSE, scale.length = TRUE)

fg2 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z = c(1), size = 20,
                       petal.base.shape = 0.5, petal.tip.shape = 1,
                       petal.width = 0.15,
                       petal.curvature = -2.5,
                       petal.curvature.width = 0.2,
                       scale.area = FALSE, scale.length = TRUE)

fg3 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                       z = c(1), size = 20,
                       petal.base.shape = 0.5, petal.tip.shape = 1,
                       petal.width = 0.15,
                       petal.curvature = -2.5,
                       petal.curvature.width = 0.5,
```

```
                        scale.area = FALSE, scale.length = TRUE)

grid.arrange(fg1, fg2, fg3, nrow = 1)

# Sigmoid curve
fg1 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = c(1), size = 20,
                        petal.base.shape = 0.5, petal.tip.shape = 1,
                        petal.width = 0.15,
                        petal.curvature = c(4, -4),
                        petal.curvature.position = c(0.2, 0.5),
                        scale.area = FALSE, scale.length = TRUE)

fg2 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = c(1), size = 20,
                        petal.base.shape = 0.5, petal.tip.shape = 1,
                        petal.width = 0.15,
                        petal.curvature = c(0, 0),
                        petal.curvature.position = 0.2,
                        scale.area = FALSE, scale.length = TRUE)

fg3 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = c(1), size = 20,
                        petal.base.shape = 0.5, petal.tip.shape = 1,
                        petal.width = 0.15,
                        petal.curvature = c(-4, 4),
                        petal.curvature.position = c(0.2, 0.5),
                        scale.area = FALSE, scale.length = TRUE)

grid.arrange(fg1, fg2, fg3, nrow = 1)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjusting petal waist
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Waist
fg1 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = c(0.5), size = 20,
                        petal.base.shape = 1, petal.tip.shape  = 1,
                        petal.width = 0.75,
                        petal.waist = 0.5,
                        scale.area = FALSE, scale.length = FALSE)

fg2 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = c(0.5), size = 20,
                        petal.base.shape = 1, petal.tip.shape  = 1,
                        petal.width = 0.75,
                        petal.waist = 0,
                        scale.area = FALSE, scale.length = FALSE)

fg3 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = c(0.5), size = 20,
                        petal.base.shape = 1, petal.tip.shape  = 1,
```

```
                        petal.width = 0.75,
                        petal.waist = -0.5,
                        scale.area = FALSE, scale.length = FALSE)

grid.arrange(fg1, fg2, fg3, nrow = 1)

# Waist width
fg1 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = c(0.5), size = 20,
                        petal.base.shape = 1, petal.tip.shape  = 1,
                        petal.width = 0.5,
                        petal.waist = 0.5,
                        petal.waist.width = 0.05,
                        scale.area = FALSE, scale.length = FALSE)

fg2 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = c(0.5), size = 20,
                        petal.base.shape = 1, petal.tip.shape  = 1,
                        petal.width = 0.5,
                        petal.waist = 0.5,
                        petal.waist.width = 0.15,
                        scale.area = FALSE, scale.length = FALSE)

fg3 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = c(0.5), size = 20,
                        petal.base.shape = 1, petal.tip.shape  = 1,
                        petal.width = 0.5,
                        petal.waist = 0.5,
                        petal.waist.width = 0.3,
                        scale.area = FALSE, scale.length = FALSE)

grid.arrange(fg1, fg2, fg3, nrow = 1)

# Waist position
fg1 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = c(0.5), size = 20,
                        petal.base.shape = 1, petal.tip.shape  = 1,
                        petal.width = 0.5,
                        petal.waist = 0.5,
                        petal.waist.width = 0.05,
                        petal.waist.position = 0.1,
                        scale.area = FALSE, scale.length = FALSE)

fg2 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                        z = c(0.5), size = 20,
                        petal.base.shape = 1, petal.tip.shape  = 1,
                        petal.width = 0.5,
                        petal.waist = 0.5,
                        petal.waist.width = 0.05,
                        petal.waist.position = 0.5,
                        scale.area = FALSE, scale.length = FALSE)

fg3 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
```

```
                        z = c(0.5), size = 20,
                        petal.base.shape = 1, petal.tip.shape  = 1,
                        petal.width = 0.5,
                        petal.waist = 0.5,
                        petal.waist.width = 0.05,
                        petal.waist.position = 0.9,
                        scale.area = FALSE, scale.length = FALSE)

grid.arrange(fg1, fg2, fg3, nrow = 1)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Generate botanical shapes
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Linear
fg1 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.1, "npc"),
                        z = c(0.5), size = 20,
                        petal.base.shape = 0.2, petal.tip.shape = 0.2,
                        petal.width = 0.2,
                        scale.area = FALSE, scale.length = FALSE)

ant1 <- textGrob(label = "Linear",
                 x = unit(0.5, "npc"), y = unit(0.4, "npc"),
                 gp = gpar(fontsize = 12, fontface = "bold"))

# Lanceolate
fg2 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.1, "npc"),
                        z = c(0.5), size = 20,
                        petal.base.shape = 0.8, petal.tip.shape = 2.5,
                        petal.width = 0.4,
                        scale.area = FALSE, scale.length = FALSE)

ant2 <- textGrob(label = "Lanceolate",
                 x = unit(0.5, "npc"), y = unit(0.4, "npc"),
                 gp = gpar(fontsize = 12, fontface = "bold"))

# Oblong
fg3 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.1, "npc"),
                        z = c(0.5), size = 20,
                        petal.base.shape = 0.3, petal.tip.shape = 0.3,
                        petal.width = 0.5,
                        scale.area = FALSE, scale.length = FALSE)

ant3 <- textGrob(label = "Oblong",
                 x = unit(0.5, "npc"), y = unit(0.4, "npc"),
                 gp = gpar(fontsize = 12, fontface = "bold"))

# Elliptic
fg4 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.1, "npc"),
                        z = c(0.5), size = 20,
                        petal.base.shape = 1, petal.tip.shape = 1,
                        petal.width = 0.5,
                        scale.area = FALSE, scale.length = FALSE)
```

```
ant4 <- textGrob(label = "Elliptic",
                 x = unit(0.5, "npc"), y = unit(0.4, "npc"),
                 gp = gpar(fontsize = 12, fontface = "bold"))

# Ovate
fg5 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.1, "npc"),
                       z = c(0.5), size = 20,
                       petal.base.shape = 0.8, petal.tip.shape = 1.8,
                       petal.width = 0.6,
                       scale.area = FALSE, scale.length = FALSE)

ant5 <- textGrob(label = "Ovate",
                 x = unit(0.5, "npc"), y = unit(0.4, "npc"),
                 gp = gpar(fontsize = 12, fontface = "bold"))


# Obovate
fg6 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.1, "npc"),
                       z = c(0.5), size = 20,
                       petal.base.shape = 1.8, petal.tip.shape = 0.8,
                       petal.width = 0.6,
                       scale.area = FALSE, scale.length = FALSE)

ant6 <- textGrob(label = "Obovate",
                 x = unit(0.5, "npc"), y = unit(0.4, "npc"),
                 gp = gpar(fontsize = 12, fontface = "bold"))

# Orbicular
fg7 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.1, "npc"),
                       z = c(0.5), size = 20,
                       petal.base.shape = 1, petal.tip.shape = 0.6,
                       petal.width = 1,
                       scale.area = FALSE, scale.length = FALSE)

ant7 <- textGrob(label = "Orbicular",
                 x = unit(0.5, "npc"), y = unit(0.4, "npc"),
                 gp = gpar(fontsize = 12, fontface = "bold"))

# Spatulate
fg8 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.1, "npc"),
                       z = c(0.5), size = 20,
                       petal.base.shape = 4, petal.tip.shape = 0.9,
                       petal.width = 0.5,
                       scale.area = FALSE, scale.length = FALSE)

ant8 <- textGrob(label = "Spatulate",
                 x = unit(0.5, "npc"), y = unit(0.4, "npc"),
                 gp = gpar(fontsize = 12, fontface = "bold"))

# Cuneate
fg9 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.1, "npc"),
                       z = c(0.5), size = 20,
```

```
                      petal.base.shape = 2, petal.tip.shape = 0.4,
                      petal.width = 0.6,
                      scale.area = FALSE, scale.length = FALSE)

ant9 <- textGrob(label = "Cuneate",
                 x = unit(0.5, "npc"), y = unit(0.4, "npc"),
                 gp = gpar(fontsize = 12, fontface = "bold"))

# Cuneate
fg9 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.1, "npc"),
                      z = c(0.5), size = 20,
                      petal.base.shape = 2, petal.tip.shape = 0.4,
                      petal.width = 0.6,
                      scale.area = FALSE, scale.length = FALSE)

ant9 <- textGrob(label = "Cuneate",
                 x = unit(0.5, "npc"), y = unit(0.4, "npc"),
                 gp = gpar(fontsize = 12, fontface = "bold"))

# Deltoid
fg10 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.1, "npc"),
                       z = c(0.5), size = 20,
                       petal.base.shape = 0.3, petal.tip.shape = 2.5,
                       petal.width = 0.7,
                       scale.area = FALSE, scale.length = FALSE)

ant10 <- textGrob(label = "Deltoid",
                  x = unit(0.5, "npc"), y = unit(0.4, "npc"),
                  gp = gpar(fontsize = 12, fontface = "bold"))

# Rhomboid
fg11 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.1, "npc"),
                       z = c(0.5), size = 20,
                       petal.base.shape = 1.5, petal.tip.shape = 1.5,
                       petal.width = 0.7,
                       scale.area = FALSE, scale.length = FALSE)

ant11 <- textGrob(label = "Rhomboid",
                  x = unit(0.5, "npc"), y = unit(0.4, "npc"),
                  gp = gpar(fontsize = 12, fontface = "bold"))

# Flabellate
fg12 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.1, "npc"),
                       z = c(0.5), size = 20,
                       petal.base.shape = 2, petal.tip.shape = 0.2,
                       petal.width = 0.8,
                       petal.tip.notch = -0.05,
                       petal.tip.notch.width = 0.01,
                       scale.area = FALSE, scale.length = FALSE)

ant12 <- textGrob(label = "Flabellate",
                  x = unit(0.5, "npc"), y = unit(0.4, "npc"),
                  gp = gpar(fontsize = 12, fontface = "bold"))
```

```
# Oblanceolate
fg13 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.1, "npc"),
                        z = c(0.5), size = 20,
                        petal.base.shape = 2.5, petal.tip.shape  = 0.8,
                        petal.width = 0.45,
                        scale.area = FALSE, scale.length = FALSE)

ant13 <- textGrob(label = "Oblanceolate",
                  x = unit(0.5, "npc"), y = unit(0.4, "npc"),
                  gp = gpar(fontsize = 12, fontface = "bold"))

# Pandurate (fiddle-shaped)
fg14 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.1, "npc"),
                        z = c(0.5), size = 20,
                        petal.base.shape = 1, petal.tip.shape  = 1,
                        petal.width = 0.75,
                        petal.waist = 0.65,
                        petal.waist.position = 0.50,
                        petal.waist.width = 0.12,
                        scale.area = FALSE, scale.length = FALSE)

ant14 <- textGrob(label = "Pandurate",
                  x = unit(0.5, "npc"), y = unit(0.4, "npc"),
                  gp = gpar(fontsize = 12, fontface = "bold"))

# Falcate (sickle-shaped)
fg15 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.1, "npc"),
                        z = c(0.5), size = 20,
                        petal.base.shape = 0.5, petal.tip.shape  = 2,
                        petal.width = 0.2,
                        petal.curvature = 2.8,
                        petal.curvature.position = 0.3,
                        petal.curvature.width = 0.5,
                        scale.area = FALSE, scale.length = FALSE)

ant15 <- textGrob(label = "Falcate",
                  x = unit(0.5, "npc"), y = unit(0.4, "npc"),
                  gp = gpar(fontsize = 12, fontface = "bold"))

# Cordate (Heart-shaped)
fg16 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.1, "npc"),
                        z = c(0.5), size = 20,
                        petal.base.shape = 0.6, petal.tip.shape = 2,
                        petal.width = 0.8,
                        petal.base.notch = 0.25,
                        petal.base.notch.width = 0.01,
                        scale.area = FALSE, scale.length = FALSE)

ant16 <- textGrob(label = "Cordate",
                  x = unit(0.5, "npc"), y = unit(0.4, "npc"),
                  gp = gpar(fontsize = 12, fontface = "bold"))
```

```
# Obcordate (Inverted heart-shaped)
fg17 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.1, "npc"),
                        z = c(0.5), size = 20,
                        petal.base.shape = 2, petal.tip.shape = 0.6,
                        petal.width = 0.8,
                        petal.tip.notch = 0.25,
                        petal.tip.notch.width = 0.01,
                        scale.area = FALSE, scale.length = FALSE)

ant17 <- textGrob(label = "Obcordate",
                  x = unit(0.5, "npc"), y = unit(0.4, "npc"),
                  gp = gpar(fontsize = 12, fontface = "bold"))

# Reniform (Kidney-shaped)
fg18 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.1, "npc"),
                        z = c(0.5), size = 20,
                        petal.base.shape = 0.6, petal.tip.shape = 0.5,
                        petal.width = 1.3,
                        petal.base.notch = 0.35,
                        petal.base.notch.width = 0.01,
                        scale.area = FALSE, scale.length = FALSE)

ant18 <- textGrob(label = "Reniform",
                  x = unit(0.5, "npc"), y = unit(0.4, "npc"),
                  gp = gpar(fontsize = 12, fontface = "bold"))

# Sagittate (Arrowhead-shaped)
fg19 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.1, "npc"),
                        z = c(0.5), size = 20,
                        petal.base.shape = 0.4, petal.tip.shape = 2.5,
                        petal.width = 0.3,
                        petal.base.notch = 0.25,
                        petal.base.notch.width = 0.01,
                        scale.area = FALSE, scale.length = FALSE)

ant19 <- textGrob(label = "Sagittate",
                  x = unit(0.5, "npc"), y = unit(0.4, "npc"),
                  gp = gpar(fontsize = 12, fontface = "bold"))

# Hastate (Spear-shaped, flaring lobes)
fg20 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.1, "npc"),
                        z = c(0.5), size = 20,
                        petal.base.shape = 0.2, petal.tip.shape = 2.8,
                        petal.width = 0.75,
                        petal.base.notch = 0.12,
                        petal.base.notch.width = 0.01,
                        scale.area = FALSE, scale.length = FALSE)

ant20 <- textGrob(label = "Hastate",
                  x = unit(0.5, "npc"), y = unit(0.4, "npc"),
                  gp = gpar(fontsize = 12, fontface = "bold"))

# Auriculate (Ear-lapped base)
```

```
fg21 <- flowerglyphGrob(x = unit(0.5, "npc"), y = unit(0.1, "npc"),
                        z = c(0.5), size = 20,
                        petal.base.shape = 1.5, petal.tip.shape = 1.0,
                        petal.width = 0.5,
                        petal.base.notch = 0.10,
                        petal.base.notch.width = 0.04,
                        scale.area = FALSE, scale.length = FALSE)

ant21 <- textGrob(label = "Auriculate",
                  x = unit(0.5, "npc"), y = unit(0.4, "npc"),
                  gp = gpar(fontsize = 12, fontface = "bold"))


grid.arrange(
  arrangeGrob(fg1, ant1, nrow = 2),
  arrangeGrob(fg2, ant2, nrow = 2),
  arrangeGrob(fg3, ant3, nrow = 2),
  arrangeGrob(fg4, ant4, nrow = 2),
  arrangeGrob(fg5, ant5, nrow = 2),
  arrangeGrob(fg6, ant6, nrow = 2),
  nrow = 2)

grid.arrange(
  arrangeGrob(fg7, ant7, nrow = 2),
  arrangeGrob(fg8, ant8, nrow = 2),
  arrangeGrob(fg9, ant9, nrow = 2),
  arrangeGrob(fg10, ant10, nrow = 2),
  arrangeGrob(fg11, ant11, nrow = 2),
  arrangeGrob(fg12, ant12, nrow = 2),
  nrow = 2)

grid.arrange(
  arrangeGrob(fg13, ant13, nrow = 2),
  arrangeGrob(fg14, ant14, nrow = 2),
  arrangeGrob(fg15, ant15, nrow = 2),
  arrangeGrob(fg16, ant16, nrow = 2),
  arrangeGrob(fg17, ant17, nrow = 2),
  arrangeGrob(fg18, ant18, nrow = 2),
  nrow = 2)

grid.arrange(
  arrangeGrob(fg19, ant19, nrow = 2),
  arrangeGrob(fg20, ant20, nrow = 2),
  nrow = 1)
```

---

geom_bubbleglyph *Add Bubble Glyphs as a Scatterplot*

Description

The bubbleglyph geom is used to plot multivariate data as bubble glyphs in a scatterplot. Each variable specified in `cols` is depicted as a circle, with the radius (or area) scaled according to the corresponding value.

Usage

```
geom_bubbleglyph(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
  cols = character(0L),
  scale.radius = TRUE,
  scale.area = FALSE,
  bubble.layout = c("annulus", "circle", "line", "chain", "hub", "pack"),
  connector = c("none", "foreground", "background"),
  fill.bubble = NULL,
  fill.gradient = NULL,
  colour.bubble = NULL,
  colour.grid = NULL,
  linewidth = 1,
  linewidth.grid = linewidth,
  linejoin = c("mitre", "round", "bevel"),
  angle.start = 0,
  angle.stop = 2 * base::pi,
  draw.grid = FALSE,
  legend.glyph.dims = setNames(rep(0.5, length(cols)), cols),
  show.legend = NA,
  repel = FALSE,
  repel.control = ggmultiglyph.repel.control(),
  inherit.aes = TRUE
)
```

Arguments

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

can be created from a `formula` (e.g. `~ head(.x, 10)`).

`stat` The statistical transformation to use on the data for this layer. When using a `geom_*()` function to construct a layer, the `stat` argument can be used to override the default coupling between geoms and stats. The `stat` argument accepts the following:

- A `Stat` ggproto subclass, for example `StatCount`.
- A string naming the stat. To give the stat as a string, strip the function name of the `stat_` prefix. For example, to use `stat_count()`, give the stat as `"count"`.
- For more information and other ways to specify the stat, see the layer stat documentation.

`position` A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The `position` argument accepts the following:

- The result of calling a position function, such as `position_jitter()`. This method allows for passing extra arguments to the position.
- A string naming the position adjustment. To give the position as a string, strip the function name of the `position_` prefix. For example, to use `position_jitter()`, give the position as `"jitter"`.
- For more information and other ways to specify the position, see the layer position documentation.

`...` 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` A character vector containing the names of at least two columns specifying the variables to be plotted in the glyphs. The selected columns must be either numeric or factor variables.

`scale.radius` logical. If `TRUE`, the values in `z` are used directly as the bubble radii.

`scale.area` logical. If `TRUE`, values in `z` are treated as bubble areas, and radii are derived as $\sqrt{z}$, so that bubble area (rather than radius) is proportional to `z`.

`bubble.layout` The layout algorithm used to position the bubbles. One of `"circle"`, `"line"`, `"chain"`, `"hub"`, `"pack"` or `"annulus"` (See **Layouts** for more details).

`connector` The style used to connect bubbles to one another. One of `"none"` (no connectors are drawn), `"foreground"` (connectors are drawn on top of the bubbles) or `"background"` (connectors are drawn underneath the bubbles).

`fill.bubble` The fill colour of the bubbles.

`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.bubble` The colour of bubbles.

`colour.grid` The colour of grid lines.

`linewidth` The line width of the circles.

`linewidth.grid` The line width of the grid circles.

`linejoin` The line join style for the tile polygon. Either `"mitre"`, `"round"` or `"bevel"`.

`angle.start` The start angle for the bubbles in radians for `layout = "circle"` and `layout = "hub"`. Default is zero.

`angle.stop` The stop angle for the bubbles in radians for `layout = "circle"` and `layout = "hub"`. Default is 2π.

`draw.grid` logical. If `TRUE`, grid levels are plotted as nested circles within each bubble. Default is `FALSE`.

`legend.glyph.dims` The dimensions of the legend glyph plot. Can be a numeric vector of unit length (where all the dimensions will have same value) or a numeric vector of same length as "cols" with the "cols" as names.

`show.legend` logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use `TRUE`. If `NA`, all levels are shown in legend, but unobserved levels are omitted.

`repel` logical. If `TRUE`, the glyphs are repel away from each other to avoid overlaps. Default is `FALSE`.

`repel.control` A list of control settings for the repel algorithm. Ignored if `repel = FALSE`. See `ggmultiglyph.repel.control` for details on the various control parameters.

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

Value

A `geom` layer.

Aesthetics

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

- **x**
- **y**
- alpha
- colour
- fill
- group
- size
- centre.size

See `vignette("ggplot2-specs", package = "ggplot2")` for further details on setting these aesthetics.

The following additional aesthetics are considered if `repel = TRUE`:

- point.size

- segment.linetype
- segment.colour
- segment.size
- segment.alpha
- segment.curvature
- segment.angle
- segment.ncp
- segment.shape
- segment.square
- segment.squareShape
- segment.inflect
- segment.debug

See `ggrepel` examples page for further details on setting these aesthetics.

Layouts

The following layouts are available.

`"circle"` Bubbles are arranged around the circumference of an invisible circle, evenly spaced by angle between `angle.start` and `angle.stop`. The radius of this invisible circle is derived from the bubble radii (the largest bubble radius plus the mean bubble radius, with a small buffer added) so that bubbles do not overlap regardless of their individual sizes. This is the default layout.

`"line"` Bubbles are arranged side by side along a straight line passing through the glyph centre, each bubble touching its neighbours. The line may be rotated to any orientation using the `line.angle` argument (in radians).

`"chain"` Bubbles are linked together in a meandering tangent chain. Each new bubble is placed tangent to its immediate predecessor, alternating between opposite sides of the chain to produce a distinctive zig-zag arrangement. Where necessary, the placement is adjusted to minimise overlap with previously placed bubbles while preserving the ordering of the data.

`"hub"` One bubble (the one with the largest radius) is treated as a central "hub", and all other bubbles are placed around it, each touching the hub bubble, at angles spaced between `angle.start` and `angle.stop`. This is a quick, deterministic layout: because the surrounding bubbles are not tested against one another for overlap, bubbles can overlap when several of them are large relative to the hub.

`"pack"` Bubbles are arranged using a circle-packing algorithm (`circleProgressiveLayout`), which places bubbles as close together as possible without any overlap (Collins and Stephenson 2003; Wang et al. 2006; Bedward et al. 2024). This produces the most compact, space-efficient arrangement of the available layouts, at the cost of the arrangement having no inherent order or direction (bubbles are not ordered along an axis or around a hub).

`"annulus"` Bubbles are arranged, in the order supplied, around the circumference of a ring (annulus), with each bubble touching its immediate neighbours on either side (the last bubble touching the first, closing the ring). The radius of the ring is solved numerically so that all adjacent bubbles are mutually tangent.

References

Bedward M, Eppstein D, Menzel P (2024). "packcircles: Circle Packing. R package version 0.3.7."

Collins CR, Stephenson K (2003). "A circle packing algorithm." *Computational Geometry*, **25**(3), 233–256.

Wang W, Wang H, Dai G, Wang H (2006). "Visualization of large hierarchical data by circle packing." In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 517–520. ISBN 978-1-59593-372-0.

See Also

bubbleglyphGrob

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

Examples

```
library(ggplot2)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Prepare the data ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Variables to map to glyphs
zs <- c("hp", "drat", "wt", "qsec", "vs", "am", "gear", "carb")

# Keep a copy of the original data
mtcars_fct <- mtcars

# Scaled numeric data
mtcars[zs] <- lapply(mtcars[zs], scales::rescale)

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

# Ordered factor data
mtcars_fct[zs[1:3]] <-
  lapply(mtcars_fct[zs[1:3]], function(x)
    ordered(cut(x, breaks = 3,
                labels = c("low", "medium", "high"))))

mtcars_fct[zs[4:8]] <-
  lapply(mtcars_fct[zs[4:8]], function(x)
    ordered(cut(x, breaks = 4,
                labels = c("tiny", "small", "medium", "large"))))

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

```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Mapped fill + scaled radius ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

ggplot(data = mtcars) +
  geom_bubbleglyph(aes(x = mpg, y = disp, fill = cyl),
                   cols = zs, size = 1.5,
                   alpha =  0.8) +
  ylim(c(-0, 550))

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Mapped fill + scaled area ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

ggplot(data = mtcars) +
  geom_bubbleglyph(aes(x = mpg, y = disp, fill = cyl),
                   cols = zs, size = 1.5,
                   scale.radius = FALSE, scale.area = TRUE,
                   alpha =  0.8) +
  ylim(c(-0, 550))


#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Mapped colour + scaled radius ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

ggplot(data = mtcars) +
  geom_bubbleglyph(aes(x = mpg, y = disp, colour = cyl),
                   cols = zs, size = 1.5, fill = "white",
                   alpha =  0.8, linewidth = 2) +
  ylim(c(-0, 550))

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Mapped colour + scaled area ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

ggplot(data = mtcars) +
  geom_bubbleglyph(aes(x = mpg, y = disp, colour = cyl),
                   cols = zs, size = 1.5, fill = "white",
                   scale.radius = FALSE, scale.area = TRUE,
                   alpha =  0.8, linewidth = 2) +
  ylim(c(-0, 550))

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Bubble layout variations ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Annulus
ggplot(data = mtcars) +
  geom_bubbleglyph(aes(x = mpg, y = disp, fill = cyl),
                   cols = zs, size = 1.5,
                   alpha =  0.8) +
```

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

# Circle
ggplot(data = mtcars) +
  geom_bubbleglyph(aes(x = mpg, y = disp, fill = cyl),
                   cols = zs, size = 1.5,
                   layout = "circle",
                   alpha =  0.8) +
  ylim(c(-0, 550))

# Line
ggplot(data = mtcars) +
  geom_bubbleglyph(aes(x = mpg, y = disp, fill = cyl),
                   cols = zs, size = 1.5,
                   layout = "line",
                   alpha =  0.8) +
  ylim(c(-0, 550))

# Chain
ggplot(data = mtcars) +
  geom_bubbleglyph(aes(x = mpg, y = disp, fill = cyl),
                   cols = zs, size = 1.5,
                   layout = "chain",
                   alpha =  0.8) +
  ylim(c(-0, 550))

# Hub
ggplot(data = mtcars) +
  geom_bubbleglyph(aes(x = mpg, y = disp, fill = cyl),
                   cols = zs, size = 1.5,
                   layout = "hub",
                   alpha =  0.8) +
  ylim(c(-0, 550))

# Pack
ggplot(data = mtcars) +
  geom_bubbleglyph(aes(x = mpg, y = disp, fill = cyl),
                   cols = zs, size = 1.5,
                   layout = "pack",
                   alpha =  0.8) +
  ylim(c(-0, 550))
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Bubbles with multivariate colours ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

ggplot(data = mtcars) +
  geom_bubbleglyph(aes(x = mpg, y = disp),
                   cols = zs, size = 1.5,
                   fill.bubble = RColorBrewer::brewer.pal(8, "Dark2"),
                   alpha =  0.8) +
  ylim(c(-0, 550))

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
# Gradient fill ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

ggplot(data = mtcars) +
  geom_bubbleglyph(aes(x = mpg, y = disp),
                   cols = zs, size = 1.5,
                   fill.gradient = "viridis",
                   alpha =  0.8) +
  ylim(c(-0, 550))

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Faceted ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

ggplot(data = mtcars) +
  geom_bubbleglyph(aes(x = mpg, y = disp, fill = cyl),
                   cols = zs, size = 1.5,
                   alpha =  0.8) +
  ylim(c(-0, 550)) +
  facet_grid(. ~ cyl)

ggplot(data = mtcars) +
  geom_bubbleglyph(aes(x = mpg, y = disp, colour = cyl),
                   cols = zs, size = 1.5,
                   alpha =  0.8) +
  ylim(c(-0, 550)) +
  facet_grid(. ~ cyl)

ggplot(data = mtcars) +
  geom_bubbleglyph(aes(x = mpg, y = disp),
                   cols = zs, size = 1.5,
                   fill.bubble = RColorBrewer::brewer.pal(8, "Dark2"),
                   alpha =  0.8) +
  ylim(c(-0, 550)) +
  facet_grid(. ~ cyl)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Repel glyphs ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

ggplot(data = mtcars) +
  geom_point(aes(x = mpg, y = disp, colour = cyl)) +
  geom_bubbleglyph(aes(x = mpg, y = disp, fill = cyl),
                   cols = zs, size = 1.5,
                   alpha =  1, repel = TRUE) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_point(aes(x = mpg, y = disp, colour = cyl)) +
  geom_bubbleglyph(aes(x = mpg, y = disp, colour = cyl),
                   cols = zs, size = 1.5,
                   alpha =  1, repel = TRUE) +
  ylim(c(-0, 550))
```

```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Grid lines (ordered factor variables) ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

ggplot(data = mtcars_fct) +
  geom_bubbleglyph(aes(x = mpg, y = disp, fill = cyl),
                   cols = zs, size = 0.5,
                   alpha =  0.8, draw.grid = TRUE) +
  ylim(c(-0, 550))

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Legend options ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Theme modifications for legend
legend_theme <-
  theme_bw(base_size = 7.5) +
  theme(legend.direction = "vertical",
        legend.box = "horizontal",
        legend.position = "bottom",
        legend.text = element_text(margin = margin(l = 7)),
        legend.key.height = unit(1, 'lines'))

# Glyph variable-wise legends
ggplot(data = mtcars) +
  geom_point(aes(x = mpg, y = disp, colour = cyl), show.legend = FALSE) +
  geom_bubbleglyph(aes(x = mpg, y = disp, fill = cyl),
                   cols = zs, size = 1,
                   alpha =  1, repel = TRUE) +
  ylim(c(-0, 550)) +
  scale_z_continuous(z = zs) +
  guide_z_order(z = zs, default_aes = "fill") +
  legend_theme

# Using custom guide
# bubbleglyphGrob
guide_bubblegrob <- bubbleglyphGrob(
  z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33, 0.6, 0.25),
  layout = "annulus",
  size = 5)
# guide_bubblegrob <-
#   addlabel.glyphGrob(grob = guide_bubblegrob, label = zs,
#                      push = 1, segment = FALSE)

ggplot(data = mtcars) +
  geom_point(aes(x = mpg, y = disp, colour = cyl), show.legend = FALSE) +
  geom_bubbleglyph(aes(x = mpg, y = disp, fill = cyl),
                   cols = zs, size = 1,
                   alpha =  1, repel = TRUE) +
  ylim(c(-0, 550)) +
  guides(fill = guide_legend(order = 1, position = "right"),
         custom = guide_custom(guide_bubblegrob,
```

```
                                width = unit(0.1, "npc"),
                                height = unit(0.1, "npc"),
                                position = "bottom",
                                theme = theme(legend.margin = margin(t = 40, b = 30))))
```

---

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,
  legend.glyph.dims = setNames(rep(1, length(cols)), cols),
  show.legend = NA,
  repel = FALSE,
  repel.control = ggmultiglyph.repel.control(),
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created.

A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).

stat — The statistical transformation to use on the data for this layer. When using a geom_*() function to construct a layer, the stat argument can be used to override the default coupling between geoms and stats. The stat argument accepts the following:

- A Stat ggproto subclass, for example StatCount.
- A string naming the stat. To give the stat as a string, strip the function name of the stat_ prefix. For example, to use stat_count(), give the stat as "count".
- For more information and other ways to specify the stat, see the layer stat documentation.

position — A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following:

- The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position.
- A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter".
- For more information and other ways to specify the position, see the layer position documentation.

... — 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 — A character vector containing the names of at least two columns specifying the variables to be plotted in the glyphs. The selected columns must be factor variables.

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.

legend.glyph.dims — The dimensions of the legend glyph plot. Can be a numeric vector of unit length (where all the dimensions will have same value) or a numeric vector of same length as "cols" with the "cols" as names.

show.legend — logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.

repel	logical. If TRUE, the glyphs are repel away from each other to avoid overlaps. Default is FALSE.
repel.control	A list of control settings for the repel algorithm. Ignored if repel = FALSE. See ggmultiglyph.repel.control for details on the various control parameters.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. annotation_borders().

Value

A geom layer.

Aesthetics

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

- **x**
- **y**
- alpha
- colour
- fill
- group

See vignette("ggplot2-specs", package = "ggplot2") for further details on setting these aesthetics.

The following additional aesthetics are considered if repel = TRUE:

- point.size
- segment.linetype
- segment.colour
- segment.size
- segment.alpha
- segment.curvature
- segment.angle
- segment.ncp
- segment.shape
- segment.square
- segment.squareShape
- segment.inflect
- segment.debug

See ggrepel examples page for further details on setting these aesthetics.

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_bubbleglyph(), geom_flowerglyph(), geom_metroglyph(), geom_pieglyph(), geom_profileglyph(), geom_starglyph(), geom_tileglyph()

Examples

```
library(ggplot2)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Prepare the data ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Variables to map to glyphs
zs <- c("hp", "drat", "wt", "qsec", "vs", "am", "gear", "carb")

# Keep a copy of the original data
mtcars_fct <- mtcars

# Ordered factor data
mtcars_fct[zs[1:3]] <-
  lapply(mtcars_fct[zs[1:3]], function(x)
    ordered(cut(x, breaks = 3,
                labels = c("low", "medium", "high"))))

mtcars_fct[zs[4:8]] <-
  lapply(mtcars_fct[zs[4:8]], function(x)
    ordered(cut(x, breaks = 4,
                labels = c("tiny", "small", "medium", "large"))))

mtcars_fct$cyl <- factor(mtcars_fct$cyl)
mtcars_fct$lab <- row.names(mtcars_fct)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Mapped fill ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

ggplot(data = mtcars_fct) +
  geom_dotglyph(aes(x = mpg, y = disp, fill = cyl),
                cols = zs, radius = 0.5,
                alpha =  0.8) +
```

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

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

ggplot(data = mtcars_fct) +
  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_fct) +
  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_fct) +
  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))

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Multivariate fill colours ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

ggplot(data = mtcars_fct) +
  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_fct) +
  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_fct) +
  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_fct) +
  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_fct) +
  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_fct) +
  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_fct) +
  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_fct) +
  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_fct) +
  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_fct) +
  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_fct) +
  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_fct) +
  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)
```

```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Repel glyphs ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

ggplot(data = mtcars_fct) +
  geom_point(aes(x = mpg, y = disp, colour = cyl)) +
  geom_dotglyph(aes(x = mpg, y = disp, fill = cyl),
                cols = zs, radius = 0.5,
                alpha =  1, repel = TRUE) +
  ylim(c(-0, 550))

ggplot(data = mtcars_fct) +
  geom_point(aes(x = mpg, y = disp, colour = cyl)) +
  geom_dotglyph(aes(x = mpg, y = disp, colour = cyl),
                cols = zs, radius = 0.5,
                mirror = FALSE,
                alpha =  1, repel = TRUE) +
  ylim(c(-0, 550))

ggplot(data = mtcars_fct) +
  geom_point(aes(x = mpg, y = disp)) +
  geom_dotglyph(aes(x = mpg, y = disp),
                cols = zs, radius = 0.5,
                flip.axes = TRUE,
                fill.dot = RColorBrewer::brewer.pal(8, "Dark2"),
                alpha =  1, repel = TRUE) +
  ylim(c(-0, 550))

ggplot(data = mtcars_fct) +
  geom_point(aes(x = mpg, y = disp)) +
  geom_dotglyph(aes(x = mpg, y = disp),
                cols = zs, radius = 0.5,
                mirror = FALSE, flip.axes = TRUE,
                fill.gradient = "viridis",
                alpha =  1, repel = TRUE) +
  ylim(c(-0, 550))

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Legend options ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Default legend/guide
ggplot(data = mtcars_fct) +
  geom_point(aes(x = mpg, y = disp, colour = cyl)) +
  geom_dotglyph(aes(x = mpg, y = disp, fill = cyl),
                cols = zs, radius = 0.5,
                alpha =  1, repel = TRUE) +
  ylim(c(-0, 550))

# Theme modifications for legend
legend_theme <-
  theme_bw(base_size = 7.5) +
  theme(legend.direction = "vertical",
        legend.box = "horizontal",
```

```
        legend.position = "bottom",
        legend.text = element_text(margin = margin(l = 7)),
        legend.key.height = unit(1.5, 'lines'))

# Glyph variable-wise legends
ggplot(data = mtcars_fct) +
  geom_point(aes(x = mpg, y = disp, colour = cyl), show.legend = FALSE) +
  geom_dotglyph(aes(x = mpg, y = disp, fill = cyl),
                cols = zs, radius = 0.5,
                alpha =  1, repel = TRUE) +
  ylim(c(-0, 550)) +
  scale_z_discrete(z = zs) +
  guide_z_order(z = zs, default_aes = "fill") +
  legend_theme


# Modifying the `legend.glyph.dims`
z_grid <- c(hp = 3, drat = 3, wt = 2,
            qsec = 2, vs = 3, am = 4,
            gear = 2, carb = 3)

ggplot(data = mtcars_fct) +
  geom_point(aes(x = mpg, y = disp, colour = cyl)) +
  geom_dotglyph(aes(x = mpg, y = disp, fill = cyl),
                cols = zs, radius = 0.5,
                alpha =  1, repel = TRUE,
                legend.glyph.dims = z_grid) +
  ylim(c(-0, 550))


ggplot(data = mtcars_fct) +
  geom_point(aes(x = mpg, y = disp, colour = cyl), show.legend = FALSE) +
  geom_dotglyph(aes(x = mpg, y = disp, fill = cyl),
                cols = zs, radius = 0.5,
                alpha =  1, repel = TRUE,
                legend.glyph.dims = z_grid) +
  ylim(c(-0, 550)) +
  scale_z_discrete(z = zs) +
  guide_z_order(z = zs, default_aes = "fill") +
  legend_theme

# Using custom guide
# dotglyphGrob
guide_dotgrob <-
  dotglyphGrob(z = z_grid, radius = 2.5)
# Add labels to starglyphGrob
guide_dotgrob <-
  addlabel.glyphGrob(grob = guide_dotgrob, label = zs,
                     push = 3, segment = FALSE,
                     angle = 45, hjust = 1)

# Another version
guide_dotgrob2 <-
```

```
  dotglyphGrob(z = rep(3, length(z_grid)), radius = 2,
               mirror = TRUE)
guide_dotgrob2 <-
  addlabel.glyphGrob(grob = guide_dotgrob2, label = zs,
                     push = 8, segment = TRUE,
                     angle = 45, hjust = 1)

# Multivariate colour version
clrs <- mapply(function(a, b) rep(a, b),
               RColorBrewer::brewer.pal(8, "Dark2"),
               rep(3, 8))
clrs <- unlist(clrs)

guide_dotgrob_mcol <-
  dotglyphGrob(z = rep(3, length(z_grid)), radius = 2,
               mirror = TRUE,
               fill = clrs)
guide_dotgrob_mcol <-
  addlabel.glyphGrob(grob = guide_dotgrob_mcol, label = zs,
                     push = 8, segment = FALSE,
                     angle = 45, hjust = 1)

ggplot(data = mtcars_fct) +
  geom_point(aes(x = mpg, y = disp, colour = cyl), show.legend = FALSE) +
  geom_dotglyph(aes(x = mpg, y = disp, fill = cyl),
                cols = zs, radius = 0.5, mirror = FALSE,
                alpha =  1, repel = TRUE) +
  ylim(c(-0, 550)) +
  guides(fill = guide_legend(order = 1, position = "right"),
         custom = guide_custom(guide_dotgrob,
                               width = unit(0.1, "npc"),
                               height = unit(0.1, "npc"),
                               position = "bottom",
                               theme = theme(legend.margin = margin(t = 30, b = 20))))

ggplot(data = mtcars_fct) +
  geom_point(aes(x = mpg, y = disp, colour = cyl), show.legend = FALSE) +
  geom_dotglyph(aes(x = mpg, y = disp, fill = cyl),
                cols = zs, radius = 0.5, mirror = TRUE,
                alpha =  1, repel = TRUE) +
  ylim(c(-0, 550)) +
  guides(fill = guide_legend(order = 1, position = "right"),
         custom = guide_custom(guide_dotgrob2,
                               width = unit(0.1, "npc"),
                               height = unit(0.1, "npc"),
                               position = "bottom",
                               theme = theme(legend.margin = margin(t = 5, b = 30))))

ggplot(data = mtcars_fct) +
  geom_point(aes(x = mpg, y = disp), show.legend = FALSE) +
  geom_dotglyph(aes(x = mpg, y = disp),
                cols = zs, radius = 0.5,
                fill.dot = RColorBrewer::brewer.pal(8, "Dark2"),
```

```
              alpha = 1, repel = TRUE) +
  ylim(c(-0, 550)) +
  guides(fill = guide_legend(order = 1, position = "right"),
         custom = guide_custom(guide_dotgrob_mcol,
                               width = unit(0.1, "npc"),
                               height = unit(0.1, "npc"),
                               position = "bottom",
                               theme = theme(legend.margin = margin(t = 5, b = 30))))
```

`geom_flowerglyph` *Add Flower Glyphs as a Scatterplot*

Description

The flowerglyph geom is used to plot multivariate data as flower glyphs (Van Onzenoodt et al. 2023) in a scatterplot. Each variable specified in `cols` is depicted as a "petal" radiating from the data point, with the petal length (or area) scaled according to the corresponding value.

Usage

```
geom_flowerglyph(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
  cols = character(0L),
  edges = 30,
  petal.width = 0.1,
  petal.base.shape = 2,
  petal.tip.shape = 1,
  petal.tip.notch = 0,
  petal.tip.notch.width = 0.15,
  petal.base.notch = 0,
  petal.base.notch.width = 0.15,
  petal.waist = 0,
  petal.waist.position = 0.5,
  petal.waist.width = 0.15,
  petal.curvature = 0,
  petal.curvature.position = 0.7,
  petal.curvature.width = 0.2,
  scale.length = TRUE,
  scale.area = FALSE,
  fill.petal = NULL,
  fill.gradient = NULL,
  colour.petal = NULL,
```

```
  colour.grid = NULL,
  linewidth = 1,
  linewidth.grid = linewidth,
  linejoin = c("mitre", "round", "bevel"),
  centre = TRUE,
  centre.size = 0.5,
  colour.centre = NULL,
  fill.centre = NULL,
  angle.start = 0,
  angle.stop = 2 * base::pi,
  draw.grid = FALSE,
  legend.glyph.dims = setNames(rep(0.5, length(cols)), cols),
  show.legend = NA,
  repel = FALSE,
  repel.control = ggmultiglyph.repel.control(),
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
stat	The statistical transformation to use on the data for this layer. When using a geom_*() function to construct a layer, the stat argument can be used to override the default coupling between geoms and stats. The stat argument accepts the following: • A Stat ggproto subclass, for example StatCount. • A string naming the stat. To give the stat as a string, strip the function name of the stat_ prefix. For example, to use stat_count(), give the stat as "count". • For more information and other ways to specify the stat, see the layer stat documentation.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position.

- A string naming the position adjustment. To give the position as a string, strip the function name of the `position_` prefix. For example, to use `position_jitter()`, give the position as `"jitter"`.
- For more information and other ways to specify the position, see the layer position documentation.

`...` 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` A character vector containing the names of at least two columns specifying the variables to be plotted in the glyphs. The selected columns must be either numeric or factor variables.

`edges` The number of points used to approximate the taper profile of one side of a petal. Higher values give smoother petal outlines.

`petal.width` The width of a petal (as a proportion of `size|`) at its widest point.

`petal.base.shape`
A positive numeric value controlling how rapidly the petal widens from its base. Smaller values produce a broader attachment to the centre of the glyph, whereas larger values delay the expansion of the petal, producing a narrower, more clawed base.

`petal.tip.shape`
A positive numeric value controlling how rapidly the petal tapers towards its tip. Smaller values produce broader, more rounded tips, whereas larger values concentrate the petal width closer to its middle, producing progressively narrower, more pointed or lanceolate tips.

`petal.tip.notch`
A numeric value controlling deformation of the petal tip. A value of `0` produces a smooth, unmodified tip. Positive values (typically > 0) create an increasingly deep inward cleft with smooth shoulders, producing retuse, emarginate, obcordate, or bifid petal tips. Negative values extend the tip beyond its nominal length, producing progressively more acuminate, cuspidate, or aristate tips.

`petal.tip.notch.width`
A positive numeric value controlling the breadth of the tip notch. Smaller values produce a narrow, sharply incised cleft, or point, whereas larger values produce a broader, more rounded indentation or extension.

`petal.base.notch`
A numeric value controlling deformation of the petal base. A value of `0` produces a smooth, unmodified base. Positive values shift the basal shoulders backward relative to the central notch vertex (which remains anchored at the origin), producing cordate, sagittate, reniform, hastate, or auriculate petal shapes. Negative values extend the base into a basal protrusion or claw.

`petal.base.notch.width`
A positive numeric value controlling the breadth of the basal notch or protrusion. Smaller values produce a narrow, sharply defined sinus or point, whereas larger values produce a broader, more rounded basal indentation or extension.

`petal.waist` A numeric value controlling the depth of a constriction along the petal. A value of `0` produces a uniformly tapered petal, whereas larger values produce an in-

creasingly pronounced narrowing, resulting in pandurate, fiddle-shaped, or otherwise constricted petals.

`petal.waist.position`
: A numeric value between `0` and `1` controlling the position of the waist along the petal axis, where `0` corresponds to the base and `1` to the tip.

`petal.waist.width`
: A positive numeric value controlling the breadth of the waist. Smaller values produce a sharp, localized constriction, whereas larger values produce a broader, more gradual narrowing.

`petal.curvature`
: A numeric value or vector controlling the curvature of the petal centreline. A value of `0` produces a straight petal, whereas single positive or negative values produce simple arcuate or falcate (C-shaped) curvature in opposite directions. Passing a vector specifies multiple curvature bends along the petal axis; for example, a vector of length 2 with opposing signs (e.g., `c(3.0, -3.0)`) produces a sigmoid (S-shaped) petal.

`petal.curvature.position`
: A numeric value or vector between `0` and `1` controlling the position of maximum curvature along the petal axis, where `0` corresponds to the base and `1` to the tip. When `petal.curvature` is a vector specifying multiple bends (e.g., for sigmoid shapes), this parameter specifies the location along the axis for each corresponding bend.

`petal.curvature.width`
: A positive numeric value controlling the breadth of the region over which the petal bends. Smaller values concentrate the curvature into a localized bend, whereas larger values distribute the curvature more evenly along the petal, producing a broader, more gradual arc.

`scale.length` logical. If `TRUE`, the petal lengths are scaled according to value of `z`.

`scale.area` logical. If `TRUE`, the area of the petals are scaled according to value of `z`.

`fill.petal` The fill colour of the petals.

`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.petal` The outline colour of the petals.

`colour.grid` The colour of the grid lines.

`linewidth` The petal outline line width.

`linewidth.grid` The line width for the grid lines.

`linejoin` The line join style for the petal outlines. Either `"mitre"`, `"round"` or `"bevel"`.

`centre` logical. If `TRUE`, a central point is drawn at the glyph origin. Default is `TRUE`.

`centre.size` The size (radius, in mm) of the central point.

`colour.centre` The line colour of the central point.

`fill.centre` The fill colour of the central point.

`angle.start` The start angle for the petals in radians. Default is zero.

`angle.stop` The stop angle for the petals in radians. Default is 2π.

`draw.grid` logical. If `TRUE`, grid levels are plotted along the central axis of each petal if all the variables specified in `cols` are an ordered factor. Default is `FALSE`.

`legend.glyph.dims` The dimensions of the legend glyph plot. Can be a numeric vector of unit length (where all the dimensions will have same value) or a numeric vector of same length as "cols" with the "cols" as names.

`show.legend` logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use `TRUE`. If `NA`, all levels are shown in legend, but unobserved levels are omitted.

`repel` logical. If `TRUE`, the glyphs are repel away from each other to avoid overlaps. Default is `FALSE`.

`repel.control` A list of control settings for the repel algorithm. Ignored if `repel = FALSE`. See `ggmultiglyph.repel.control` for details on the various control parameters.

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

Value

A `geom` layer.

Aesthetics

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

- **x**
- **y**
- alpha
- colour
- fill
- group
- size
- centre.size

See `vignette("ggplot2-specs", package = "ggplot2")` for further details on setting these aesthetics.

The following additional aesthetics are considered if `repel = TRUE`:

- point.size
- segment.linetype
- segment.colour
- segment.size
- segment.alpha

- segment.curvature
- segment.angle
- segment.ncp
- segment.shape
- segment.square
- segment.squareShape
- segment.inflect
- segment.debug

See ggrepel examples page for further details on setting these aesthetics.

References

Van Onzenoodt C, Vazquez P, Ropinski T (2023). "Out of the plane: Flower versus star glyphs to support high-dimensional exploration in two-dimensional embeddings." *IEEE Transactions on Visualization and Computer Graphics*, **29**(12), 5468–5482.

See Also

flowerglyphGrob

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

Examples

```
library(ggplot2)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Prepare the data ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Variables to map to glyphs
zs <- c("hp", "drat", "wt", "qsec", "vs", "am", "gear", "carb")

# Keep a copy of the original data
mtcars_fct <- mtcars

# Scaled numeric data
mtcars[zs] <- lapply(mtcars[zs], scales::rescale)

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

# Ordered factor data
mtcars_fct[zs[1:3]] <-
  lapply(mtcars_fct[zs[1:3]], function(x)
    ordered(cut(x, breaks = 3,
```

```
                labels = c("low", "medium", "high"))))

mtcars_fct[zs[4:8]] <-
  lapply(mtcars_fct[zs[4:8]], function(x)
    ordered(cut(x, breaks = 4,
                labels = c("tiny", "small", "medium", "large"))))

mtcars_fct$cyl <- factor(mtcars_fct$cyl)
mtcars_fct$lab <- row.names(mtcars_fct)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Mapped fill + scaled length ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

ggplot(data = mtcars) +
  geom_flowerglyph(aes(x = mpg, y = disp, fill = cyl),
                   cols = zs, size = 10,
                   petal.base.shape = 3, petal.width = 0.25,
                   alpha =  0.8) +
  ylim(c(-0, 550))


#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Mapped fill + scaled area ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

ggplot(data = mtcars) +
  geom_flowerglyph(aes(x = mpg, y = disp, fill = cyl),
                   cols = zs, size = 10,
                   petal.base.shape = 3, petal.width = 0.25,
                   scale.length = FALSE, scale.area = TRUE,
                   alpha =  0.8) +
  ylim(c(-0, 550))

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Mapped colour + scaled length ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

ggplot(data = mtcars) +
  geom_flowerglyph(aes(x = mpg, y = disp, colour  = cyl),
                   cols = zs, size = 10,
                   petal.base.shape = 3, petal.width = 0.25,
                   alpha =  0.8) +
  ylim(c(-0, 550))

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Mapped colour + scaled area ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

ggplot(data = mtcars) +
  geom_flowerglyph(aes(x = mpg, y = disp, colour  = cyl),
                   cols = zs, size = 10,
                   petal.base.shape = 3, petal.width = 0.25,
```

```
                  scale.length = FALSE, scale.area = TRUE,
                  alpha =  0.8) +
  ylim(c(-0, 550))

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Petal shape variations ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

ggplot(data = mtcars) +
  geom_flowerglyph(aes(x = mpg, y = disp, fill = cyl),
                  cols = zs, size = 10,
                  petal.base.shape = 5, petal.width = 0.25,
                  alpha =  0.8) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_flowerglyph(aes(x = mpg, y = disp, fill = cyl),
                  cols = zs, size = 10,
                  petal.base.shape = 3, petal.width = 0.25,
                  petal.tip.notch = 0.3,
                  alpha =  0.8) +
  ylim(c(-0, 550))

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Petals with multivariate colours ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

ggplot(data = mtcars) +
  geom_flowerglyph(aes(x = mpg, y = disp),
                  cols = zs, size = 10,
                  petal.base.shape = 3, petal.width = 0.25,
                  fill.petal = RColorBrewer::brewer.pal(8, "Dark2"),
                  alpha =  0.8) +
  ylim(c(-0, 550))

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Gradient fill ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

ggplot(data = mtcars) +
  geom_flowerglyph(aes(x = mpg, y = disp),
                  cols = zs, size = 10,
                  petal.base.shape = 3, petal.width = 0.25,
                  fill.gradient = "Greens",
                  alpha =  0.8) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_flowerglyph(aes(x = mpg, y = disp),
                  cols = zs, size = 10,
                  petal.base.shape = 3, petal.width = 0.25,
                  fill.gradient = "RdYlBu",
                  alpha =  0.8) +
```

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

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Faceted ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

ggplot(data = mtcars) +
  geom_flowerglyph(aes(x = mpg, y = disp, fill = cyl),
                   cols = zs, size = 10,
                   petal.base.shape = 3, petal.width = 0.25,
                   alpha =  0.8) +
  ylim(c(-0, 550)) +
  facet_grid(. ~ cyl)

ggplot(data = mtcars) +
  geom_flowerglyph(aes(x = mpg, y = disp, colour = cyl),
                   cols = zs, size = 10,
                   petal.base.shape = 3, petal.width = 0.25,
                   alpha =  0.8) +
  ylim(c(-0, 550)) +
  facet_grid(. ~ cyl)

ggplot(data = mtcars) +
  geom_flowerglyph(aes(x = mpg, y = disp),
                   cols = zs, size = 10,
                   petal.base.shape = 3, petal.width = 0.25,
                   fill.petal = RColorBrewer::brewer.pal(8, "Dark2"),
                   alpha =  0.8) +
  ylim(c(-0, 550)) +
  facet_grid(. ~ cyl)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Repel glyphs ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

ggplot(data = mtcars) +
  geom_point(aes(x = mpg, y = disp, colour = cyl)) +
  geom_flowerglyph(aes(x = mpg, y = disp, fill = cyl),
                   cols = zs, size = 10,
                   petal.base.shape = 3, petal.width = 0.25,
                   alpha =  1, repel = TRUE) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_point(aes(x = mpg, y = disp, colour = cyl)) +
  geom_flowerglyph(aes(x = mpg, y = disp, colour = cyl),
                   cols = zs, size = 10,
                   petal.base.shape = 3, petal.width = 0.25,
                   alpha =  1, repel = TRUE) +
  ylim(c(-0, 550))

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Grid as nested petals ----
```

```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

ggplot(data = mtcars_fct) +
  geom_flowerglyph(aes(x = mpg, y = disp, fill = cyl),
                   cols = zs, size = 4,
                   petal.base.shape = 3, petal.width = 0.25,
                   alpha =  0.8, draw.grid = TRUE) +
  ylim(c(-0, 550))

ggplot(data = mtcars_fct) +
  geom_flowerglyph(aes(x = mpg, y = disp, fill = cyl),
                   cols = zs, size = 5,
                   petal.base.shape = 3, petal.width = 0.25,
                   scale.length = FALSE, scale.area = TRUE,
                   alpha =  0.8, draw.grid = TRUE) +
  ylim(c(-0, 550))

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Legend options ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Theme modifications for legend
legend_theme <-
  theme_bw(base_size = 7.5) +
  theme(legend.direction = "vertical",
        legend.box = "horizontal",
        legend.position = "bottom",
        legend.text = element_text(margin = margin(l = 7)),
        legend.key.height = unit(1.5, 'lines'))

# Glyph variable-wise legends
ggplot(data = mtcars) +
  geom_point(aes(x = mpg, y = disp, colour = cyl), show.legend = FALSE) +
  geom_flowerglyph(aes(x = mpg, y = disp, fill = cyl),
                   cols = zs, size = 5,
                   petal.base.shape = 3, petal.width = 0.25,
                   alpha =  1, repel = TRUE) +
  ylim(c(-0, 550)) +
  scale_z_continuous(z = zs) +
  guide_z_order(z = zs, default_aes = "fill") +
  legend_theme

# Using custom guide
# flowerglyphGrob
guide_flowergrob <-
  flowerglyphGrob(z = c(0.54, 0.75, 0.8, 1.4, 0.85, 0.53, 0.6, 0.65),
                  petal.base.shape = 3, petal.width = 0.25,
                  size = 15)
# guide_flowergrob <-
#   addlabel.glyphGrob(grob = guide_flowergrob, label = zs,
#                      push = 1, segment = FALSE)

ggplot(data = mtcars) +
```

```
  geom_point(aes(x = mpg, y = disp, colour = cyl), show.legend = FALSE) +
  geom_flowerglyph(aes(x = mpg, y = disp, fill = cyl),
                   cols = zs, size = 5,
                   petal.base.shape = 3, petal.width = 0.25,
                   alpha =  1, repel = TRUE) +
  ylim(c(-0, 550)) +
  guides(fill = guide_legend(order = 1, position = "right"),
         custom = guide_custom(guide_flowergrob,
                               width = unit(0.1, "npc"),
                               height = unit(0.1, "npc"),
                               position = "bottom",
                               theme = theme(legend.margin = margin(t = 40, b = 30))))
```

`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,
  lineend = "butt",
  full = TRUE,
  draw.grid = FALSE,
  grid.point.size = 1,
  show.legend = NA,
  legend.glyph.dims = setNames(rep(0.5, length(cols)), cols),
  repel = FALSE,
  repel.control = ggmultiglyph.repel.control(),
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
stat	The statistical transformation to use on the data for this layer. When using a geom_*() function to construct a layer, the stat argument can be used to override the default coupling between geoms and stats. The stat argument accepts the following: • A Stat ggproto subclass, for example StatCount. • A string naming the stat. To give the stat as a string, strip the function name of the stat_ prefix. For example, to use stat_count(), give the stat as "count". • For more information and other ways to specify the stat, see the layer stat documentation.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.
...	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	A character vector containing the names of at least two columns specifying the variables to be plotted in the glyphs. The selected columns must be either numeric or factor variables.
circle.size	The size of the central circle (radius).
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.

`lineend`
: The line end style for the rays. Either `"round"`, `"butt"` or `"square"`.

`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 rays if all the variables specified in `cols` are an ordered factor. Default is `FALSE`.

`grid.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. To include legend keys for all levels, even when no data exists, use `TRUE`. If `NA`, all levels are shown in legend, but unobserved levels are omitted.

`legend.glyph.dims`
: The dimensions of the legend glyph plot. Can be a numeric vector of unit length (where all the dimensions will have same value) or a numeric vector of same length as "cols" with the "cols" as names.

`repel`
: logical. If `TRUE`, the glyphs are repel away from each other to avoid overlaps. Default is `FALSE`.

`repel.control`
: A list of control settings for the repel algorithm. Ignored if `repel = FALSE`. See `ggmultiglyph.repel.control` for details on the various control parameters.

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

Value

A `geom` layer.

Aesthetics

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

- **x**
- **y**
- alpha
- colour
- fill
- group
- circle.size

See `vignette("ggplot2-specs", package = "ggplot2")` for further details on setting these aesthetics.

The following additional aesthetics are considered if `repel = TRUE`:

- point.size
- segment.linetype
- segment.colour
- segment.size
- segment.alpha
- segment.curvature
- segment.angle
- segment.ncp
- segment.shape
- segment.square
- segment.squareShape
- segment.inflect
- segment.debug

See `ggrepel` examples page for further details on setting these aesthetics.

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_bubbleglyph()`, `geom_dotglyph()`, `geom_flowerglyph()`, `geom_pieglyph()`, `geom_profileglyph()`, `geom_starglyph()`, `geom_tileglyph()`

Examples

```
library(ggplot2)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Prepare the data ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Variables to map to glyphs
zs <- c("hp", "drat", "wt", "qsec", "vs", "am", "gear", "carb")

# Keep a copy of the original data
mtcars_fct <- mtcars

# Scaled numeric data
```

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

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

# Ordered factor data
mtcars_fct[zs[1:3]] <-
  lapply(mtcars_fct[zs[1:3]], function(x)
    ordered(cut(x, breaks = 3,
                labels = c("low", "medium", "high"))))

mtcars_fct[zs[4:8]] <-
  lapply(mtcars_fct[zs[4:8]], function(x)
    ordered(cut(x, breaks = 4,
                labels = c("tiny", "small", "medium", "large"))))

mtcars_fct$cyl <- factor(mtcars_fct$cyl)
mtcars_fct$lab <- row.names(mtcars_fct)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# 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) +
  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) +
  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) +
  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) +
  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) +
  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) +
  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) +
  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) +
  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) +
  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) +
```

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

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Rays with multivariate 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) +
  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) +
  ylim(c(-0, 550)) +
  facet_grid(. ~ cyl)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Repel glyphs ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

ggplot(data = mtcars) +
  geom_point(aes(x = mpg, y = disp, colour = cyl)) +
  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 =  1, repel = TRUE) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_point(aes(x = mpg, y = disp, colour = cyl)) +
  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,
                  colour.circle = "transparent",
                  size = 10, alpha =  1, repel = TRUE) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_point(aes(x = mpg, y = disp)) +
  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 =  1, repel = TRUE) +
  ylim(c(-0, 550))

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# With grid points ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

ggplot(data = mtcars_fct) +
  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,
                  draw.grid = TRUE, grid.point.size = 5) +
  ylim(c(-0, 550))

ggplot(data = mtcars_fct) +
  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,
                  draw.grid = TRUE, grid.point.size = 5) +
  ylim(c(-0, 550))

ggplot(data = mtcars_fct) +
  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, grid.point.size = 5) +
  ylim(c(-0, 550))

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
# Legend options ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Default legend/guide
ggplot(data = mtcars) +
  geom_point(aes(x = mpg, y = disp, colour = cyl)) +
  geom_metroglyph(aes(x = mpg, y = disp, colour = cyl),
                  cols = zs, circle.size = 2, colour.ray = NULL,
                  linewidth.circle = 2, linewidth.ray = 2,
                  size = 5, alpha =  1, repel = TRUE) +
  ylim(c(-0, 550))

# Theme modifications for legend
legend_theme <-
  theme_bw(base_size = 7.5) +
  theme(legend.direction = "vertical",
        legend.box = "horizontal",
        legend.position = "bottom",
        legend.text = element_text(margin = margin(l = 7)),
        legend.key.height = unit(1.5, 'lines'))

# Glyph variable-wise legends
ggplot(data = mtcars) +
  geom_point(aes(x = mpg, y = disp, colour = cyl)) +
  geom_metroglyph(aes(x = mpg, y = disp, colour = cyl),
                  cols = zs, circle.size = 1, colour.ray = NULL,
                  linewidth.circle = 2, linewidth.ray = 2,
                  size = 5, alpha =  1, repel = TRUE) +
  ylim(c(-0, 550)) +
  scale_z_continuous(z = zs) +
  guide_z_order(z = zs, default_aes = "colour") +
  legend_theme

# Modifying the `legend.glyph.dims`
zavg <- # Scaled average of the variables
  apply(mtcars[, zs], 2,
        function(x) {
          scales::rescale(mean(x),
                          from = range(x), # same range as in scale_z_continuous
                          to = c(0, 1))
        })
zavg

ggplot(data = mtcars) +
  geom_point(aes(x = mpg, y = disp, colour = cyl)) +
  geom_metroglyph(aes(x = mpg, y = disp, colour = cyl),
                  cols = zs, circle.size = 2, colour.ray = NULL,
                  linewidth.circle = 2, linewidth.ray = 2,
                  size = 5, alpha =  1, repel = TRUE) +
  ylim(c(-0, 550)) +
  xlim(c(8, 35))

ggplot(data = mtcars) +
```

```
  geom_point(aes(x = mpg, y = disp, colour = cyl)) +
  geom_metroglyph(aes(x = mpg, y = disp, colour = cyl),
                  cols = zs, circle.size = 1, colour.ray = NULL,
                  linewidth.circle = 2, linewidth.ray = 2,
                  size = 5, alpha =  1, repel = TRUE) +
  ylim(c(-0, 550)) +
  scale_z_continuous(z = zs) +
  guide_z_order(z = zs, default_aes = "colour")  +
  legend_theme

# Using custom guide
# metroglyphGrob
guide_metrogrob <- metroglyphGrob(
  z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33, 0.6, 0.25) * 0.5,
  size = 25)
# Add labels to metroglyphGrob
guide_metrogrob <-
  addlabel.glyphGrob(grob = guide_metrogrob, label = zs,
                     push = 1, segment = FALSE)

# Another version
guide_metrogrob2 <- metroglyphGrob(
  z = rep(0.3, length(zs)),
  size = 25)
guide_metrogrob2 <-
  addlabel.glyphGrob(grob = guide_metrogrob2, label = zs,
                     push = 1, segment = FALSE)

ggplot(data = mtcars) +
  geom_point(aes(x = mpg, y = disp, colour = cyl)) +
  geom_metroglyph(aes(x = mpg, y = disp, colour = cyl),
                  cols = zs, circle.size = 1, colour.ray = NULL,
                  linewidth.circle = 2, linewidth.ray = 2,
                  size = 5, alpha =  1, repel = TRUE) +
  ylim(c(-0, 550)) +
  guides(colour = guide_legend(order = 1, position = "right"),
         custom = guide_custom(guide_metrogrob,
                               width = unit(0.1, "npc"),
                               height = unit(0.1, "npc"),
                               position = "bottom",
                               theme = theme(legend.margin = margin(t = 40, b = 30))))

ggplot(data = mtcars) +
  geom_point(aes(x = mpg, y = disp, colour = cyl)) +
  geom_metroglyph(aes(x = mpg, y = disp, colour = cyl),
                  cols = zs, circle.size = 1, colour.ray = NULL,
                  linewidth.circle = 2, linewidth.ray = 2,
                  size = 5, alpha =  1, repel = TRUE) +
  ylim(c(-0, 550)) +
  guides(colour = guide_legend(order = 1, position = "right"),
         custom = guide_custom(guide_metrogrob2,
                               width = unit(0.1, "npc"),
                               height = unit(0.1, "npc"),
```

```
                                position = "bottom",
                                theme = theme(legend.margin = margin(t = 30, b = 30))))

# Legend in plots with grid points

z_grid <- c(hp = 3, drat = 3, wt = 2,
            qsec = 2, vs = 3, am = 4,
            gear = 2, carb = 3)

ggplot(data = mtcars_fct) +
  geom_metroglyph(aes(x = mpg, y = disp, colour = cyl),
                  cols = zs, circle.size = 1, colour.ray = NULL,
                  linewidth.circle = 2, linewidth.ray = 2,
                  size = 1.5, alpha =  0.8,
                  draw.grid = TRUE, grid.point.size = 5) +
  ylim(c(-0, 550))

ggplot(data = mtcars_fct) +
  geom_metroglyph(aes(x = mpg, y = disp, colour = cyl),
                  cols = zs, circle.size = 1, colour.ray = NULL,
                  linewidth.circle = 2, linewidth.ray = 2,
                  size = 1.5, alpha =  0.8,
                  draw.grid = TRUE, grid.point.size = 5) +
  ylim(c(-0, 550)) +
  scale_z_discrete(z = zs) +
  guide_z_order(z = zs, default_aes = "fill") +
  legend_theme

ggplot(data = mtcars_fct) +
  geom_metroglyph(aes(x = mpg, y = disp, colour = cyl),
                  cols = zs, circle.size = 1, colour.ray = NULL,
                  linewidth.circle = 2, linewidth.ray = 2,
                  size = 1.5, alpha =  0.8,
                  draw.grid = TRUE, grid.point.size = 5,
                  legend.glyph.dims = z_grid) +
  ylim(c(-0, 550))

ggplot(data = mtcars_fct) +
  geom_metroglyph(aes(x = mpg, y = disp, colour = cyl),
                  cols = zs, circle.size = 1, colour.ray = NULL,
                  linewidth.circle = 2, linewidth.ray = 2,
                  size = 1.5, alpha =  0.8,
                  draw.grid = TRUE, grid.point.size = 5,
                  legend.glyph.dims = z_grid) +
  ylim(c(-0, 550)) +
  scale_z_discrete(z = zs) +
  guide_z_order(z = zs, default_aes = "fill") +
  legend_theme

# metroglyphGrob with grid levels

z_grid_levels <- lapply(z_grid, function(x) 1:x)
```

```
guide_metrogrob_grid <-
  metroglyphGrob(z = z_grid,
                 size = 2.5, draw.grid = TRUE, circle.size = 2,
                 grid.levels = z_grid_levels,
                 grid.point.size = 0.1)

guide_metrogrob_grid <-
  addlabel.glyphGrob(grob = guide_metrogrob_grid, label = zs,
                     push = 1, segment = FALSE)

# Another version
guide_metrogrob_grid2 <-
  metroglyphGrob(z = rep(3, length(z_grid)),
                 size = 2.5, draw.grid = TRUE, circle.size = 2,
                 grid.levels = replicate(8, 1:3,
                                         simplify = FALSE),
                 grid.point.size = 0.1)
guide_metrogrob_grid2 <-
  addlabel.glyphGrob(grob = guide_metrogrob_grid2, label = zs,
                     push = 1, segment = FALSE)

ggplot(data = mtcars_fct) +
  geom_metroglyph(aes(x = mpg, y = disp, colour = cyl),
                  cols = zs, circle.size = 1, colour.ray = NULL,
                  linewidth.circle = 2, linewidth.ray = 2,
                  size = 1.5, alpha =  0.8,
                  draw.grid = TRUE, grid.point.size = 5,
                  legend.glyph.dims = z_grid) +
  ylim(c(-0, 550)) +
  guides(fill = guide_legend(order = 1, position = "right"),
         custom = guide_custom(guide_metrogrob_grid,
                               width = unit(0.1, "npc"),
                               height = unit(0.1, "npc"),
                               position = "bottom",
                               theme = theme(legend.margin = margin(t = 20, b = 30))))

ggplot(data = mtcars_fct) +
  geom_metroglyph(aes(x = mpg, y = disp, colour = cyl),
                  cols = zs, circle.size = 1, colour.ray = NULL,
                  linewidth.circle = 2, linewidth.ray = 2,
                  size = 1.5, alpha =  0.8,
                  draw.grid = TRUE, grid.point.size = 5,
                  legend.glyph.dims = z_grid) +
  ylim(c(-0, 550)) +
  guides(fill = guide_legend(order = 1, position = "right"),
         custom = guide_custom(guide_metrogrob_grid2,
                               width = unit(0.1, "npc"),
                               height = unit(0.1, "npc"),
                               position = "bottom",
                               theme = theme(legend.margin = margin(t = 20, b = 30))))
```

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,
  legend.glyph.dims = setNames(rep(0.5, length(cols)), cols),
  show.legend = NA,
  repel = FALSE,
  repel.control = ggmultiglyph.repel.control(),
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created.

A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).

stat — The statistical transformation to use on the data for this layer. When using a geom_*() function to construct a layer, the stat argument can be used to override the default coupling between geoms and stats. The stat argument accepts the following:

- A Stat ggproto subclass, for example StatCount.
- A string naming the stat. To give the stat as a string, strip the function name of the stat_ prefix. For example, to use stat_count(), give the stat as "count".
- For more information and other ways to specify the stat, see the layer stat documentation.

position — A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following:

- The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position.
- A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter".
- For more information and other ways to specify the position, see the layer position documentation.

... — 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 — A character vector containing the names of at least two columns specifying the variables to be plotted in the glyphs. The selected columns must be either numeric or factor variables.

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 levels are plotted along the sectors if all the variables specified in cols are an ordered factor. Default is FALSE.

`legend.glyph.dims`
: The dimensions of the legend glyph plot. Can be a numeric vector of unit length (where all the dimensions will have same value) or a numeric vector of same length as "cols" with the "cols" as names.

`show.legend`
: logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use `TRUE`. If `NA`, all levels are shown in legend, but unobserved levels are omitted.

`repel`
: logical. If `TRUE`, the glyphs are repel away from each other to avoid overlaps. Default is `FALSE`.

`repel.control`
: A list of control settings for the repel algorithm. Ignored if `repel = FALSE`. See `ggmultiglyph.repel.control` for details on the various control parameters.

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

Value

A `geom` layer.

Aesthetics

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

- **x**
- **y**
- alpha
- colour
- fill
- group
- size

See `vignette("ggplot2-specs", package = "ggplot2")` for further details on setting these aesthetics.

The following additional aesthetics are considered if `repel = TRUE`:

- point.size
- segment.linetype
- segment.colour
- segment.size
- segment.alpha
- segment.curvature
- segment.angle
- segment.ncp

- segment.shape
- segment.square
- segment.squareShape
- segment.inflect
- segment.debug

See ggrepel examples page for further details on setting these aesthetics.

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_bubbleglyph(), geom_dotglyph(), geom_flowerglyph(), geom_metroglyph(), geom_profileglyph(), geom_starglyph(), geom_tileglyph()

Examples

```
library(ggplot2)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Prepare the data ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Variables to map to glyphs
zs <- c("hp", "drat", "wt", "qsec", "vs", "am", "gear", "carb")

# Keep a copy of the original data
mtcars_fct <- mtcars

# Scaled numeric data
mtcars[zs] <- lapply(mtcars[zs], scales::rescale)

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

# Ordered factor data
mtcars_fct[zs[1:3]] <-
  lapply(mtcars_fct[zs[1:3]], function(x)
    ordered(cut(x, breaks = 3,
                labels = c("low", "medium", "high"))))
```

```
mtcars_fct[zs[4:8]] <-
  lapply(mtcars_fct[zs[4:8]], function(x)
    ordered(cut(x, breaks = 4,
                labels = c("tiny", "small", "medium", "large"))))

mtcars_fct$cyl <- factor(mtcars_fct$cyl)
mtcars_fct$lab <- row.names(mtcars_fct)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# 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 multivariate 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 multivariate 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)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Repel glyphs ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

ggplot(data = mtcars) +
  geom_point(aes(x = mpg, y = disp, colour = cyl)) +
  geom_pieglyph(aes(x = mpg, y = disp, fill = cyl),
                cols = zs, size = 10,
```

```
                alpha =  1, repel = TRUE) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_point(aes(x = mpg, y = disp, colour = cyl)) +
  geom_pieglyph(aes(x = mpg, y = disp, fill = cyl),
                cols = zs, size = 5,
                scale.radius = FALSE, scale.segment = TRUE,
                alpha =  1, full = FALSE, repel = TRUE) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_point(aes(x = mpg, y = disp, colour = cyl)) +
  geom_pieglyph(aes(x = mpg, y = disp, colour = cyl),
                cols = zs, size = 10,
                alpha =  1, repel = TRUE) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_point(aes(x = mpg, y = disp, colour = cyl)) +
  geom_pieglyph(aes(x = mpg, y = disp, colour = cyl),
                cols = zs, size = 5,
                scale.radius = FALSE, scale.segment = TRUE,
                alpha =  1, full = FALSE, repel = TRUE) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_point(aes(x = mpg, y = disp)) +
  geom_pieglyph(aes(x = mpg, y = disp),
                cols = zs, size = 10,
                fill.segment = RColorBrewer::brewer.pal(8, "Dark2"),
                alpha =  1, repel = TRUE) +
  ylim(c(-0, 550))

ggplot(data = mtcars) +
  geom_point(aes(x = mpg, y = disp)) +
  geom_pieglyph(aes(x = mpg, y = disp),
                cols = zs, size = 5,
                scale.radius = FALSE, scale.segment = FALSE,
                fill.gradient = "viridis",
                alpha =  1, repel = TRUE) +
  ylim(c(-0, 550))

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Grid lines (when scale.radius = TRUE) ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

ggplot(data = mtcars_fct) +
  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_fct) +
  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_fct) +
  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))

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Legend options ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Default legend/guide
ggplot(data = mtcars) +
  geom_point(aes(x = mpg, y = disp, colour = cyl)) +
  geom_pieglyph(aes(x = mpg, y = disp, fill = cyl),
                cols = zs, size = 5,
                alpha =  1, repel = TRUE) +
  ylim(c(-0, 550))

# Theme modifications for legend
legend_theme <-
  theme_bw(base_size = 7.5) +
  theme(legend.direction = "vertical",
        legend.box = "horizontal",
        legend.position = "bottom",
        legend.text = element_text(margin = margin(l = 7)),
        legend.key.height = unit(1.5, 'lines'))

# Glyph variable-wise legends
ggplot(data = mtcars) +
  geom_point(aes(x = mpg, y = disp, colour = cyl), show.legend = FALSE) +
  geom_pieglyph(aes(x = mpg, y = disp, fill = cyl),
                cols = zs, size = 5,
                alpha =  1, repel = TRUE) +
  ylim(c(-0, 550)) +
  scale_z_continuous(z = zs) +
  guide_z_order(z = zs, default_aes = "fill") +
  legend_theme

ggplot(data = mtcars) +
  geom_point(aes(x = mpg, y = disp, colour = cyl), show.legend = FALSE) +
  geom_pieglyph(aes(x = mpg, y = disp, fill = cyl),
                cols = zs, size = 3,
                scale.radius = FALSE, scale.segment = TRUE,
                fill.segment = RColorBrewer::brewer.pal(8, "Dark2"),
                alpha =  1, repel = TRUE) +
  ylim(c(-0, 550)) +
```

```
  scale_z_continuous(z = zs) +
  guide_z_order(z = zs, default_aes = "fill") +
  legend_theme

# Modifying the `legend.glyph.dims`
zavg <- # Scaled average of the variables
  apply(mtcars[, zs], 2,
        function(x) {
          scales::rescale(mean(x),
                          from = range(x), # same range as in scale_z_continuous
                          to = c(0, 1))
        })
zavg

ggplot(data = mtcars) +
  geom_point(aes(x = mpg, y = disp, colour = cyl)) +
  geom_pieglyph(aes(x = mpg, y = disp, fill = cyl),
                cols = zs, size = 5,
                alpha =  1, repel = TRUE,
                legend.glyph.dims = zavg) +
  ylim(c(-0, 550)) +
  xlim(c(8, 35))

ggplot(data = mtcars) +
  geom_point(aes(x = mpg, y = disp, colour = cyl), show.legend = FALSE) +
  geom_pieglyph(aes(x = mpg, y = disp, fill = cyl),
                cols = zs, size = 5,
                alpha =  1, repel = TRUE,
                legend.glyph.dims = zavg) +
  ylim(c(-0, 550)) +
  scale_z_continuous(z = zs) +
  guide_z_order(z = zs, default_aes = "fill")  +
  legend_theme

ggplot(data = mtcars) +
  geom_point(aes(x = mpg, y = disp, colour = cyl), show.legend = FALSE) +
  geom_pieglyph(aes(x = mpg, y = disp),
                cols = zs, size = 3,
                scale.radius = FALSE, scale.segment = TRUE,
                fill.segment = RColorBrewer::brewer.pal(8, "Dark2"),
                alpha =  1, repel = TRUE,
                legend.glyph.dims = zavg) +
  ylim(c(-0, 550)) +
  scale_z_continuous(z = zs) +
  guide_z_order(z = zs, default_aes = "fill")  +
  legend_theme

# Using custom guide
# pieglyphGrob
guide_piegrob <- pieglyphGrob(
  z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33, 0.6, 0.25) * 0.75,
  size = 25)
# Add labels to pieglyphGrob
```

```
guide_piegrob <-
  addlabel.glyphGrob(grob = guide_piegrob, label = zs,
                     push = 1, segment = FALSE)

# Another version
guide_piegrob2 <- pieglyphGrob(
  z = rep(0.5, length(zs)),
  size = 25)
guide_piegrob2 <-
  addlabel.glyphGrob(grob = guide_piegrob2, label = zs,
                     push = 1, segment = FALSE)

# Multivariate colour version
guide_piegrob_mcol <- pieglyphGrob(
  z = rep(0.5, length(zs)),
  size = 25, fill = RColorBrewer::brewer.pal(8, "Dark2"))
guide_piegrob_mcol <-
  addlabel.glyphGrob(grob = guide_piegrob_mcol, label = zs,
                     push = 1, segment = FALSE)

ggplot(data = mtcars) +
  geom_point(aes(x = mpg, y = disp, colour = cyl), show.legend = FALSE) +
  geom_pieglyph(aes(x = mpg, y = disp, fill = cyl),
                cols = zs, size = 5,
                alpha =  1, repel = TRUE) +
  ylim(c(-0, 550)) +
  guides(fill = guide_legend(order = 1, position = "right"),
         custom = guide_custom(guide_piegrob,
                               width = unit(0.1, "npc"),
                               height = unit(0.1, "npc"),
                               position = "bottom",
                               theme = theme(legend.margin = margin(t = 40, b = 30))))

ggplot(data = mtcars) +
  geom_point(aes(x = mpg, y = disp, colour = cyl), show.legend = FALSE) +
  geom_pieglyph(aes(x = mpg, y = disp, fill = cyl),
                cols = zs, size = 5,
                alpha =  1, repel = TRUE) +
  ylim(c(-0, 550)) +
  guides(fill = guide_legend(order = 1, position = "right"),
         custom = guide_custom(guide_piegrob2,
                               width = unit(0.1, "npc"),
                               height = unit(0.1, "npc"),
                               position = "bottom",
                               theme = theme(legend.margin = margin(t = 30, b = 30))))

ggplot(data = mtcars) +
  geom_point(aes(x = mpg, y = disp, colour = cyl), show.legend = FALSE) +
  geom_pieglyph(aes(x = mpg, y = disp),
                cols = zs, size = 3,
                scale.radius = FALSE, scale.segment = TRUE,
                fill.segment = RColorBrewer::brewer.pal(8, "Dark2"),
                alpha =  1, repel = TRUE) +
```

```
  ylim(c(-0, 550)) +
  guides(fill = guide_legend(order = 1, position = "right"),
         custom = guide_custom(guide_piegrob_mcol,
                               width = unit(0.1, "npc"),
                               height = unit(0.1, "npc"),
                               position = "bottom",
                               theme = theme(legend.margin = margin(t = 30, b = 30))))

# Legend in plots with grid lines (when scale.radius = TRUE)

z_grid <- c(hp = 3, drat = 3, wt = 2,
            qsec = 2, vs = 3, am = 4,
            gear = 2, carb = 3)

ggplot(data = mtcars_fct) +
  geom_pieglyph(aes(x = mpg, y = disp, fill = cyl),
                cols = zs, size = 1,
                alpha =  0.8, draw.grid = TRUE) +
  ylim(c(-0, 550))

ggplot(data = mtcars_fct) +
  geom_pieglyph(aes(x = mpg, y = disp, fill = cyl),
                cols = zs, size = 1,
                alpha =  0.8, draw.grid = TRUE) +
  ylim(c(-0, 550)) +
  scale_z_discrete(z = zs) +
  guide_z_order(z = zs, default_aes = "fill") +
  legend_theme

ggplot(data = mtcars_fct) +
  geom_pieglyph(aes(x = mpg, y = disp, fill = cyl),
                cols = zs, size = 1,
                alpha =  0.8, draw.grid = TRUE,
                legend.glyph.dims = z_grid) +
  ylim(c(-0, 550))

ggplot(data = mtcars_fct) +
  geom_pieglyph(aes(x = mpg, y = disp, fill = cyl),
                cols = zs, size = 1,
                alpha =  0.8, draw.grid = TRUE,
                legend.glyph.dims = z_grid) +
  ylim(c(-0, 550)) +
  scale_z_discrete(z = zs) +
  guide_z_order(z = zs, default_aes = "fill") +
  legend_theme

# pieglyphGrob with grid levels

z_grid_levels <- lapply(z_grid, function(x) 1:x)

guide_piegrob_grid <-
  pieglyphGrob(z = z_grid,
               size = 3, draw.grid = TRUE,
```

```
                col.grid = "black",
                grid.levels = z_grid_levels)

  guide_piegrob_grid <-
    addlabel.glyphGrob(grob = guide_piegrob_grid, label = zs,
                       push = 1, segment = FALSE)

  # Another version
  guide_piegrob_grid2 <-
    pieglyphGrob(z = rep(3, length(z_grid)),
                size = 3, draw.grid = TRUE,
                col.grid = "black",
                grid.levels = replicate(8, 1:3,
                                        simplify = FALSE))
  guide_piegrob_grid2 <-
    addlabel.glyphGrob(grob = guide_piegrob_grid2, label = zs,
                       push = 1, segment = FALSE)

  ggplot(data = mtcars_fct) +
    geom_pieglyph(aes(x = mpg, y = disp, fill = cyl),
                  cols = zs, size = 1,
                  alpha =  0.8, draw.grid = TRUE,
                  legend.glyph.dims = z_grid) +
    ylim(c(-0, 550)) +
    guides(fill = guide_legend(order = 1, position = "right"),
           custom = guide_custom(guide_piegrob_grid,
                                 width = unit(0.1, "npc"),
                                 height = unit(0.1, "npc"),
                                 position = "bottom",
                                 theme = theme(legend.margin = margin(t = 20, b = 30))))

  ggplot(data = mtcars_fct) +
    geom_pieglyph(aes(x = mpg, y = disp, fill = cyl),
                  cols = zs, size = 1,
                  alpha =  0.8, draw.grid = TRUE,
                  legend.glyph.dims = z_grid) +
    ylim(c(-0, 550)) +
    guides(fill = guide_legend(order = 1, position = "right"),
           custom = guide_custom(guide_piegrob_grid2,
                                 width = unit(0.1, "npc"),
                                 height = unit(0.1, "npc"),
                                 position = "bottom",
                                 theme = theme(legend.margin = margin(t = 20, b = 30))))
```

---

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,
  size = 1,
  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,
  legend.glyph.dims = setNames(rep(0.5, length(cols)), cols),
  show.legend = NA,
  repel = FALSE,
  repel.control = ggmultiglyph.repel.control(),
  inherit.aes = TRUE
)
```

Arguments

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

can be created from a `formula` (e.g. `~ head(.x, 10)`).

`stat` The statistical transformation to use on the data for this layer. When using a `geom_*()` function to construct a layer, the `stat` argument can be used to override the default coupling between geoms and stats. The `stat` argument accepts the following:

- A `Stat` ggproto subclass, for example `StatCount`.
- A string naming the stat. To give the stat as a string, strip the function name of the `stat_` prefix. For example, to use `stat_count()`, give the stat as `"count"`.
- For more information and other ways to specify the stat, see the layer stat documentation.

`position` A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The `position` argument accepts the following:

- The result of calling a position function, such as `position_jitter()`. This method allows for passing extra arguments to the position.
- A string naming the position adjustment. To give the position as a string, strip the function name of the `position_` prefix. For example, to use `position_jitter()`, give the position as `"jitter"`.
- For more information and other ways to specify the position, see the layer position documentation.

`...` 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` A character vector containing the names of at least two columns specifying the variables to be plotted in the glyphs. The selected columns must be either numeric or factor variables.

`width` The width of the bars.

`size` The size of glyphs.

`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 levels are plotted along the profile bars if all the variables specified in `cols` are an ordered factor. Default is `FALSE`.

`legend.glyph.dims` The dimensions of the legend glyph plot. Can be a numeric vector of unit length (where all the dimensions will have same value) or a numeric vector of same length as "cols" with the "cols" as names.

`show.legend` logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use `TRUE`. If `NA`, all levels are shown in legend, but unobserved levels are omitted.

`repel` logical. If `TRUE`, the glyphs are repel away from each other to avoid overlaps. Default is `FALSE`.

`repel.control` A list of control settings for the repel algorithm. Ignored if `repel = FALSE`. See `ggmultiglyph.repel.control` for details on the various control parameters.

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

Value

A `geom` layer.

Aesthetics

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

- **x**
- **y**
- alpha
- colour
- fill
- group

See `vignette("ggplot2-specs", package = "ggplot2")` for further details on setting these aesthetics.

The following additional aesthetics are considered if `repel = TRUE`:

- point.size
- segment.linetype
- segment.colour
- segment.size
- segment.alpha
- segment.curvature
- segment.angle

- segment.ncp
- segment.shape
- segment.square
- segment.squareShape
- segment.inflect
- segment.debug

See ggrepel examples page for further details on setting these aesthetics.

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_bubbleglyph(), geom_dotglyph(), geom_flowerglyph(), geom_metroglyph(), geom_pieglyph(), geom_starglyph(), geom_tileglyph()

Examples

```
library(ggplot2)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Prepare the data ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Variables to map to glyphs
zs <- c("hp", "drat", "wt", "qsec", "vs", "am", "gear", "carb")

# Keep a copy of the original data
mtcars_fct <- mtcars

# Scaled numeric data
mtcars[zs] <- lapply(mtcars[zs], scales::rescale)

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

# Ordered factor data
mtcars_fct[zs[1:3]] <-
  lapply(mtcars_fct[zs[1:3]], function(x)
    ordered(cut(x, breaks = 3,
                labels = c("low", "medium", "high"))))
```

```
mtcars_fct[zs[4:8]] <-
  lapply(mtcars_fct[zs[4:8]], function(x)
    ordered(cut(x, breaks = 4,
                labels = c("tiny", "small", "medium", "large"))))

mtcars_fct$cyl <- factor(mtcars_fct$cyl)
mtcars_fct$lab <- row.names(mtcars_fct)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Bar, line, mirror and flip.axes combinations ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

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 multivariate 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 multivariate 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)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Repel glyphs ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

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

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

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

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

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

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# With grid lines (when bar = TRUE) ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

ggplot(data = mtcars_fct) +
  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_fct) +
  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_fct) +
  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))

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Legend options ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Default legend/guide
ggplot(data = mtcars) +
  geom_point(aes(x = mpg, y = disp, colour = cyl)) +
  geom_profileglyph(aes(x = mpg, y = disp, fill = cyl),
                    cols = zs, size = 5, width = 1,
                    alpha =  1, repel = TRUE) +
  ylim(c(-0, 550))

# Theme modifications for legend
legend_theme <-
  theme_bw(base_size = 7.5) +
  theme(legend.direction = "vertical",
        legend.box = "horizontal",
        legend.position = "bottom",
        legend.text = element_text(margin = margin(l = 7)),
        legend.key.height = unit(1.5, 'lines'))

# Glyph variable-wise legends
ggplot(data = mtcars) +
  geom_point(aes(x = mpg, y = disp, colour = cyl), show.legend = FALSE) +
  geom_profileglyph(aes(x = mpg, y = disp, fill = cyl),
                    cols = zs, size = 5, width = 1,
                    alpha =  1, repel = TRUE) +
  ylim(c(-0, 550)) +
  scale_z_continuous(z = zs) +
  guide_z_order(z = zs, default_aes = "fill") +
  legend_theme

# Modifying the `legend.glyph.dims`
zavg <- # Scaled average of the variables
  apply(mtcars[, zs], 2,
        function(x) {
          scales::rescale(mean(x),
                          from = range(x), # same range as in scale_z_continuous
                          to = c(0, 1))
```

```
        })
zavg

ggplot(data = mtcars) +
  geom_point(aes(x = mpg, y = disp, colour = cyl), show.legend = FALSE) +
  geom_profileglyph(aes(x = mpg, y = disp, fill = cyl),
                    cols = zs, size = 5, width = 1,
                    alpha =  1, repel = TRUE,
                    legend.glyph.dims = zavg) +
  ylim(c(-0, 550)) +
  xlim(c(8, 35))

ggplot(data = mtcars) +
  geom_point(aes(x = mpg, y = disp, colour = cyl), show.legend = FALSE) +
  geom_profileglyph(aes(x = mpg, y = disp, fill = cyl),
                    cols = zs, size = 5, width = 1,
                    alpha =  1, repel = TRUE,
                    legend.glyph.dims = zavg) +
  ylim(c(-0, 550)) +
  scale_z_continuous(z = zs) +
  guide_z_order(z = zs, default_aes = "fill")  +
  legend_theme

# Using custom guide
# profileglyphGrob
guide_profilegrob <- profileglyphGrob(
  z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33, 0.6, 0.25) * 0.75,
  size = 25)
# Add labels to profileglyphGrob
guide_profilegrob <-
  addlabel.glyphGrob(grob = guide_profilegrob, label = zs,
                     segment = TRUE,
                     angle = 45, push = 1, hjust = 1)

# Another version
guide_profilegrob2 <- profileglyphGrob(
  z = rep(0.5, length(zs)),
  size = 25)
guide_profilegrob2 <-
  addlabel.glyphGrob(grob = guide_profilegrob2, label = zs,
                     push = 1, segment = FALSE,
                     angle = 45, hjust = 1)

# Multivariate colour version
guide_profilegrob_mcol <- profileglyphGrob(
  z = rep(0.5, length(zs)),
  size = 25, fill = RColorBrewer::brewer.pal(8, "Dark2"))
guide_profilegrob_mcol <-
  addlabel.glyphGrob(grob = guide_profilegrob_mcol, label = zs,
                     push = 1, segment = FALSE,
                     angle = 45, hjust = 1)

ggplot(data = mtcars) +
```

```
  geom_point(aes(x = mpg, y = disp, colour = cyl), show.legend = FALSE) +
  geom_profileglyph(aes(x = mpg, y = disp, fill = cyl),
                    cols = zs, size = 5, width = 1,
                    alpha =  1, repel = TRUE) +
  ylim(c(-0, 550)) +
  guides(fill = guide_legend(order = 1, position = "right"),
         custom = guide_custom(guide_profilegrob,
                               width = unit(0.1, "npc"),
                               height = unit(0.1, "npc"),
                               position = "bottom",
                               theme = theme(legend.margin = margin(t = 20, b = 50))))

ggplot(data = mtcars) +
  geom_point(aes(x = mpg, y = disp, colour = cyl), show.legend = FALSE) +
  geom_profileglyph(aes(x = mpg, y = disp, fill = cyl),
                    cols = zs, size = 5, width = 1,
                    alpha =  1, repel = TRUE) +
  ylim(c(-0, 550)) +
  guides(fill = guide_legend(order = 1, position = "right"),
         custom = guide_custom(guide_profilegrob2,
                               width = unit(0.1, "npc"),
                               height = unit(0.1, "npc"),
                               position = "bottom",
                               theme = theme(legend.margin = margin(t = 7, b = 30))))

ggplot(data = mtcars) +
  geom_point(aes(x = mpg, y = disp)) +
  geom_profileglyph(aes(x = mpg, y = disp),
                    cols = zs, size = 5, width = 1,
                    fill.bar = RColorBrewer::brewer.pal(8, "Dark2"),
                    alpha =  1, repel = TRUE) +
  ylim(c(-0, 550)) +
  guides(fill = guide_legend(order = 1, position = "right"),
         custom = guide_custom(guide_profilegrob_mcol,
                               width = unit(0.1, "npc"),
                               height = unit(0.1, "npc"),
                               position = "bottom",
                               theme = theme(legend.margin = margin(t = 5, b = 30))))

# Legend in plots with grid lines

z_grid <- c(hp = 3, drat = 3, wt = 2,
            qsec = 2, vs = 3, am = 4,
            gear = 2, carb = 3)

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

ggplot(data = mtcars_fct) +
  geom_profileglyph(aes(x = mpg, y = disp, fill = cyl),
```

```
                    cols = zs, size = 1, width = 1,
                    alpha =  0.8, draw.grid = TRUE) +
  ylim(c(-0, 550)) +
  scale_z_discrete(z = zs) +
  guide_z_order(z = zs, default_aes = "fill") +
  legend_theme

ggplot(data = mtcars_fct) +
  geom_profileglyph(aes(x = mpg, y = disp, fill = cyl),
                    cols = zs, size = 1, width = 1,
                    alpha =  0.8, draw.grid = TRUE,
                    legend.glyph.dims = z_grid) +
  ylim(c(-0, 550))

ggplot(data = mtcars_fct) +
  geom_profileglyph(aes(x = mpg, y = disp, fill = cyl),
                    cols = zs, size = 1, width = 1,
                    alpha =  0.8, draw.grid = TRUE,
                    legend.glyph.dims = z_grid) +
  ylim(c(-0, 550)) +
  scale_z_discrete(z = zs) +
  guide_z_order(z = zs, default_aes = "fill") +
  legend_theme

# profileglyphGrob with grid levels

z_grid_levels <- lapply(z_grid, function(x) 1:x)

guide_profilegrob_grid <-
  profileglyphGrob(z = z_grid,
                   size = 3, draw.grid = TRUE,
                   grid.levels = z_grid_levels)

guide_profilegrob_grid <-
  addlabel.glyphGrob(grob = guide_profilegrob_grid, label = zs,
                     push = 1, segment = TRUE,
                     angle = 45, hjust = 1)

# Another version
guide_profilegrob_grid2 <-
  profileglyphGrob(z = rep(3, length(z_grid)),
                   size = 3, draw.grid = TRUE,
                   grid.levels = replicate(8, 1:3,
                                           simplify = FALSE))
guide_profilegrob_grid2 <-
  addlabel.glyphGrob(grob = guide_profilegrob_grid2, label = zs,
                     push = 1, segment = FALSE,
                     angle = 45, hjust = 1)

ggplot(data = mtcars_fct) +
  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)) +
  guides(fill = guide_legend(order = 1, position = "right"),
         custom = guide_custom(guide_profilegrob_grid,
                               width = unit(0.1, "npc"),
                               height = unit(0.1, "npc"),
                               position = "bottom",
                               theme = theme(legend.margin = margin(t = 5, b = 30))))

ggplot(data = mtcars_fct) +
  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)) +
  guides(fill = guide_legend(order = 1, position = "right"),
         custom = guide_custom(guide_profilegrob_grid2,
                               width = unit(0.1, "npc"),
                               height = unit(0.1, "npc"),
                               position = "bottom",
                               theme = theme(legend.margin = margin(t = 5, b = 30))))
```

`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,
  grid.point.size = 1,
```

```
  legend.glyph.dims = setNames(rep(0.5, length(cols)), cols),
  show.legend = NA,
  repel = FALSE,
  repel.control = ggmultiglyph.repel.control(),
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
stat	The statistical transformation to use on the data for this layer. When using a geom_*() function to construct a layer, the stat argument can be used to override the default coupling between geoms and stats. The stat argument accepts the following: • A Stat ggproto subclass, for example StatCount. • A string naming the stat. To give the stat as a string, strip the function name of the stat_ prefix. For example, to use stat_count(), give the stat as "count". • For more information and other ways to specify the stat, see the layer stat documentation.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.
...	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	A character vector containing the names of at least two columns specifying the variables to be plotted in the glyphs. The selected columns must be either numeric or factor variables.

`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 an ordered factor. Default is `FALSE`.

`grid.point.size`
The size of the grid points in native units.

`legend.glyph.dims`
The dimensions of the legend glyph plot. Can be a numeric vector of unit length (where all the dimensions will have same value) or a numeric vector of same length as "cols" with the "cols" as names.

`show.legend` logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use `TRUE`. If `NA`, all levels are shown in legend, but unobserved levels are omitted.

`repel` logical. If `TRUE`, the glyphs are repel away from each other to avoid overlaps. Default is `FALSE`.

`repel.control` A list of control settings for the repel algorithm. Ignored if `repel = FALSE`. See `ggmultiglyph.repel.control` for details on the various control parameters.

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

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

See `vignette("ggplot2-specs", package = "ggplot2")` for further details on setting these aesthetics.

The following additional aesthetics are considered if `repel = TRUE`:

- point.size
- segment.linetype
- segment.colour
- segment.size
- segment.alpha
- segment.curvature
- segment.angle
- segment.ncp
- segment.shape
- segment.square
- segment.squareShape
- segment.inflect
- segment.debug

See `ggrepel` examples page for further details on setting these aesthetics.

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.

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

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

`starglyphGrob`

Other geoms: `geom_bubbleglyph()`, `geom_dotglyph()`, `geom_flowerglyph()`, `geom_metroglyph()`, `geom_pieglyph()`, `geom_profileglyph()`, `geom_tileglyph()`

Examples

```
library(ggplot2)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Prepare the data ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Variables to map to glyphs
zs <- c("hp", "drat", "wt", "qsec", "vs", "am", "gear", "carb")

# Keep a copy of the original data
mtcars_fct <- mtcars

# Scaled numeric data
mtcars[zs] <- lapply(mtcars[zs], scales::rescale)

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

# Ordered factor data
mtcars_fct[zs[1:3]] <-
  lapply(mtcars_fct[zs[1:3]], function(x)
    ordered(cut(x, breaks = 3,
                labels = c("low", "medium", "high"))))

mtcars_fct[zs[4:8]] <-
  lapply(mtcars_fct[zs[4:8]], function(x)
    ordered(cut(x, breaks = 4,
                labels = c("tiny", "small", "medium", "large"))))

mtcars_fct$cyl <- factor(mtcars_fct$cyl)
mtcars_fct$lab <- row.names(mtcars_fct)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# 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 multivariate 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)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Repel glyphs ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

ggplot(data = mtcars) +
  geom_point(aes(x = mpg, y = disp, colour = cyl)) +
  geom_starglyph(aes(x = mpg, y = disp, fill = cyl),
                 cols = zs, whisker = TRUE, contour = TRUE,
                 size = 10, alpha = 1, repel = TRUE) +
  ylim(c(-0, 550)) +
  xlim(c(8, 35))

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# With grid points ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

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


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

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

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Legend options ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Default legend/guide
ggplot(data = mtcars) +
  geom_point(aes(x = mpg, y = disp, colour = cyl)) +
  geom_starglyph(aes(x = mpg, y = disp, fill = cyl),
                 cols = zs, whisker = TRUE, contour = TRUE,
                 size = 10, alpha = 1, repel = TRUE) +
  ylim(c(-0, 550)) +
  xlim(c(8, 35))

# Theme modifications for legend
legend_theme <-
  theme_bw(base_size = 7.5) +
  theme(legend.direction = "vertical",
        legend.box = "horizontal",
        legend.position = "bottom",
        legend.text = element_text(margin = margin(l = 7)),
        legend.key.height = unit(1.5, 'lines'))

# Glyph variable-wise legends
ggplot(data = mtcars) +
  geom_point(aes(x = mpg, y = disp, colour = cyl), show.legend = FALSE) +
  geom_starglyph(aes(x = mpg, y = disp, fill = cyl),
                 cols = zs, whisker = TRUE, contour = TRUE,
                 size = 5, alpha =  0.8, repel = TRUE) +
  ylim(c(-0, 550)) +
  scale_z_continuous(z = zs) +
```

```
  guide_z_order(z = zs, default_aes = "fill") +
  legend_theme

# Modifying the `legend.glyph.dims`
zavg <- # Scaled average of the variables
  apply(mtcars[, zs], 2,
        function(x) {
          scales::rescale(mean(x),
                          from = range(x), # same range as in scale_z_continuous
                          to = c(0, 1))
        })
zavg

ggplot(data = mtcars) +
  geom_point(aes(x = mpg, y = disp, colour = cyl)) +
  geom_starglyph(aes(x = mpg, y = disp, fill = cyl),
                 cols = zs, whisker = TRUE, contour = TRUE,
                 size = 10, alpha = 1, repel = TRUE,
                 legend.glyph.dims = zavg) +
  ylim(c(-0, 550)) +
  xlim(c(8, 35))

ggplot(data = mtcars) +
  geom_point(aes(x = mpg, y = disp, colour = cyl), show.legend = FALSE) +
  geom_starglyph(aes(x = mpg, y = disp, fill = cyl),
                 cols = zs, whisker = TRUE, contour = TRUE,
                 size = 5, alpha =  0.8, repel = TRUE,
                 legend.glyph.dims = zavg) +
  ylim(c(-0, 550)) +
  scale_z_continuous(z = zs) +
  guide_z_order(z = zs, default_aes = "fill")  +
  legend_theme

# Using custom guide
# starglyphGrob
guide_stargrob <- starglyphGrob(
  z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33, 0.6, 0.25) * 0.75,
  size = 25)
# Add labels to starglyphGrob
guide_stargrob <-
  addlabel.glyphGrob(grob = guide_stargrob, label = zs,
                     push = 1, segment = FALSE)

# Another version
guide_stargrob2 <- starglyphGrob(
  z = rep(0.5, length(zs)),
  size = 25)
guide_stargrob2 <-
  addlabel.glyphGrob(grob = guide_stargrob2, label = zs,
                     push = 1, segment = FALSE)

ggplot(data = mtcars) +
  geom_point(aes(x = mpg, y = disp, colour = cyl), show.legend = FALSE) +
```

```
  geom_starglyph(aes(x = mpg, y = disp, fill = cyl),
                 cols = zs, whisker = TRUE, contour = TRUE,
                 size = 5, alpha =  0.8, repel = TRUE) +
  ylim(c(-0, 550)) +
  guides(fill = guide_legend(order = 1, position = "right"),
         custom = guide_custom(guide_stargrob,
                               width = unit(0.1, "npc"),
                               height = unit(0.1, "npc"),
                               position = "bottom",
                               theme = theme(legend.margin = margin(t = 40, b = 30))))

ggplot(data = mtcars) +
  geom_point(aes(x = mpg, y = disp, colour = cyl), show.legend = FALSE) +
  geom_starglyph(aes(x = mpg, y = disp, fill = cyl),
                 cols = zs, whisker = TRUE, contour = TRUE,
                 size = 5, alpha =  0.8, repel = TRUE) +
  ylim(c(-0, 550)) +
  guides(fill = guide_legend(order = 1, position = "right"),
         custom = guide_custom(guide_stargrob2,
                               width = unit(0.1, "npc"),
                               height = unit(0.1, "npc"),
                               position = "bottom",
                               theme = theme(legend.margin = margin(t = 30, b = 30))))

# Legend in plots with grid points

z_grid <- c(hp = 3, drat = 3, wt = 2,
            qsec = 2, vs = 3, am = 4,
            gear = 2, carb = 3)

ggplot(data = mtcars_fct) +
  geom_starglyph(aes(x = mpg, y = disp, fill = cyl),
                 cols = zs, whisker = TRUE, contour = TRUE,
                 size = 1.5, alpha =  0.5, draw.grid = TRUE,
                 grid.point.size = 5) +
  ylim(c(-0, 550))

ggplot(data = mtcars_fct) +
  geom_starglyph(aes(x = mpg, y = disp, fill = cyl),
                 cols = zs, whisker = TRUE, contour = TRUE,
                 size = 1.5, alpha =  0.5, draw.grid = TRUE,
                 grid.point.size = 5) +
  ylim(c(-0, 550)) +
  scale_z_discrete(z = zs) +
  guide_z_order(z = zs, default_aes = "fill") +
  legend_theme

ggplot(data = mtcars_fct) +
  geom_starglyph(aes(x = mpg, y = disp, fill = cyl),
                 cols = zs, whisker = TRUE, contour = TRUE,
                 size = 1.5, alpha =  0.5, draw.grid = TRUE,
                 grid.point.size = 5,
                 legend.glyph.dims = z_grid) +
```

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

ggplot(data = mtcars_fct) +
  geom_starglyph(aes(x = mpg, y = disp, fill = cyl),
                 cols = zs, whisker = TRUE, contour = TRUE,
                 size = 2, alpha =  0.5, draw.grid = TRUE,
                 grid.point.size = 5,
                 legend.glyph.dims = z_grid) +
  ylim(c(-0, 550)) +
  scale_z_discrete(z = zs) +
  guide_z_order(z = zs, default_aes = "fill") +
  legend_theme

# starglyphGrob with grid levels

z_grid_levels <- lapply(z_grid, function(x) 1:x)

guide_stargrob_grid <-
  starglyphGrob(z = z_grid,
                size = 3, draw.grid = TRUE,
                grid.levels = z_grid_levels,
                grid.point.size = 0.1)

guide_stargrob_grid <-
  addlabel.glyphGrob(grob = guide_stargrob_grid, label = zs,
                     push = 1, segment = FALSE)

# Another version
guide_stargrob_grid2 <-
  starglyphGrob(z = rep(3, length(z_grid)),
                size = 3, draw.grid = TRUE,
                grid.levels = replicate(8, 1:3,
                                        simplify = FALSE),
                grid.point.size = 0.1)
guide_stargrob_grid2 <-
  addlabel.glyphGrob(grob = guide_stargrob_grid2, label = zs,
                     push = 1, segment = FALSE)

ggplot(data = mtcars_fct) +
  geom_starglyph(aes(x = mpg, y = disp, fill = cyl),
                 cols = zs, whisker = TRUE, contour = TRUE,
                 size = 2, alpha =  0.5, draw.grid = TRUE,
                 grid.point.size = 5,
                 legend.glyph.dims = z_grid) +
  ylim(c(-0, 550)) +
  guides(fill = guide_legend(order = 1, position = "right"),
         custom = guide_custom(guide_stargrob_grid,
                               width = unit(0.1, "npc"),
                               height = unit(0.1, "npc"),
                               position = "bottom",
                               theme = theme(legend.margin = margin(t = 20, b = 30))))

ggplot(data = mtcars_fct) +
```

```
  geom_starglyph(aes(x = mpg, y = disp, fill = cyl),
                 cols = zs, whisker = TRUE, contour = TRUE,
                 size = 2, alpha =  0.5, draw.grid = TRUE,
                 grid.point.size = 5,
                 legend.glyph.dims = z_grid) +
  ylim(c(-0, 550)) +
  guides(fill = guide_legend(order = 1, position = "right"),
         custom = guide_custom(guide_stargrob_grid2,
                               width = unit(0.1, "npc"),
                               height = unit(0.1, "npc"),
                               position = "bottom",
                               theme = theme(legend.margin = margin(t = 20, b = 30))))
```

`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),
  colour = "black",
  ratio = 1,
  nrow = 1,
  linewidth = 1,
  legend.glyph.dims = setNames(rep(0.5, length(cols)), cols),
  show.legend = NA,
  repel = FALSE,
  repel.control = ggmultiglyph.repel.control(),
  inherit.aes = TRUE
)
```

Arguments

`mapping`	Set of aesthetic mappings created by `aes()`. If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.

`data` The data to be displayed in this layer. There are three options:

If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`.

A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created.

A `function` will be called with a single argument, the plot data. The return value must be a `data.frame`, and will be used as the layer data. A `function` can be created from a `formula` (e.g. `~ head(.x, 10)`).

`stat` The statistical transformation to use on the data for this layer. When using a `geom_*()` function to construct a layer, the `stat` argument can be used to override the default coupling between geoms and stats. The `stat` argument accepts the following:

- A `Stat` ggproto subclass, for example `StatCount`.
- A string naming the stat. To give the stat as a string, strip the function name of the `stat_` prefix. For example, to use `stat_count()`, give the stat as `"count"`.
- For more information and other ways to specify the stat, see the layer stat documentation.

`position` A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The `position` argument accepts the following:

- The result of calling a position function, such as `position_jitter()`. This method allows for passing extra arguments to the position.
- A string naming the position adjustment. To give the position as a string, strip the function name of the `position_` prefix. For example, to use `position_jitter()`, give the position as `"jitter"`.
- For more information and other ways to specify the position, see the layer position documentation.

`...` 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` A character vector containing the names of at least two columns specifying the variables to be plotted in the glyphs. The selected columns must be either numeric or factor variables.

`colour` The colour of the tile glyphs.

`ratio` The aspect ratio (height / width).

`nrow` The number of rows.

`linewidth` The line width of the tile glyphs.

`legend.glyph.dims`

The dimensions of the legend glyph plot. Can be a numeric vector of unit length (where all the dimensions will have same value) or a numeric vector of same length as "cols" with the "cols" as names.

show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.
repel	logical. If TRUE, the glyphs are repel away from each other to avoid overlaps. Default is FALSE.
repel.control	A list of control settings for the repel algorithm. Ignored if repel = FALSE. See ggmultiglyph.repel.control for details on the various control parameters.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. annotation_borders().

Value

A geom layer.

Aesthetics

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

- **x**
- **y**
- alpha
- group
- size

See vignette("ggplot2-specs", package = "ggplot2") for further details on setting these aesthetics.

The following additional aesthetics are considered if repel = TRUE:

- point.size
- segment.linetype
- segment.colour
- segment.size
- segment.alpha
- segment.curvature
- segment.angle
- segment.ncp
- segment.shape
- segment.square
- segment.squareShape
- segment.inflect
- segment.debug

See ggrepel examples page for further details on setting these aesthetics.

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_bubbleglyph(), geom_dotglyph(), geom_flowerglyph(), geom_metroglyph(), geom_pieglyph(), geom_profileglyph(), geom_starglyph()

Examples

```
library(ggplot2)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Prepare the data ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Variables to map to glyphs
zs <- c("hp", "drat", "wt", "qsec", "vs", "am", "gear", "carb")

# Scaled numeric data
mtcars[zs] <- lapply(mtcars[zs], scales::rescale)

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

# Theme modifications for legend
legend_theme <-
  theme_bw(base_size = 7.5) +
  theme(legend.direction = "vertical",
        legend.box = "horizontal",
        legend.position = "bottom",
        legend.text = element_text(margin = margin(l = 7)),
        legend.key.height = unit(1.5, 'lines'))

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Palette and dimension combos ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
ggplot(data = mtcars) +
  geom_tileglyph(aes(x = mpg, y = disp),
                 cols = zs, size = 2,
                 alpha =  0.5) +
  ylim(c(-0, 550)) +
  scale_z_fill_continuous(z = zs, palette = "Blues") +
```

```
  guide_z_order(z = zs, default_aes = "fill") +
  theme_bw(base_size = 7.5) +
  legend_theme

ggplot(data = mtcars) +
  geom_tileglyph(aes(x = mpg, y = disp),
                 cols = zs, size = 2,
                 nrow = 2,
                 alpha =  0.5) +
  ylim(c(-0, 550)) +
  scale_z_fill_continuous(z = zs, palette = "Greens") +
  guide_z_order(z = zs, default_aes = "fill") +
  theme_bw(base_size = 7.5) +
  legend_theme

ggplot(data = mtcars) +
  geom_tileglyph(aes(x = mpg, y = disp),
                 cols = zs, size = 1,
                 ratio = 4,
                 alpha =  0.5) +
  ylim(c(-0, 550)) +
  scale_z_fill_continuous(z = zs, palette = "RdYlBu") +
  guide_z_order(z = zs, default_aes = "fill") +
  theme_bw(base_size = 7.5) +
  legend_theme

ggplot(data = mtcars) +
  geom_tileglyph(aes(x = mpg, y = disp),
                 cols = zs, size = 1,
                 ratio = 4, nrow = 2,
                 alpha =  0.5) +
  ylim(c(-0, 550)) +
  scale_z_fill_continuous(z = zs, palette = "viridis") +
  guide_z_order(z = zs, default_aes = "fill") +
  theme_bw(base_size = 7.5) +
  legend_theme


#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Repel glyphs ----
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
ggplot(data = mtcars) +
  geom_point(aes(x = mpg, y = disp)) +
  geom_tileglyph(aes(x = mpg, y = disp),
                 cols = zs, size = 2,
                 alpha =  1, repel = TRUE) +
  ylim(c(-0, 550)) +
  scale_z_fill_continuous(z = zs, palette = "Blues") +
  guide_z_order(z = zs, default_aes = "fill") +
  theme_bw(base_size = 7.5) +
  legend_theme

ggplot(data = mtcars) +
```

```
  geom_point(aes(x = mpg, y = disp)) +
  geom_tileglyph(aes(x = mpg, y = disp),
                 cols = zs, size = 1,
                 ratio = 4, nrow = 2,
                 alpha =  1, repel = TRUE) +
  ylim(c(-0, 550)) +
  scale_z_fill_continuous(z = zs, palette = "viridis") +
  guide_z_order(z = zs, default_aes = "fill") +
  theme_bw(base_size = 7.5) +
  legend_theme
```

`ggmultiglyph.repel.control` *Control Parameters for the Repel Algorithm*

Description

Set the contol parameters for the repel algorithm (Slowikowski 2021) used in various geoms (`geom_*()`) implemented in `ggmultiglyph` to repel the glyphs from each other.

Usage

```
ggmultiglyph.repel.control(
  box.padding = 0.25,
  point.padding = 0.000001,
  min.segment.length = 0.5,
  arrow = NULL,
  force = 1,
  force_pull = 1,
  max.time = 0.5,
  max.iter = 10000,
  max.overlaps = getOption("ggrepel.max.overlaps", default = 10),
  nudge_x = 0,
  nudge_y = 0,
  xlim = c(NA, NA),
  ylim = c(NA, NA),
  direction = c("both", "y", "x"),
  seed = NA,
  verbose = FALSE,
  ...
)
```

Arguments

`box.padding`	Amount of padding around bounding box, as unit or number. Defaults to 0.25. (Default unit is lines, but other units can be specified by passing `unit(x, "units")`).

point.padding Amount of padding around labeled point, as unit or number. Defaults to 0. (Default unit is lines, but other units can be specified by passing `unit(x, "units")`).

min.segment.length
Skip drawing segments shorter than this, as unit or number. Defaults to 0.5. (Default unit is lines, but other units can be specified by passing `unit(x, "units")`).

arrow specification for arrow heads, as created by `arrow`.

force Force of repulsion between overlapping glyphs. Defaults to 1.

force_pull Force of attraction between a glyph and its corresponding data point. Defaults to 1.

max.time Maximum number of seconds to try to resolve overlaps. Defaults to 0.5.

max.iter Maximum number of iterations to try to resolve overlaps. Defaults to 10000.

max.overlaps Exclude glyphs that overlap too many things. Defaults to 10.

nudge_x, nudge_y
Horizontal and vertical adjustments to nudge the starting position of each glyph. The units for `nudge_x` and `nudge_y` are the same as for the data units on the x-axis and y-axis.

xlim, ylim Limits for the x and y axes. Glyphs will be constrained to these limits. By default, glyphs are constrained to the entire plot area.

direction "both", "x", or "y" – direction in which to adjust position of glyphs.

seed Random seed passed to `set.seed`. Defaults to `NA`, which means that `set.seed` will not be called.

verbose If `TRUE`, some diagnostics of the repel algorithm are printed.

... other repel arguments such as segment controls.

Value

A list with the following components to control the repel algorithm corresponding to the same in **Arguments**.

box.padding

point.padding
min.segment.length

arrow

force

force_pull

max.time

max.iter

max.overlaps
nudge_x, nudge_y

xlim, ylim

direction

seed

verbose

References

Slowikowski K (2021). "ggrepel: Automatically position non-overlapping text labels with 'ggplot2'. R package version 0.9.1."

See Also

ggrepel

Examples

```
# Adjust force
ggmultiglyph.repel.control(force = 0.5)

# Adjust max.iter
ggmultiglyph.repel.control(max.iter = 5000)
```

---

ggmultiglyph.segment.control

Control Parameters for Drawing Curved Connecting Segments

Description

Set the contol parameters for drawing xurved segments connecting grobs to labels in addlabel.glyphGrob.

Usage

```
ggmultiglyph.segment.control(
  segment.curvature = 0,
  segment.angle = 90,
  segment.ncp = 1,
  segment.shape = 0.5,
  segment.square = TRUE,
  segment.squareShape = 1,
  segment.inflect = FALSE,
  segment.debug = FALSE,
  segment.colour = "black",
  segment.size = 0.2,
  segment.linetype = 1,
  segment.arrow = NULL
)
```

Arguments

`segment.curvature`	Numeric, negative for left-hand and positive for right-hand curves, 0 for straight lines. Defaults to 0.
`segment.angle`	0-180, less than 90 skews control points toward the start point. Defaults to 90.
`segment.ncp`	Number of control points to make a smoother curve. Defaults to 1.
`segment.shape`	Curve shape by control points approximation/interpolation (1 for cubic B-spline, -1 for Catmull-Rom spline). Defaults to 0.5.
`segment.square`	`TRUE` to place control points in city-block fashion, `FALSE` for oblique placement. Default is `TRUE`.
`segment.squareShape`	Shape of the curve relative to additional control points inserted if square is `TRUE`. Defaults to 1.
`segment.inflect`	Curve inflection at the midpoint. Default is `FALSE`.
`segment.debug`	Display the curve debug information. Default is `FALSE`.
`segment.colour`	The line segment color. Default is `"black"`.
`segment.size`	The line segment thickness. Default is 0.5 mm.
`segment.linetype`	The line segment solid, dashed, etc. Default is 1 (solid).
`segment.arrow`	Render line segment as an arrow with `grid::arrow()`.

Value

A list with the following components to control the xurved segments corresponding to the same in **Arguments**.

`segment.curvature`

`segment.angle`

`segment.ncp`

`segment.shape`

`segment.square`

`segment.squareShape`

`segment.inflect`

`segment.debug`

`segment.colour`

`segment.size`

`segment.linetype`

`segment.arrow`

See Also

addlabel.glyphGrob

---

guide_z_order *Set guide order for* col *aesthetics*

Description

Helper function to preserve the display order of legends associated with multiple col aesthetics.

Usage

```
guide_z_order(z, default_aes = "fill")
```

Arguments

z	A character vector of aesthetic names whose legend order should be preserved.
default_aes	A character scalar giving the primary ggplot2 aesthetic whose legend should appear first. Defaults to "fill".

Details

This function is primarily intended for use with ggmultiglyph scales such as scale_z_continuous(). Internally, **ggplot2** may not preserve the order of aesthetics supplied to a scale when guides are assembled during plot construction (via the internal guide setup machinery i.e. ggplot2:::Guides$setup() in ggplot2 v 4.0.0). As a result, legends corresponding to z aesthetics can appear in an order that differs from the order specified by the user.

guide_z_order() constructs a guides() specification that explicitly assigns guide orders, ensuring that the default ggplot2 aesthetic legend (FOr example, "fill") is displayed first, followed by the supplied z aesthetics in the order given.

Value

A guides object that can be added to a ggplot.

See Also

guides, guide_legend

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"))


# Theme for proper display of all the guides
guide_theme <-
  theme(legend.direction = "vertical",
        legend.box = "horizontal",
        legend.position = "bottom",
        legend.text = element_text(margin = margin(l = 7)),
        legend.key.height = unit(1.5, 'lines'))


# Order of guides is scrambled
ggplot(data = mtcars) +
  geom_starglyph(aes(x = mpg, y = disp, fill = cyl),
                 cols = zs, whisker = TRUE, contour = TRUE,
                 size = 5, alpha = 0.8) +
  ylim(c(-0, 550)) +
  scale_z_continuous(z = zs) +
  theme_bw(base_size = 7.5) +
  guide_theme


# Fixing the order of guides by individual assignment
ggplot(data = mtcars) +
  geom_starglyph(aes(x = mpg, y = disp, fill = cyl),
                 cols = zs, whisker = TRUE, contour = TRUE,
                 size = 5, alpha =  0.8) +
  ylim(c(-0, 550)) +
  scale_z_continuous(z = zs) +
  theme_bw(base_size = 7.5) +
  guides(fill = guide_legend(order = 1),
         hp = guide_legend(order = 2),
         drat = guide_legend(order = 3),
         wt = guide_legend(order = 4),
         qsec = guide_legend(order = 5),
         vs = guide_legend(order = 6),
         am = guide_legend(order = 7),
         gear = guide_legend(order = 8),
         carb = guide_legend(order = 9)) +
  guide_theme

# Work-around using guide_z_order
ggplot(data = mtcars) +
  geom_starglyph(aes(x = mpg, y = disp, fill = cyl),
                 cols = zs, whisker = TRUE, contour = TRUE,
                 size = 5, alpha = 0.8) +
  ylim(c(-0, 550)) +
  scale_z_continuous(z = zs) +
  guide_z_order(z = zs, default_aes = "fill") +
```

```
  theme_bw(base_size = 7.5) +
  guide_theme
```

---

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,
  grid.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 (radius).
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`.
`grid.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: `bubbleglyphGrob()`, `dotglyphGrob()`, `flowerglyphGrob()`, `pieglyphGrob()`, `profileglyphGrob()`, `starglyphGrob()`, `tileglyphGrob()`

Examples

```
library(ggmultiglyph)
library(grid)
library(gridExtra)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust ray size
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

mg1 <- metroglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                      z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 10)

mg2 <- metroglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                      z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15)

mg3 <- metroglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                      z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 20)
```

```
grid.arrange(mg1, mg2, mg3, nrow = 1)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust circle size
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

mg1 <- metroglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                      z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                      circle.size = 0, size = 15)

mg2 <- metroglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                      z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                      circle.size = 2, size = 15)

mg3 <- metroglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                      z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                      circle.size = 5, size = 15)

grid.arrange(mg1, mg2, mg3, nrow = 1)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust angle
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

mg1 <- metroglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                      z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                      size = 15, circle.size = 2)

mg2 <- metroglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                      z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                      size = 15, circle.size = 2,
                      angle.start = 0, angle.stop = base::pi)

mg3 <- metroglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                      z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                      size = 15, circle.size = 2,
                      angle.start = 90 * (base::pi/180),
                      angle.stop = 270 * (base::pi/180))

mg4 <- metroglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                      z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                      size = 15, circle.size = 2,
                      angle.start = 2 * base::pi,
                      angle.stop = 0)

mg5 <- metroglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                      z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                      size = 15, circle.size = 2,
                      angle.start = base::pi, angle.stop = 0)

mg6 <- metroglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                      z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
```

```
                      size = 15, circle.size = 2,
                      angle.start = 270 * (base::pi/180),
                      angle.stop = 90 * (base::pi/180))

grid.arrange(mg1, mg2, mg3, mg4, mg5, mg6, nrow = 2, ncol = 3)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust circle and ray line width
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

mg1 <- metroglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                      z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15)

mg2 <- metroglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                      z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                      lwd.ray = 3, lwd.circle = 0.1)

mg3 <- metroglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                      z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                      lwd.ray = 0.1, lwd.circle = 3)

mg4 <- metroglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                      z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                      angle.start = 0, angle.stop = base::pi)

mg5 <- metroglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                      z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                      lwd.ray = 3, lwd.circle = 0.1,
                      angle.start = 0, angle.stop = base::pi)

mg6 <- metroglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                      z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                      lwd.ray = 0.1, lwd.circle = 3,
                      angle.start = 0, angle.stop = base::pi)

grid.arrange(mg1, mg2, mg3, mg4, mg5, mg6, nrow = 2, ncol = 3)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust fill colour
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

mg1 <- metroglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                      z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                      fill = "salmon")

mg2 <- metroglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                      z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                      lwd.ray = 3, lwd.circle = 0.1,
                      fill = "cyan")

mg3 <- metroglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                      z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                      lwd.ray = 0.1, lwd.circle = 3,
```

```
                      fill = "green")

mg4 <- metroglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                      z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                      angle.start = 0, angle.stop = base::pi,
                      fill = "salmon")

mg5 <- metroglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                      z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                      lwd.ray = 3, lwd.circle = 0.1,
                      angle.start = 0, angle.stop = base::pi,
                      fill = "cyan")

mg6 <- metroglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                      z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                      lwd.ray = 0.1, lwd.circle = 3,
                      angle.start = 0, angle.stop = base::pi,
                      fill = "green")

grid.arrange(mg1, mg2, mg3, mg4, mg5, mg6, nrow = 2, ncol = 3)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust ray colour
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

mg1 <- metroglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                      z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                      lwd.ray = 3,
                      col.ray = "salmon")

mg2 <- metroglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                      z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                      lwd.ray = 3,
                      col.ray = "cyan")

mg3 <- metroglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                      z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                      lwd.ray = 3,
                      col.ray = "green")

mg4 <- metroglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                      z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                      lwd.ray = 3,
                      angle.start = 0, angle.stop = base::pi,
                      col.ray = "salmon")

mg5 <- metroglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                      z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                      lwd.ray = 3,
                      angle.start = 0, angle.stop = base::pi,
                      col.ray = "cyan")

mg6 <- metroglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
```

```
                      z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                      lwd.ray = 3,
                      angle.start = 0, angle.stop = base::pi,
                      col.ray = "green")

grid.arrange(mg1, mg2, mg3, mg4, mg5, mg6, nrow = 2, ncol = 3)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Multivariate ray colour
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

mg1 <- metroglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                      z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                      col.ray = RColorBrewer::brewer.pal(6, "Dark2"),
                      col.circle = "gray")

mg2 <- metroglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                      z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                      lwd.ray = 3, lwd.circle = 0.1,
                      col.ray = RColorBrewer::brewer.pal(6, "Dark2"),
                      col.circle = "gray")

mg3 <- metroglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                      z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                      lwd.ray = 0.1, lwd.circle = 3,
                      col.ray = RColorBrewer::brewer.pal(6, "Dark2"),
                      col.circle = "gray")

mg4 <- metroglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                      z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                      angle.start = 0, angle.stop = base::pi,
                      col.ray = RColorBrewer::brewer.pal(6, "Dark2"),
                      col.circle = "gray")

mg5 <- metroglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                      z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                      lwd.ray = 3, lwd.circle = 0.1,
                      angle.start = 0, angle.stop = base::pi,
                      col.ray = RColorBrewer::brewer.pal(6, "Dark2"),
                      col.circle = "gray")

mg6 <- metroglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                      z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                      lwd.ray = 0.1, lwd.circle = 3,
                      angle.start = 0, angle.stop = base::pi,
                      col.ray = RColorBrewer::brewer.pal(6, "Dark2"),
                      col.circle = "gray")

grid.arrange(mg1, mg2, mg3, mg4, mg5, mg6, nrow = 2, ncol = 3)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust line end style
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
mg1 <- metroglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                      z = c(0.28, 0.33, 0.8, 1.2, 0.6, 0.5, 0.7), size = 15,
                      lwd.ray = 10)

mg2 <- metroglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                      z = c(0.28, 0.33, 0.8, 1.2, 0.6, 0.5, 0.7), size = 15,
                      lwd.ray = 10, lineend = "butt")

mg3 <- metroglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                      z = c(0.28, 0.33, 0.8, 1.2, 0.6, 0.5, 0.7), size = 15,
                      lwd.ray = 10, lineend = "square")

grid.arrange(mg1, mg2, mg3, nrow = 1, ncol = 3)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust grid levels
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Grid levels
gl <- split(x = c(rep(c(1, 2, 3), 4),
                  rep(c(1, 2, 3, 4), 2)),
            f = c(rep(1:4, each = 3),
                  rep(5:6, each = 4)))
gl

mg1 <- metroglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                      z = c(1, 3, 2, 1, 2, 3), size = 5,
                      draw.grid = TRUE, grid.levels = gl,
                      grid.point.size = 0.01)

mg2 <- metroglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                      z = c(1, 3, 2, 1, 2, 3), size = 5,
                      lwd.ray = 3, col.points = "white",
                      draw.grid = TRUE, grid.levels = gl,
                      grid.point.size = 0.05)

mg3 <- metroglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                      z = c(1, 3, 2, 1, 2, 3), size = 5,
                      draw.grid = FALSE, grid.levels = gl,
                      grid.point.size = 0.05)

mg4 <- metroglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                      z = c(1, 3, 2, 1, 2, 3), size = 5,
                      angle.start = 0, angle.stop = base::pi,
                      draw.grid = TRUE, grid.levels = gl,
                      grid.point.size = 0.01)

mg5 <- metroglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                      z = c(1, 3, 2, 1, 2, 3), size = 5,
                      lwd.ray = 3,
                      angle.start = 0, angle.stop = base::pi,
                      draw.grid = TRUE, grid.levels = gl,
```

```
                      grid.point.size = 0.05)

mg6 <- metroglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                      z = c(1, 3, 2, 1, 2, 3), size = 5,
                      angle.start = 0, angle.stop = base::pi,
                      draw.grid = FALSE, grid.levels = gl)

grid.arrange(mg1, mg2, mg3, mg4, mg5, mg6, nrow = 2, ncol = 3)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust fill, ray and grid level colours
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

mg1 <- metroglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                      z = c(1, 3, 2, 1, 2, 3), size = 5,
                      col.ray = RColorBrewer::brewer.pal(6, "Dark2"),
                      draw.grid = TRUE, grid.levels = gl,
                      col.points = NA,
                      grid.point.size = 0.05)

mg2 <- metroglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                      z = c(1, 3, 2, 1, 2, 3), size = 5,
                      lwd.ray = 3,
                      col.ray = RColorBrewer::brewer.pal(6, "Dark2"),
                      draw.grid = TRUE, grid.levels = gl,
                      grid.point.size = 0.05)

mg3 <- metroglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                      z = c(1, 3, 2, 1, 2, 3), size = 5,
                      circle.size = 0, col.points = "white",
                      lwd.ray = 3,
                      draw.grid = TRUE, grid.levels = gl,
                      col.ray = RColorBrewer::brewer.pal(6, "Dark2"),
                      grid.point.size = 0.05)

mg4 <- metroglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                      z = c(1, 3, 2, 1, 2, 3), size = 5,
                      angle.start = 0, angle.stop = base::pi,
                      draw.grid = TRUE, grid.levels = gl,
                      grid.point.size = 0.01)

mg5 <- metroglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                      z = c(1, 3, 2, 1, 2, 3), size = 5,
                      col.circle = "gray",
                      col.ray = RColorBrewer::brewer.pal(6, "Dark2"),
                      angle.start = 0, angle.stop = base::pi,
                      draw.grid = TRUE, grid.levels = gl,
                      grid.point.size = 0.03, col.points = "gray")

mg6 <- metroglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                      z = c(1, 3, 2, 1, 2, 3), size = 5,
                      angle.start = 0, angle.stop = base::pi,
                      draw.grid = FALSE, grid.levels = gl,
```

```
                    col.ray = RColorBrewer::brewer.pal(6, "Dark2"),
                    grid.point.size = 0.05)

grid.arrange(mg1, mg2, mg3, mg4, mg5, mg6, nrow = 2, ncol = 3)
```

`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 (radius).
`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: `bubbleglyphGrob()`, `dotglyphGrob()`, `flowerglyphGrob()`, `metroglyphGrob()`, `profileglyphGrob()`, `starglyphGrob()`, `tileglyphGrob()`

Examples

```
library(ggmultiglyph)
library(grid)
library(gridExtra)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust radius and segment scaling
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
pg1 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                    size = 15)

pg2 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                    size = 15, scale.radius = FALSE)

pg3 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                    size = 15, scale.segment = TRUE, scale.radius = FALSE)

pg4 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                     size = 15,
                     angle.start = 0, angle.stop = base::pi)

pg5 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                    size = 15, scale.radius = FALSE,
                    angle.start = 0, angle.stop = base::pi)

pg6 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                    size = 15, scale.segment = TRUE, scale.radius = FALSE,
                    angle.start = 0, angle.stop = base::pi)

grid.arrange(pg1, pg2, pg3, pg4, pg5, pg6, nrow = 2, ncol = 3)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust size
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

pg1 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 5)

pg2 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 10)

pg3 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15)

pg4 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 5,
                     angle.start = 0, angle.stop = base::pi)

pg5 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 10,
                     angle.start = 0, angle.stop = base::pi)

pg6 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                     angle.start = 0, angle.stop = base::pi)
```

```
grid.arrange(pg1, pg2, pg3, pg4, pg5, pg6, nrow = 2, ncol = 3)

pg1 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 5,
                    scale.radius = FALSE)

pg2 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 10,
                    scale.radius = FALSE)

pg3 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                    scale.radius = FALSE)

pg4 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 5,
                    scale.radius = FALSE,
                    angle.start = 0, angle.stop = base::pi)

pg5 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 10,
                    scale.radius = FALSE,
                    angle.start = 0, angle.stop = base::pi)

pg6 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                    scale.radius = FALSE,
                    angle.start = 0, angle.stop = base::pi)

grid.arrange(pg1, pg2, pg3, pg4, pg5, pg6, nrow = 2, ncol = 3)

pg1 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 5,
                    scale.segment = TRUE, scale.radius = FALSE)

pg2 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 10,
                    scale.segment = TRUE, scale.radius = FALSE)

pg3 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                    scale.segment = TRUE, scale.radius = FALSE)

pg4 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 5,
                    scale.segment = TRUE, scale.radius = FALSE,
                    angle.start = 0, angle.stop = base::pi)

pg5 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 10,
                    scale.segment = TRUE, scale.radius = FALSE,
                    angle.start = 0, angle.stop = base::pi)
```

```
pg6 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                    scale.segment = TRUE, scale.radius = FALSE,
                    angle.start = 0, angle.stop = base::pi)

grid.arrange(pg1, pg2, pg3, pg4, pg5, pg6, nrow = 2, ncol = 3)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust line width
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

pg1 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15)

pg2 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                    size = 15, lwd = 3)

pg3 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                    size = 15, lwd = 5)

pg4 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                    angle.start = 0, angle.stop = base::pi)

pg5 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                    size = 15, lwd = 3,
                    angle.start = 0, angle.stop = base::pi)

pg6 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                    size = 15, lwd = 5,
                    angle.start = 0, angle.stop = base::pi)

grid.arrange(pg1, pg2, pg3, pg4, pg5, pg6, nrow = 2, ncol = 3)

pg1 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                    scale.radius = FALSE)

pg2 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                    size = 15, lwd = 3,
                    scale.radius = FALSE)

pg3 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                    size = 15, lwd = 5,
                    scale.radius = FALSE)
```

```
pg4 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                    scale.radius = FALSE,
                    angle.start = 0, angle.stop = base::pi)

pg5 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                    size = 15, lwd = 3,
                    scale.radius = FALSE,
                    angle.start = 0, angle.stop = base::pi)

pg6 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                    size = 15, lwd = 5,
                    scale.radius = FALSE,
                    angle.start = 0, angle.stop = base::pi)

grid.arrange(pg1, pg2, pg3, pg4, pg5, pg6, nrow = 2, ncol = 3)

pg1 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                    scale.segment = TRUE, scale.radius = FALSE)

pg2 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                    size = 15, lwd = 3,
                    scale.segment = TRUE, scale.radius = FALSE)

pg3 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                    size = 15, lwd = 5,
                    scale.segment = TRUE, scale.radius = FALSE)

pg4 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                    scale.segment = TRUE, scale.radius = FALSE,
                    angle.start = 0, angle.stop = base::pi)

pg5 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                    size = 15, lwd = 3,
                    scale.segment = TRUE, scale.radius = FALSE,
                    angle.start = 0, angle.stop = base::pi)

pg6 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                    size = 15, lwd = 5,
                    scale.segment = TRUE, scale.radius = FALSE,
                    angle.start = 0, angle.stop = base::pi)

grid.arrange(pg1, pg2, pg3, pg4, pg5, pg6, nrow = 2, ncol = 3)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
# Adjust angle
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

pg1 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 10)

pg2 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 10,
                    angle.start = 0, angle.stop = base::pi)

pg3 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 10,
                    angle.start = 90 * (base::pi/180),
                    angle.stop = 270 * (base::pi/180))

pg4 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 10,
                    scale.radius = FALSE)

pg5 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 10,
                    scale.radius = FALSE,
                    angle.start = 0, angle.stop = base::pi)

pg6 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 10,
                    scale.radius = FALSE,
                    angle.start = 90 * (base::pi/180),
                    angle.stop = 270 * (base::pi/180))

pg7 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 10,
                    scale.segment = TRUE, scale.radius = FALSE)

pg8 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 10,
                    scale.segment = TRUE, scale.radius = FALSE,
                    angle.start = 0, angle.stop = base::pi)

pg9 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 10,
                    scale.segment = TRUE, scale.radius = FALSE,
                    angle.start = 90 * (base::pi/180),
                    angle.stop = 270 * (base::pi/180))

grid.arrange(pg1, pg2, pg3, pg4, pg5, pg6, pg7, pg8, pg9,
             nrow = 3, ncol = 3)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust fill colour
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
pg1 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                    size = 10, fill = "salmon")

pg2 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                    angle.start = 0, angle.stop = base::pi,
                    size = 10, fill = "salmon")

pg3 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                    angle.start = 90 * (base::pi/180),
                    angle.stop = 270 * (base::pi/180),
                    size = 10, fill = "salmon")

pg4 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                    size = 10, fill = "cyan")

pg5 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                    angle.start = 0, angle.stop = base::pi,
                    size = 10, fill = "cyan")

pg6 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                    angle.start = 90 * (base::pi/180),
                    angle.stop = 270 * (base::pi/180),
                    size = 10, fill = "cyan")

pg7 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                    size = 10, fill = "green")

pg8 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                    angle.start = 0, angle.stop = base::pi,
                    size = 10, fill = "green")

pg9 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                    angle.start = 90 * (base::pi/180),
                    angle.stop = 270 * (base::pi/180),
                    size = 10, fill = "green")

grid.arrange(pg1, pg2, pg3, pg4, pg5, pg6, pg7, pg8, pg9,
             nrow = 3, ncol = 3)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust line colour
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

pg1 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
```

```
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                    size = 10, lwd = 3, col = "salmon")

pg2 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                    angle.start = 0, angle.stop = base::pi,
                    size = 10, lwd = 3, col = "salmon")

pg3 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                    angle.start = 90 * (base::pi/180),
                    angle.stop = 270 * (base::pi/180),
                    size = 10, lwd = 3, col = "salmon")

pg4 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                    size = 10, lwd = 3, col = "cyan")

pg5 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                    angle.start = 0, angle.stop = base::pi,
                    size = 10, lwd = 3, col = "cyan")

pg6 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                    angle.start = 90 * (base::pi/180),
                    angle.stop = 270 * (base::pi/180),
                    size = 10, lwd = 3, col = "cyan")

pg7 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                    size = 10, lwd = 3, col = "green")

pg8 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                    angle.start = 0, angle.stop = base::pi,
                    size = 10, lwd = 3, col = "green")

pg9 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                    angle.start = 90 * (base::pi/180),
                    angle.stop = 270 * (base::pi/180),
                    size = 10, lwd = 3, col = "green")

grid.arrange(pg1, pg2, pg3, pg4, pg5, pg6, pg7, pg8, pg9,
             nrow = 3, ncol = 3)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Multivariate fill colour
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

pg1 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 10,
```

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

pg2 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 10,
                    angle.start = 0, angle.stop = base::pi,
                    fill = RColorBrewer::brewer.pal(6, "Dark2"))

pg3 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 10,
                    angle.start = 90 * (base::pi/180),
                    angle.stop = 270 * (base::pi/180),
                    fill = RColorBrewer::brewer.pal(6, "Dark2"))

pg4 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 10,
                    scale.radius = FALSE,
                    fill = RColorBrewer::brewer.pal(6, "Dark2"))

pg5 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 10,
                    scale.radius = FALSE,
                    angle.start = 0, angle.stop = base::pi,
                    fill = RColorBrewer::brewer.pal(6, "Dark2"))

pg6 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 10,
                    scale.radius = FALSE,
                    angle.start = 90 * (base::pi/180),
                    angle.stop = 270 * (base::pi/180),
                    fill = RColorBrewer::brewer.pal(6, "Dark2"))

pg7 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 10,
                    scale.segment = TRUE, scale.radius = FALSE,
                    fill = RColorBrewer::brewer.pal(6, "Dark2"))

pg8 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 10,
                    scale.segment = TRUE, scale.radius = FALSE,
                    angle.start = 0, angle.stop = base::pi,
                    fill = RColorBrewer::brewer.pal(6, "Dark2"))

pg9 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 10,
                    scale.segment = TRUE, scale.radius = FALSE,
                    angle.start = 90 * (base::pi/180),
                    angle.stop = 270 * (base::pi/180),
                    fill = RColorBrewer::brewer.pal(6, "Dark2"))

grid.arrange(pg1, pg2, pg3, pg4, pg5, pg6, pg7, pg8, pg9,
             nrow = 3, ncol = 3)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
# Adjust line join style
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

pg1 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                    size = 15, lwd = 5)

pg2 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                    size = 15, lwd = 5, linejoin = "round")

pg3 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                    size = 15, lwd = 5, linejoin = "bevel")

grid.arrange(pg1, pg2, pg3, nrow = 1, ncol = 3)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust grid levels (when scale.radius = TRUE)
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Grid levels
gl <- split(x = c(rep(c(1, 2, 3), 4),
                  rep(c(1, 2, 3, 4), 2)),
            f = c(rep(1:4, each = 3),
                  rep(5:6, each = 4)))
gl

pg1 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(1, 3, 2, 1, 2, 3), size = 5,
                    draw.grid = TRUE, grid.levels = gl,
                    lwd = 2, col.grid = "black")

pg2 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    angle.start = 0, angle.stop = base::pi,
                    z = c(1, 3, 2, 1, 2, 3), size = 5,
                    draw.grid = TRUE, grid.levels = gl,
                    lwd = 2, col.grid = "black")

grid.arrange(pg1, pg2, ncol = 2)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust grid level colours (when scale.radius = TRUE)
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Grid levels
gl <- split(x = c(rep(c(1, 2, 3), 4),
                  rep(c(1, 2, 3, 4), 2)),
            f = c(rep(1:4, each = 3),
                  rep(5:6, each = 4)))
gl

pg1 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
```

```
                    z = c(1, 3, 2, 1, 2, 3), size = 5,
                    draw.grid = TRUE, grid.levels = gl,
                    lwd = 2, col = "white", col.grid = "white",
                    fill = RColorBrewer::brewer.pal(6, "Dark2"))

pg2 <- pieglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    angle.start = 0, angle.stop = base::pi,
                    z = c(1, 3, 2, 1, 2, 3), size = 5,
                    draw.grid = TRUE, grid.levels = gl,
                    lwd = 2, col = "white", col.grid = "white",
                    fill = RColorBrewer::brewer.pal(6, "Dark2"))

grid.arrange(pg1, pg2, ncol = 2)
```

---

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: bubbleglyphGrob(), dotglyphGrob(), flowerglyphGrob(), metroglyphGrob(), pieglyphGrob(), starglyphGrob(), tileglyphGrob()

Examples

```
library(ggmultiglyph)
library(grid)
library(gridExtra)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Profile bar, line, mirror and flip.axes combination
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

barglyph1 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 10, line = FALSE)

profileglyph1 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 10, line = TRUE, bar = FALSE)

barprofileglyph1 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 10)

barglyph2 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 10, line = FALSE, mirror = FALSE)

profileglyph2 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 10, , mirror = FALSE,
                   line = TRUE, bar = FALSE)

barprofileglyph2 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 10, mirror = FALSE)

barglyph3 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 10, line = FALSE, flip.axes = TRUE)

profileglyph3 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 10, line = TRUE, bar = FALSE,
                   flip.axes = TRUE)

barprofileglyph3 <-
```

```
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 10, flip.axes = TRUE)

barglyph4 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 10, line = FALSE,
                   mirror = FALSE, flip.axes = TRUE)

profileglyph4 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 10, , mirror = FALSE, flip.axes = TRUE,
                   line = TRUE, bar = FALSE)

barprofileglyph4 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 10, mirror = FALSE, flip.axes = TRUE)

grid.arrange(barglyph1, profileglyph1, barprofileglyph1,
             barglyph2, profileglyph2, barprofileglyph2,
             nrow = 2, ncol = 3)
grid.arrange(barglyph3, profileglyph3, barprofileglyph3,
             barglyph4, profileglyph4, barprofileglyph4,
             nrow = 2, ncol = 3)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust size
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

barglyph1 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 5, line = FALSE)

profileglyph1 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 5, line = TRUE, bar = FALSE)

barprofileglyph1 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 5)

barglyph2 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 15, line = FALSE)

barprofileglyph2 <-
```

```
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 15)

profileglyph2 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 15, line = TRUE, bar = FALSE)

barglyph3 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 20, line = FALSE)

barprofileglyph3 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 20)

profileglyph3 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 20, line = TRUE, bar = FALSE)

grid.arrange(barglyph1, barglyph2, barglyph3, nrow = 1)
grid.arrange(profileglyph1, profileglyph2, profileglyph3, nrow = 1)
grid.arrange(barprofileglyph1, barprofileglyph2, barprofileglyph3, nrow = 1)

barglyph1 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 5, line = FALSE, mirror = FALSE)

profileglyph1 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 5, line = TRUE,
                   bar = FALSE, mirror = FALSE)

barprofileglyph1 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 5, mirror = FALSE)

barglyph2 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 15, line = FALSE, mirror = FALSE)

profileglyph2 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 15, line = TRUE,
```

```
                  bar = FALSE, mirror = FALSE)

barprofileglyph2 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                  z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                  size = 15, mirror = FALSE)

barglyph3 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                  z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                  size = 20, line = FALSE, mirror = FALSE)

profileglyph3 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                  z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                  size = 20, line = TRUE,
                  bar = FALSE, mirror = FALSE)

barprofileglyph3 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                  z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                  size = 20, mirror = FALSE)

grid.arrange(barglyph1, barglyph2, barglyph3, nrow = 1)
grid.arrange(profileglyph1, profileglyph2, profileglyph3, nrow = 1)
grid.arrange(barprofileglyph1, barprofileglyph2, barprofileglyph3, nrow = 1)

barglyph1 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                  z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                  size = 5, line = FALSE, flip.axes = TRUE)

profileglyph1 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                  z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                  size = 5, line = TRUE, bar = FALSE,
                  flip.axes = TRUE)

barprofileglyph1 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                  z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                  size = 5, flip.axes = TRUE)

barglyph2 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                  z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                  size = 15, line = FALSE, flip.axes = TRUE)

barprofileglyph2 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                  z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                  size = 15, flip.axes = TRUE)
```

```
profileglyph2 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 15, line = TRUE, bar = FALSE,
                   flip.axes = TRUE)

barglyph3 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 20, line = FALSE, flip.axes = TRUE)

profileglyph3 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 20, line = TRUE, bar = FALSE,
                   flip.axes = TRUE)

barprofileglyph3 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 20, flip.axes = TRUE)

grid.arrange(barglyph1, barglyph2, barglyph3, nrow = 1)
grid.arrange(profileglyph1, profileglyph2, profileglyph3, nrow = 1)
grid.arrange(barprofileglyph1, barprofileglyph2, barprofileglyph3, nrow = 1)

barglyph1 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 5, line = FALSE,
                   mirror = FALSE, flip.axes = TRUE)

profileglyph1 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 5, line = TRUE,
                   bar = FALSE, mirror = FALSE, flip.axes = TRUE)

barprofileglyph1 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 5, mirror = FALSE, flip.axes = TRUE)

barglyph2 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 15, line = FALSE,
                   mirror = FALSE, flip.axes = TRUE)

profileglyph2 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 15, line = TRUE,
```

```
                  bar = FALSE, mirror = FALSE, flip.axes = TRUE)

barprofileglyph2 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                  z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                  size = 15, mirror = FALSE, flip.axes = TRUE)

barglyph3 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                  z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                  size = 20, line = FALSE,
                  mirror = FALSE, flip.axes = TRUE)

profileglyph3 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                  z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                  size = 20, line = TRUE,
                  bar = FALSE, mirror = FALSE, flip.axes = TRUE)

barprofileglyph3 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                  z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                  size = 20, mirror = FALSE, flip.axes = TRUE)

grid.arrange(barglyph1, barglyph2, barglyph3, nrow = 1)
grid.arrange(profileglyph1, profileglyph2, profileglyph3, nrow = 1)
grid.arrange(barprofileglyph1, barprofileglyph2, barprofileglyph3, nrow = 1)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust width
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

barglyph1 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                  z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                  size = 10, width = 1, line = FALSE)

profileglyph1 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                  z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                  size = 10, width = 1, line = TRUE, bar = FALSE)

barprofileglyph1 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                  z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                  size = 10, width = 1)

barglyph2 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                  z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                  size = 10, width = 3, line = FALSE)

barprofileglyph2 <-
```

```
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 10, width = 3)

profileglyph2 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 10, width = 3, line = TRUE, bar = FALSE)

barglyph3 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 10, line = FALSE)

barprofileglyph3 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 10)

profileglyph3 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 10, line = TRUE, bar = FALSE)

grid.arrange(barglyph1, barglyph2, barglyph3, nrow = 1)
grid.arrange(profileglyph1, profileglyph2, profileglyph3, nrow = 1)
grid.arrange(barprofileglyph1, barprofileglyph2, barprofileglyph3, nrow = 1)

barglyph1 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 10, width = 1, line = FALSE, mirror = FALSE)

profileglyph1 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 10, width = 1, line = TRUE,
                   bar = FALSE, mirror = FALSE)

barprofileglyph1 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 10, width = 1, mirror = FALSE)

barglyph2 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 10, width = 3, line = FALSE, mirror = FALSE)

profileglyph2 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 10, width = 3, line = TRUE,
```

```
                  bar = FALSE, mirror = FALSE)

barprofileglyph2 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 10, width = 3, mirror = FALSE)

barglyph3 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 10, line = FALSE, mirror = FALSE)

profileglyph3 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 10, line = TRUE,
                   bar = FALSE, mirror = FALSE)

barprofileglyph3 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 10, mirror = FALSE)

grid.arrange(barglyph1, barglyph2, barglyph3, nrow = 1)
grid.arrange(profileglyph1, profileglyph2, profileglyph3, nrow = 1)
grid.arrange(barprofileglyph1, barprofileglyph2, barprofileglyph3, nrow = 1)

barglyph1 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 10, width = 1, line = FALSE, flip.axes = TRUE)

profileglyph1 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 10, width = 1, line = TRUE, bar = FALSE,
                   flip.axes = TRUE)

barprofileglyph1 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 10, width = 1, flip.axes = TRUE)

barglyph2 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 10, width = 3, line = FALSE, flip.axes = TRUE)

barprofileglyph2 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 10, width = 3, flip.axes = TRUE)
```

```
profileglyph2 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 10, width = 3, line = TRUE, bar = FALSE,
                   flip.axes = TRUE)

barglyph3 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 10, line = FALSE, flip.axes = TRUE)

profileglyph3 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 10, line = TRUE, bar = FALSE,
                   flip.axes = TRUE)

barprofileglyph3 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 10, flip.axes = TRUE)

grid.arrange(barglyph1, barglyph2, barglyph3, nrow = 1)
grid.arrange(profileglyph1, profileglyph2, profileglyph3, nrow = 1)
grid.arrange(barprofileglyph1, barprofileglyph2, barprofileglyph3, nrow = 1)

barglyph1 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 10, width = 1, line = FALSE,
                   mirror = FALSE, flip.axes = TRUE)

profileglyph1 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 10, width = 1, line = TRUE,
                   bar = FALSE, mirror = FALSE, flip.axes = TRUE)

barprofileglyph1 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 10, width = 1, mirror = FALSE, flip.axes = TRUE)

barglyph2 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 10, width = 3, line = FALSE,
                   mirror = FALSE, flip.axes = TRUE)

profileglyph2 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 10, width = 3, line = TRUE,
```

```
                  bar = FALSE, mirror = FALSE, flip.axes = TRUE)

barprofileglyph2 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                  z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                  size = 10, width = 3, mirror = FALSE, flip.axes = TRUE)

barglyph3 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                  z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                  size = 10, line = FALSE,
                  mirror = FALSE, flip.axes = TRUE)

profileglyph3 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                  z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                  size = 10, line = TRUE,
                  bar = FALSE, mirror = FALSE, flip.axes = TRUE)

barprofileglyph3 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                  z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                  size = 10, mirror = FALSE, flip.axes = TRUE)

grid.arrange(barglyph1, barglyph2, barglyph3, nrow = 1)
grid.arrange(profileglyph1, profileglyph2, profileglyph3, nrow = 1)
grid.arrange(barprofileglyph1, barprofileglyph2, barprofileglyph3, nrow = 1)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust bar and profile line width
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

barglyph1 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                  z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                  size = 10, line = FALSE)

profileglyph1 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                  z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                  size = 10, line = TRUE, bar = FALSE)

barprofileglyph1 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                  z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                  size = 10)

barglyph2 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                  z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                  size = 10, line = FALSE,
                  lwd.bar= 3)
```

```
profileglyph2 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 10, line = TRUE, bar = FALSE,
                   lwd.line = 3)

barprofileglyph2 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 10, lwd.bar= 3, lwd.line = 3)

barglyph3 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 10, line = FALSE,
                   lwd.bar= 5)

profileglyph3 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 10, line = TRUE, bar = FALSE,
                   lwd.line = 5)

barprofileglyph3 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 10, lwd.bar= 5, lwd.line = 5)

grid.arrange(barglyph1, barglyph2, barglyph3,
             profileglyph1, profileglyph2, profileglyph3,
             barprofileglyph1, barprofileglyph2, barprofileglyph3,
             nrow = 3, ncol = 3)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust bar and line colour
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

barglyph1 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 15, line = FALSE,
                   col.bar = "cyan")

profileglyph1 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 15, line = TRUE, bar = FALSE,
                   col.line = "green")

barprofileglyph1 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 15,
```

```
                    col.bar = "salmon", col.line = "salmon")

barglyph2 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 15, line = FALSE,
                   fill = "cyan")

profileglyph2 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 15, line = TRUE, bar = FALSE,
                   fill = "green")

barprofileglyph2 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 15,
                   fill = "salmon")

grid.arrange(barglyph1, barglyph2, nrow = 1)
grid.arrange(profileglyph1, profileglyph2, nrow = 1)
grid.arrange(barprofileglyph1, barprofileglyph2, nrow = 1)

barglyph1 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 15, line = FALSE,
                   mirror = FALSE, col.bar = "cyan")

profileglyph1 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 15, line = TRUE, bar = FALSE,
                   mirror = FALSE, col.line = "green")

barprofileglyph1 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 15,
                   mirror = FALSE, col.bar = "salmon", col.line = "salmon")

barglyph2 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 15, line = FALSE,
                   mirror = FALSE, fill = "cyan")

profileglyph2 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 15, line = TRUE, bar = FALSE,
                   mirror = FALSE, fill = "green")
```

```
barprofileglyph2 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 15,
                   mirror = FALSE, fill = "salmon")

grid.arrange(barglyph1, barglyph2, nrow = 1)
grid.arrange(profileglyph1, profileglyph2, nrow = 1)
grid.arrange(barprofileglyph1, barprofileglyph2, nrow = 1)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Multivariate bar fill colour
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

barglyph1 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 20, line = FALSE,
                   fill = RColorBrewer::brewer.pal(6, "Dark2"))

barprofileglyph1 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 20,
                   fill = RColorBrewer::brewer.pal(6, "Dark2"))

barglyph2 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 20, line = FALSE, mirror = FALSE,
                   fill = RColorBrewer::brewer.pal(6, "Dark2"))

barprofileglyph2 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 20, mirror = FALSE,
                   fill = RColorBrewer::brewer.pal(6, "Dark2"))

barglyph3 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 20, line = FALSE, flip.axes = TRUE,
                   fill = RColorBrewer::brewer.pal(6, "Dark2"))

barprofileglyph3 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 20, flip.axes = TRUE,
                   fill = RColorBrewer::brewer.pal(6, "Dark2"))

barglyph4 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
```

```
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                    size = 20, line = FALSE,
                    mirror = FALSE, flip.axes = TRUE,
                    fill = RColorBrewer::brewer.pal(6, "Dark2"))

barprofileglyph4 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                    size = 20, mirror = FALSE, flip.axes = TRUE,
                    fill = RColorBrewer::brewer.pal(6, "Dark2"))

grid.arrange(barglyph1, barprofileglyph1,
             barglyph2, barprofileglyph2,
             nrow = 2, ncol = 2)
grid.arrange(barglyph3, barprofileglyph3,
             barglyph4, barprofileglyph4,
             nrow = 2, ncol = 2)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust line join style
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

barglyph1 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                    size = 25, lwd.bar = 10, width = 5,
                    line = FALSE)

barglyph2 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                    size = 25, lwd.bar = 10, width = 5,
                    linejoin = "round", line = FALSE)

barglyph3 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                    size = 25, lwd.bar = 10, width = 5,
                    linejoin = "bevel", line = FALSE)

grid.arrange(barglyph1, barglyph2, barglyph3,
             nrow = 1, ncol = 3)

profileglyph1 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                    size = 25, lwd.line = 10, width = 5,
                    bar = FALSE)

profileglyph2 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                    size = 25, lwd.line = 10, width = 5,
```

```
                  linejoin = "round", bar = FALSE)

profileglyph3 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 25, lwd.line = 10, width = 5,
                   linejoin = "bevel", bar = FALSE)

grid.arrange(profileglyph1, profileglyph2, profileglyph3,
             nrow = 1, ncol = 3)

barprofileglyph1 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 25, lwd.bar = 10, width = 5)

barprofileglyph2 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 25, lwd.bar = 10, width = 5,
                   linejoin = "round")

barprofileglyph3 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 25, lwd.bar = 10, width = 5,
                   linejoin = "bevel")

grid.arrange(barprofileglyph1, barprofileglyph2, barprofileglyph3,
             nrow = 1, ncol = 3)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust line end style
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

barprofileglyph1 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 25, lwd.line = 10, width = 5)

barprofileglyph2 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 25, lwd.line = 10, width = 5,
                   lineend = "butt")

barprofileglyph3 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33),
                   size = 25, lwd.line = 10, width = 5,
                   lineend = "square")

grid.arrange(barprofileglyph1, barprofileglyph2, barprofileglyph3,
```

```
             nrow = 1, ncol = 3)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust grid levels
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

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

barprofileglyph1 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(1, 3, 2, 1, 2, 3),
                   size = 10, width = 5,
                   draw.grid = TRUE, lwd.bar = 5,
                   grid.levels = gl, col.grid = "black")

barprofileglyph2 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(1, 3, 2, 1, 2, 3),
                   size = 10, width = 5, lwd.bar = 5,
                   draw.grid = TRUE, mirror = FALSE,
                   grid.levels = gl, col.grid = "black")

barprofileglyph3 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(1, 3, 2, 1, 2, 3),
                   size = 10, width = 5, flip.axes = TRUE,
                   draw.grid = TRUE, lwd.bar = 5,
                   grid.levels = gl, col.grid = "black")

barprofileglyph4 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(1, 3, 2, 1, 2, 3),
                   size = 10, width = 5, flip.axes = TRUE,
                   draw.grid = TRUE, lwd.bar = 5, mirror = FALSE,
                   grid.levels = gl, col.grid = "black")

grid.arrange(barprofileglyph1, barprofileglyph2, nrow = 1)
grid.arrange(barprofileglyph3, barprofileglyph4, nrow = 1)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust grid level colours
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

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

barprofileglyph1 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                   z = c(1, 3, 2, 1, 2, 3),
                   size = 10, width = 5,
```

```
                    draw.grid = TRUE, lwd.bar = 5,
                    grid.levels = gl, line = FALSE,
                    col.grid = "white", col.bar = "white",
                    fill = RColorBrewer::brewer.pal(6, "Dark2"))

barprofileglyph2 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(1, 3, 2, 1, 2, 3),
                    size = 10, width = 5, lwd.bar = 5,
                    draw.grid = TRUE, mirror = FALSE,
                    grid.levels = gl, line = FALSE,
                    col.grid = "white", col.bar = "white",
                    fill = RColorBrewer::brewer.pal(6, "Dark2"))

barprofileglyph3 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(1, 3, 2, 1, 2, 3),
                    size = 10, width = 5, flip.axes = TRUE,
                    draw.grid = TRUE, lwd.bar = 5,
                    grid.levels = gl, line = FALSE,
                    col.grid = "white", col.bar = "white",
                    fill = RColorBrewer::brewer.pal(6, "Dark2"))

barprofileglyph4 <-
  profileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(1, 3, 2, 1, 2, 3),
                    size = 10, width = 5, flip.axes = TRUE,
                    draw.grid = TRUE, lwd.bar = 5, mirror = FALSE,
                    grid.levels = gl, line = FALSE,
                    col.grid = "white", col.bar = "white",
                    fill = RColorBrewer::brewer.pal(6, "Dark2"))

grid.arrange(barprofileglyph1, barprofileglyph2, nrow = 1)
grid.arrange(barprofileglyph3, barprofileglyph4, nrow = 1)
```

`scale_z_continuous` *Alter scales for continuous data mapped to glyphs*

Description

Scale variable(s) mapped to the glyphs to the given range using `rescale_pal`. Each variable in `z` gets its own independent continuous scale.

Usage

```
scale_z_continuous(..., range = c(0.1, 1), z)
```

Arguments

...	Additional arguments to be passed on to continuous_scale.
range	The range to which the variable(s) specified in argument z are to be scaled.
z	The variable(s) mapped to the glyph as a character vector.

Value

A continuous_scale object that can be added to a ggplot object. The returned scale applies to all aesthetics named in z simultaneously (i.e. the variables specified via cols in the corresponding geom_*glyph() layer), rescaling their combined, pooled range to range for use in determining glyph dimensions.

---

scale_z_discrete *Alter scales for discrete data mapped to glyphs*

Description

Scale variable(s) mapped to the glyphs as discrete/ordered factors. Each variable in z gets its own independent discrete scale.

Usage

```
scale_z_discrete(..., z, na.translate = TRUE, na.value = NA, guide = "legend")
```

Arguments

...	Additional arguments to be passed on to discrete_scale.
z	The variable(s) mapped to the glyph as a character vector or single variable name.
na.translate	Logical. If TRUE (default), missing values are translated to "NA".
na.value	The aesthetic value to use for missing values.
guide	A function used to create a guide or its name. See guidesfor more information.

Value

If z is of length one, a single discrete_scale object built on ScaleDiscreteZIdentity. If z has length greater than one, a list of such discrete_scale objects, one per variable named in z, which can be added to a ggplot object (via +) to attach independent identity scales to each variable. Each scale passes the ordinal rank of the ordered factor levels through unchanged (rather than mapping them to a palette), so that draw.grid = TRUE in the corresponding geom_*glyph() layer can use the values directly for grid-level placement, while still providing a legend showing the factor levels for each variable.

scale_z_fill_continuous *Alter scales for continuous data mapped to glyphs fill colour*

Description

Scale variable(s) mapped to the glyph fill colours.

Usage

```
scale_z_fill_continuous(..., palette, z, guide = c("legend"))
```

Arguments

...	Additional arguments to be passed on to continuous_scale.
palette	One of the following: • NULL for the default palette stored in the theme. • a character vector of colours. • a single string naming a palette. • a palette function that when called with a numeric vector with values between 0 and 1 returns the corresponding output values.
z	The variable(s) mapped to the glyph as a character vector.
guide	A function used to create a guide or its name. See guidesfor more information.

Value

A continuous colour scale object (one of scale_color_distiller, scale_color_viridis_c, or scale_color_continuous, depending on palette) that can be added to a ggplot object. The returned scale applies to all aesthetics named in z simultaneously, mapping their combined, pooled range to a continuous colour gradient for use as the fill/colour of glyph segments in the corresponding geom_*glyph() layer.

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,
  grid.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`.

`grid.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.

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

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_starglyph`

Other grobs: `bubbleglyphGrob()`, `dotglyphGrob()`, `flowerglyphGrob()`, `metroglyphGrob()`, `pieglyphGrob()`, `profileglyphGrob()`, `tileglyphGrob()`

Examples

```
library(ggmultiglyph)
library(grid)
library(gridExtra)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Whisker and contour combination
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

sg1 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15)

sg2 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                     whisker = FALSE)

sg3 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                     contour = FALSE)

sg4 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
```

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

sg5 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                    whisker = FALSE,
                    angle.start = 0, angle.stop = base::pi)

sg6 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                    contour = FALSE,
                    angle.start = 0, angle.stop = base::pi)

grid.arrange(sg1, sg2, sg3, sg4, sg5, sg6, nrow = 2, ncol = 3)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust size
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

sg1 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 10)

sg2 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15)

sg3 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 20)

sg4 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 10,
                    angle.start = 0, angle.stop = base::pi)

sg5 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                    angle.start = 0, angle.stop = base::pi)

sg6 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 20,
                    angle.start = 0, angle.stop = base::pi)

grid.arrange(sg1, sg2, sg3, sg4, sg5, sg6, nrow = 2, ncol = 3)

sg1 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 10,
                    whisker = FALSE)

sg2 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                    whisker = FALSE)

sg3 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 20,
                    whisker = FALSE)
```

```
sg4 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 10,
                     whisker = FALSE,
                     angle.start = 0, angle.stop = base::pi)

sg5 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                     whisker = FALSE,
                     angle.start = 0, angle.stop = base::pi)

sg6 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 20,
                     whisker = FALSE,
                     angle.start = 0, angle.stop = base::pi)

grid.arrange(sg1, sg2, sg3, sg4, sg5, sg6, nrow = 2, ncol = 3)



sg1 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 10,
                     contour = FALSE)

sg2 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                     contour = FALSE)

sg3 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 20,
                     contour = FALSE)

sg4 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 10,
                     contour = FALSE,
                     angle.start = 0, angle.stop = base::pi)

sg5 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                     contour = FALSE,
                     angle.start = 0, angle.stop = base::pi)

sg6 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 20,
                     contour = FALSE,
                     angle.start = 0, angle.stop = base::pi)

grid.arrange(sg1, sg2, sg3, sg4, sg5, sg6, nrow = 2, ncol = 3)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust angle
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

sg1 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
```

```
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15)

sg2 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                    angle.start = 0, angle.stop = base::pi)

sg3 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                    angle.start = 90 * (base::pi/180),
                    angle.stop = 270 * (base::pi/180))

sg4 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                    angle.start = 2 * base::pi,
                    angle.stop = 0)

sg5 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                    angle.start = base::pi, angle.stop = 0)

sg6 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                    angle.start = 270 * (base::pi/180),
                    angle.stop = 90 * (base::pi/180))

grid.arrange(sg1, sg2, sg3, sg4, sg5, sg6, nrow = 2, ncol = 3)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust whisker and contour line width
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

sg1 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15)

sg2 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                    lwd.whisker = 3, lwd.contour = 0.1)

sg3 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                    lwd.whisker = 0.1, lwd.contour = 3)

sg4 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                    angle.start = 0, angle.stop = base::pi)

sg5 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                    lwd.whisker = 3, lwd.contour = 0.1,
                    angle.start = 0, angle.stop = base::pi)

sg6 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
```

```
                    lwd.whisker = 0.1, lwd.contour = 3,
                    angle.start = 0, angle.stop = base::pi)

grid.arrange(sg1, sg2, sg3, sg4, sg5, sg6, nrow = 2, ncol = 3)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust fill colour
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

sg1 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                    fill = "salmon")

sg2 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                    lwd.whisker = 3, lwd.contour = 0.1,
                    fill = "cyan")

sg3 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                    lwd.whisker = 0.1, lwd.contour = 3,
                    fill = "green")

sg4 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                    angle.start = 0, angle.stop = base::pi,
                    fill = "salmon")

sg5 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                    lwd.whisker = 3, lwd.contour = 0.1,
                    angle.start = 0, angle.stop = base::pi,
                    fill = "cyan")

sg6 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                    lwd.whisker = 0.1, lwd.contour = 3,
                    angle.start = 0, angle.stop = base::pi,
                    fill = "green")

grid.arrange(sg1, sg2, sg3, sg4, sg5, sg6, nrow = 2, ncol = 3)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust whisker and contour colour
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

sg1 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                    col.whisker = "salmon", col.contour = "salmon")

sg2 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                    lwd.whisker = 3, lwd.contour = 0.1,
```

```
                    col.whisker = "cyan", col.contour = "gray")

sg3 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                    lwd.whisker = 0.1, lwd.contour = 3,
                    col.whisker = "gray", col.contour = "green")

sg4 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                    angle.start = 0, angle.stop = base::pi,
                    col.whisker = "salmon", col.contour = "salmon")

sg5 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                    lwd.whisker = 3, lwd.contour = 0.1,
                    angle.start = 0, angle.stop = base::pi,
                    col.whisker = "cyan", col.contour = "gray")

sg6 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                    lwd.whisker = 0.1, lwd.contour = 3,
                    angle.start = 0, angle.stop = base::pi,
                    col.whisker = "gray", col.contour = "green")

grid.arrange(sg1, sg2, sg3, sg4, sg5, sg6, nrow = 2, ncol = 3)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Multivariate whisker colour
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

sg1 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                    col.whisker = RColorBrewer::brewer.pal(6, "Dark2"),
                    col.contour = "gray")

sg2 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                    lwd.whisker = 3, lwd.contour = 0.1,
                    col.whisker = RColorBrewer::brewer.pal(6, "Dark2"),
                    col.contour = "gray")

sg3 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                    lwd.whisker = 0.1, lwd.contour = 3,
                    col.whisker = RColorBrewer::brewer.pal(6, "Dark2"),
                    col.contour = "gray")

sg4 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                    angle.start = 0, angle.stop = base::pi,
                    col.whisker = RColorBrewer::brewer.pal(6, "Dark2"),
                    col.contour = "gray")
```

```
sg5 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                     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 = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(0.24, 0.3, 0.8, 1.4, 0.6, 0.33), size = 15,
                     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.arrange(sg1, sg2, sg3, sg4, sg5, sg6, nrow = 2, ncol = 3)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust line join style
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

sg1 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(0.28, 0.33, 0.8, 1.2, 0.6, 0.5, 0.7), size = 15,
                     lwd.contour = 10)

sg2 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(0.28, 0.33, 0.8, 1.2, 0.6, 0.5, 0.7), size = 15,
                     lwd.contour = 10, linejoin = "bevel")

sg3 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(0.28, 0.33, 0.8, 1.2, 0.6, 0.5, 0.7), size = 15,
                     lwd.contour = 10, linejoin = "round")

grid.arrange(sg1, sg2, sg3, nrow = 1, ncol = 3)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust line end style
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

sg1 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(0.28, 0.33, 0.8, 1.2, 0.6, 0.5, 0.7), size = 15,
                     lwd.whisker = 10, contour = FALSE)

sg2 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(0.28, 0.33, 0.8, 1.2, 0.6, 0.5, 0.7), size = 15,
                     lwd.whisker = 10, lineend = "butt", contour = FALSE)

sg3 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(0.28, 0.33, 0.8, 1.2, 0.6, 0.5, 0.7), size = 15,
                     lwd.whisker = 10, lineend = "square", contour = FALSE)

grid.arrange(sg1, sg2, sg3, nrow = 1, ncol = 3)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
# Adjust grid levels
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Grid levels
gl <- split(x = c(rep(c(1, 2, 3), 4),
                  rep(c(1, 2, 3, 4), 2)),
            f = c(rep(1:4, each = 3),
                  rep(5:6, each = 4)))
gl

sg1 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(1, 3, 2, 1, 2, 3), size = 5,
                     draw.grid = TRUE, grid.levels = gl,
                     grid.point.size = 0.01)

sg2 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(1, 3, 2, 1, 2, 3), size = 5,
                     lwd.whisker = 3, col.points = "white",
                     draw.grid = TRUE, grid.levels = gl,
                     grid.point.size = 0.05,
                     contour = FALSE)

sg3 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(1, 3, 2, 1, 2, 3), size = 5,
                     lwd.contour = 3,
                     draw.grid = FALSE, grid.levels = gl,
                     grid.point.size = 0.05,
                     whisker = FALSE)

sg4 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(1, 3, 2, 1, 2, 3), size = 5,
                     angle.start = 0, angle.stop = base::pi,
                     draw.grid = TRUE, grid.levels = gl,
                     grid.point.size = 0.01)

sg5 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     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,
                     grid.point.size = 0.05, contour = FALSE)

sg6 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     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.arrange(sg1, sg2, sg3, sg4, sg5, sg6, nrow = 2, ncol = 3)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust fill, whisker, contour and grid level colours
```

```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

sg1 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     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",
                     grid.point.size = 0.05)

sg2 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(1, 3, 2, 1, 2, 3), size = 5,
                     contour = FALSE,
                     lwd.whisker = 3,
                     col.whisker = RColorBrewer::brewer.pal(6, "Dark2"),
                     draw.grid = TRUE, grid.levels = gl,
                     grid.point.size = 0.05)

sg3 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(1, 3, 2, 1, 2, 3), size = 5,
                     whisker = FALSE,
                     lwd.contour = 3,
                     draw.grid = FALSE, grid.levels = gl,
                     col.whisker = RColorBrewer::brewer.pal(6, "Dark2"),
                     grid.point.size = 0.05)

sg4 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(1, 3, 2, 1, 2, 3), size = 5,
                     angle.start = 0, angle.stop = base::pi,
                     draw.grid = TRUE, grid.levels = gl,
                     grid.point.size = 0.01)

sg5 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(1, 3, 2, 1, 2, 3), size = 5,
                     col.contour = "gray",
                     col.whisker = RColorBrewer::brewer.pal(6, "Dark2"),
                     angle.start = 0, angle.stop = base::pi,
                     draw.grid = TRUE, grid.levels = gl,
                     grid.point.size = 0.01, col.points = "gray")

sg6 <- starglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(1, 3, 2, 1, 2, 3), size = 5,
                     lwd.contour = 3,
                     whisker = FALSE,
                     angle.start = 0, angle.stop = base::pi,
                     draw.grid = FALSE, grid.levels = gl,
                     col.whisker = RColorBrewer::brewer.pal(6, "Dark2"),
                     grid.point.size = 0.05)

grid.arrange(sg1, sg2, sg3, sg4, sg5, sg6, nrow = 2, ncol = 3)
```

`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: bubbleglyphGrob(), dotglyphGrob(), flowerglyphGrob(), metroglyphGrob(), pieglyphGrob(), profileglyphGrob(), starglyphGrob()

Examples

```
library(ggmultiglyph)
library(grid)
library(gridExtra)


#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust tile size
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

tg1 <- tileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7, 4.3),
                     size = 3)

tg2 <- tileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7, 4.3),
                     size = 5)

tg3 <- tileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7, 4.3),
                     size = 8)

tg4 <- tileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7, 4.3),
                     size = 3, nrow = 2)

tg5 <- tileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7, 4.3),
                     size = 5, nrow = 2)

tg6 <- tileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7, 4.3),
                     size = 8, nrow = 2)

grid.arrange(tg1, tg2, tg3, tg4, tg5, tg6,
```

```
             nrow = 2, ncol = 3)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust line width
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

tg1 <- tileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7, 4.3),
                     size = 5)

tg2 <- tileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7, 4.3),
                     size = 5, lwd = 3)

tg3 <- tileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7, 4.3),
                     size = 5, lwd = 5)

tg4 <- tileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7, 4.3),
                     size = 5, nrow = 2)

tg5 <- tileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7, 4.3),
                     size = 5, nrow = 2, lwd = 3)

tg6 <- tileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7, 4.3),
                     size = 5, nrow = 2, lwd = 5)

grid.arrange(tg1, tg2, tg3, tg4, tg5, tg6,
             nrow = 2, ncol = 3)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust line join style
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

tg1 <- tileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(4, 3.5, 2.7, 6.8, 3.4),
                     size = 5, lwd = 10)

tg2 <- tileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(4, 3.5, 2.7, 6.8, 3.4),
                     size = 5, lwd = 10, linejoin = "bevel")

tg3 <- tileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(4, 3.5, 2.7, 6.8, 3.4),
                     size = 5, lwd = 10, linejoin = "round")

tg4 <- tileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(4, 3.5, 2.7, 6.8, 3.4),
                     size = 5, nrow = 2, lwd = 10)
```

```
tg5 <- tileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(4, 3.5, 2.7, 6.8, 3.4),
                     size = 5, nrow = 2, lwd = 10,
                     linejoin = "bevel")

tg6 <- tileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(4, 3.5, 2.7, 6.8, 3.4),
                     size = 5, nrow = 2, lwd = 10,
                     linejoin = "round")

grid.arrange(tg1, tg2, tg3, tg4, tg5, tg6,
             nrow = 2, ncol = 3)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust rows
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

tg1 <- tileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7, 4.3),
                     size = 5)

tg2 <- tileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                     size = 5)

tg3 <- tileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7, 4.3),
                     size = 5, nrow = 2)

tg4 <- tileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                     size = 5, nrow = 2)

grid.arrange(tg1, tg2, tg3, tg4,
             nrow = 2, ncol = 2)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Adjust tile ratio
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

tg1 <- tileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7, 4.3),
                     size = 5)

tg2 <- tileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7, 4.3),
                     size = 5, ratio = 0.5)

tg3 <- tileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                     z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7, 4.3),
                     size = 5, ratio = 3)

tg4 <- tileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
```

```
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7, 4.3),
                    size = 5, nrow = 2)

tg5 <- tileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7, 4.3),
                    size = 5, nrow = 2, ratio = 0.5)

tg6 <- tileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7, 4.3),
                    size = 5, nrow = 2, ratio = 3)

grid.arrange(tg1, tg2, tg3, tg4, tg5, tg6,
             nrow = 2, ncol = 3)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Multivariate tile fill
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

tg1 <- tileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7, 4.3),
                    size = 5,
                    fill = RColorBrewer::brewer.pal(7, "Dark2"))

tg2 <- tileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    size = 5,
                    fill = RColorBrewer::brewer.pal(6, "Dark2"))

tg3 <- tileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    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"))

tg4 <- tileglyphGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
                    z = c(4, 3.5, 2.7, 6.8, 3.4, 5.7),
                    size = 5, nrow = 2,
                    fill = RColorBrewer::brewer.pal(6, "Dark2"))

grid.arrange(tg1, tg2, tg3, tg4,
             nrow = 2, ncol = 2)
```

Index