Package 'donutsk'

July 22, 2025

Title Construct Advanced Donut Charts

Version 0.1.1

Description Build donut/pie charts with 'ggplot2' layer by layer, exploiting the advantages of polar symmetry. Leverage layouts to distribute labels effectively. Connect labels to donut segments using pins. Streamline annotation and highlighting.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.2.3

URL https://github.com/dkibalnikov/donutsk, https://dkibalnikov.github.io/donutsk/

BugReports https://github.com/dkibalnikov/donutsk/issues

Depends ggplot2 (>= 3.5.0), R (>= 4.2.0)

Imports dplyr (>= 1.1.2), glue (>= 1.6.2), rlang (>= 1.1.1)

Suggests knitr, rmarkdown, scales (>= 1.3.0), stringr (>= 1.5.0), testthat (>= 3.0.0), tidyr (>= 1.3.0)

Config/testthat/edition 3

LazyData true

VignetteBuilder knitr

NeedsCompilation no

Author Dmitry Kibalnikov [aut, cre, cph]

Maintainer Dmitry Kibalnikov <d.kibalnikov@gmail.com>

Repository CRAN

Date/Publication 2024-04-22 18:50:06 UTC

Contents

`donut_geom` *Create pie or donut chart*

Description

Create pie or donut charts while retaining ggplot flexibility, such as leveraging faceting and palettes, and fine-tuning appearance

- The function `geom_donut_int()` creates visually **internal** donut layer as aggregation of passed values
- The function `geom_donut_ext()` creates visually **external** donut layer of passed values
- `geom_donut_int0()` and `geom_donut_ext()` are generic geoms not supporting highlight feature

Usage

```
geom_donut_int0(
  mapping = NULL,
  data = NULL,
  stat = "donut_int",
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  r_int = 0,
  r_ext = 1,
  hl_shift = 0.1,
  ...
)

geom_donut_int(..., hl_col = "firebrick")

geom_donut_ext0(
  mapping = NULL,
  data = NULL,
  stat = "donut_ext",
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  r_int = 1.5,
  r_ext = 2,
```

```
  hl_shift = 0.1,
  ...
)

geom_donut_ext(..., hl_col = "firebrick")
```

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, either as a ggproto Geom subclass or as a string naming the stat stripped of the stat_ prefix (e.g. "count" rather than "stat_count")
position	Position adjustment, either as a string naming the adjustment (e.g. "jitter" to use position_jitter), or the result of a call to a position adjustment function. Use the latter if you need to change the settings of the adjustment.
na.rm	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders().
r_int	Internal donut radius
r_ext	External pie or donut radius
hl_shift	Sets the spacing to show highlighted segments
...	Other arguments passed on to layer(). These are often aesthetics, used to set an aesthetic to a fixed value, like colour = "red" or size = 3. They may also be parameters to the paired geom/stat.
hl_col	Sets the color for highlighted segments. It's possible to use both simultaneously hl_col and generic colour

Format

An object of class `StatDonutInt` (inherits from `Stat`, `ggproto`, gg) of length 4.

An object of class `StatDonutIntHl` (inherits from `Stat`, `ggproto`, gg) of length 4.

An object of class `StatDonutExt` (inherits from `Stat`, `ggproto`, gg) of length 4.

An object of class `StatDonutExtHl` (inherits from `Stat`, `ggproto`, gg) of length 4.

Details

There are two additional aesthetics possible to use:

- `highlight` - optional aesthetic which expects logical (TRUE/FALSE) variable in order to highlight particular donut segments
- `opacity` - operates pretty much the same as `alpha` but ensure more contrast colors and removes legend. Once `alpha` is set `opacity` does not affect a chart

Value

None

Examples

```
# Create an example
set.seed(1605)
n <- 20
df <- dplyr::tibble(
  lvl1 = sample(LETTERS[1:5], n, TRUE),
  lvl2 = sample(LETTERS[6:24], n, TRUE),
  value = sample(1:20, n, TRUE),
  highlight_ext = sample(c(FALSE,TRUE), n, TRUE, c(.7, .3))) |>
  dplyr::mutate(highlight_int = ifelse(lvl1 == "A", TRUE, FALSE))

# Create a simple pie chart
ggplot(df, aes(value = value, fill=lvl1)) +
  geom_donut_int(alpha=.6) +
  coord_polar(theta = "y")

# Create a simple donut chart that can handle more granular data
# and highlight specific segments
ggplot(df, aes(value = value, fill=lvl2, highlight=highlight_ext)) +
  geom_donut_int(r_int=.5, alpha=.6, linewidth=.2) +
  coord_polar(theta = "y") +
  xlim(0, 1.5)

# Perform data preparation tasks with `packing()`
# and apply specific color
packing(df, value) |>
  ggplot(aes(value = value, fill=lvl2, highlight=highlight_ext)) +
  geom_donut_int(r_int=.5, alpha=.6, linewidth=.2, col = "gray20") +
  coord_polar(theta = "y") +
  xlim(0, 1.5)
```

```
# Built combined donut chart with interanl and external layers
dplyr::bind_rows(
# arrange by value
`arrange()` = dplyr::arrange(df, lvl1, lvl2, value),
# pack values for better space management
`packing()` = packing(df, value, lvl1),
.id = "prep_type") |>
 ggplot(aes(value = value, fill=lvl1)) +
 geom_donut_int(aes(highlight=highlight_int), alpha=.6) +
 geom_donut_ext(aes(opacity=lvl2, highlight=highlight_int)) +
 # apply facets
 facet_grid(~prep_type) +
 # style chart with palette and theme
 scale_fill_viridis_d(option = "inferno", begin = .1, end = .7) +
 theme_void() +
 coord_polar(theta = "y") +
 xlim(0, 2.5)
```

---

donut_label *Create pie or donut label and text annotations*

Description

The set of annotation functions utilizes layout functions to effectively distribute labels within the available space Annotations are streamlined by leveraging pre-calculated special variables such as `.sum`, `.mean`, and `.n` (see Details).

- The function `geom_label_int()` creates `geom_label`-like **internal** donut layer as aggregation of passed values
- The function `geom_text_int()` creates `geom_text`-like **internal** donut layer as aggregation of passed values
- The function `geom_label_ext()` creates `geom_label`-like **external** donut layer of passed values
- The function `geom_text_ext()` creates `geom_text`-like **external** donut layer of passed values

Usage

```
geom_label_int(
  mapping = NULL,
  data = NULL,
  stat = "label_int",
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  r = 1,
```

```
  ...
)

geom_text_int(
  mapping = NULL,
  data = NULL,
  stat = "text_int",
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  r = 1,
  ...
)

geom_label_ext(
  mapping = NULL,
  data = NULL,
  stat = "label_ext",
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  layout = circle(),
  ...
)

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

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, either as a `ggproto` `Geom` subclass or as a string naming the stat stripped of the `stat_` prefix (e.g. `"count"` rather than `"stat_count"`)
`position`	Position adjustment, either as a string, or the result of a call to a position adjustment function. Cannot be jointly specified with `nudge_x` or `nudge_y`.
`na.rm`	If `FALSE`, the default, missing values are removed with a warning. If `TRUE`, missing values are silently removed.
`show.legend`	logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display.
`inherit.aes`	If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. `borders()`.
`r`	Sets the radius to place label or text for internal donut
`...`	Other arguments passed on to `layer()`. These are often aesthetics, used to set an aesthetic to a fixed value, like `colour = "red"` or `size = 3`. They may also be parameters to the paired geom/stat.
`layout`	The layout function to effectively display text and labels

Format

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

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

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

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

Details

The label functions supports `glue::glue()` for convenient label construction like `Total: {.sum}`, where `.sum` is pre-calculated variable. You can still use `glue::glue()` or `paste()` functions to pass data.frame fields for label construction.

In addition to generic aesthetics like `color`, `fill`, `alpha`, etc., the following list of pre-calculated variables is available for `geom_label_int()` and `geom_text_int()`:

- `.sum`: Summation of the value field
- `.mean`: Mean of the value field
- `.median`: Median of the value field
- `.n`: Observation count of the value field
- `.prc`: Percentage of the value field

For `geom_label_ext()` and `geom_text_ext()`, which are suitable for external donut labels, the following list of pre-calculated variables is available:

- `.prc`: Percentage of the value field for the entire multiplicity
- `.prc_grp`: Percentage of the value field for the group defined by `fill`

Value

None

See Also

layouts, pins

Examples

```
# Create an example data set
n <- 30
set.seed(2021)
df <- dplyr::tibble(
  lvl1 = sample(LETTERS[1:5], n, TRUE),
  lvl2 = sample(LETTERS[6:24], n, TRUE),
  value = sample(1:20, n, TRUE),
  highlight_ext = sample(c(FALSE,TRUE), n, TRUE, c(.9, .1))) |>
 dplyr::mutate(highlight_int = dplyr::if_else(lvl1 == "A", TRUE, FALSE))

# Starting plot with doubled donuts and annotations for internal one
p <- dplyr::group_by(df, lvl1, lvl2, highlight_ext, highlight_int) |>
 dplyr::summarise(value = sum(value), .groups = "drop") |>
 packing(value, lvl1) |>
 ggplot(aes(value = value, fill = lvl1)) +
 geom_donut_int(aes(highlight = highlight_int), alpha=.5, r_int=.25) +
 geom_text_int(lineheight = .8, r=1.2, show.legend = FALSE,
  aes(label = "Sum {fill}:\n{.sum}-{scales::percent(.prc)}", col=lvl1)) +
 geom_donut_ext(aes(opacity = lvl2, highlight = highlight_ext)) +
 scale_fill_viridis_d(option = "inferno", begin = .1, end = .7) +
 scale_color_viridis_d(option = "inferno", begin = .1, end = .7) +
 guides(alpha=guide_legend(ncol = 2), fill=guide_legend(ncol = 2)) +
 theme_void() +
 theme(legend.position = "inside", legend.position.inside = c(0.1, 0.9))

p + coord_radial(theta = "y", expand = FALSE, rotate_angle = FALSE)

# Add labels to external donut as percent inside group
p + coord_radial(theta = "y", expand = FALSE, rotate_angle = FALSE) +
 geom_label_ext(aes(label=paste0(lvl2, ": {scales::percent(.prc_grp)}")),
                show.legend = FALSE, size=3, col="white")

# Leverage ggplot2 feature for labels
p + coord_radial(theta = "y", expand = FALSE, rotate_angle = TRUE) +
 geom_label_ext(aes(label=paste0(lvl2, ": {scales::percent(.prc)}")),
                show.legend = FALSE, size=3, col="white", angle=90,
```

```
                layout = circle())

# Leverage another layout
p + coord_radial(theta = "y", expand = FALSE, rotate_angle = FALSE) +
 geom_label_ext(aes(label=paste0(lvl2, ": {scales::percent(.prc_grp)}")),
                show.legend = FALSE, size=3, col="white",
                layout = tv(thinner = TRUE, thinner_gap = 0.15))
```

GDP *The World Bank GDP, PPP (current international $)*

Description

- A pre-processed subset of GDP data from the World Bank
- GDP, PPP means gross domestic product based on purchasing power parity
- *current international $* means actual (not adjusted to inflation) US dollars

Usage

```
GDP
```

Format

GDP**:**

A data frame with 6,004 rows and 5 columns:

date Year

country Country name

region Region hierarchy

region_ISO 3 letter ISO region codes

GDP GDP, PPP (current international $) ...

Source

https://data.worldbank.org/indicator/NY.GDP.MKTP.PP.CD

`layouts` *The set of layout functions is designed to effectively display text and labels*

Description

The layout functions help to streamline displaying text and labels *geoms* without overlapping effectively leveraging space available for pie and donut charts

- `tv()` - The function builds layout resembled an old-fashioned TV screen
- `petal()` - The function builds layout resembled flower with petals
- `circle()` - The function builds circle layout
- `eye()` - The function builds two-sided layout

Usage

```
tv(
  scale_x = 1.5,
  scale_y = 1.5,
  bend_x = 1,
  bend_y = 1,
  thinner = FALSE,
  thinner_gap = 0.1
)

petal(
  rotate = 0,
  n = 4,
  scale = 2.5,
  bend = 0.3,
  thinner = FALSE,
  thinner_gap = 0.1
)

circle(r = 2.5, thinner = FALSE, thinner_gap = 0.1)

eye(scale_x = 2, bend_x = 1, alpha = 90, clove = 0.5)
```

Arguments

`scale_x`	Scales the layout in horizontal perspective
`scale_y`	Scales the layout in vertical perspective
`bend_x`	Adjusts the bend level in horizontal perspective
`bend_y`	Adjusts the bend level in vertical perspective
`thinner`	Distributes text or label elements across two different levels

thinner_gap	Sets the spacing between thinner levels
rotate	Rotates the layout clockwise
n	Sets the number of petals in the layout
scale	Scales the layout
bend	Manages the bending level
r	Sets the radius of the layout circle
alpha	Defines the angle of distribution in horizontal perspective. Pick up value from degree interval (0, 180)
clove	Determines the distribution proportion between the left and right-hand parts. Default value is 0.5. There should be numeric value from interval (0, 1) e.g. 0.4 denotes 40% cases on the right hand and 60% cases on the left hand

Value

Layout functions return layout function i.e. a function that takes a vector of angles and returns a numeric radius vector giving a position for each input value.

Layout functions are designed to be used with the layout argument of donutsk functions.

See Also

Utilized in the following functions: geom_label_ext, geom_text_ext, geom_pin

Examples

```
# Render multiple layouts simultaneously
list(petal_2n = petal(n = 2),
     petal_3n = petal(n = 3, rotate = 180),
     petal_4n = petal(n = 4),
     tv_base = tv(),
     tv_ext = tv(bend_x = 0, bend_y = 0, thinner = TRUE)) |>
  lapply(function(x){
    rlang::exec(x, 1:300/300) |>
      dplyr::tibble(r = _) |>
      dplyr::mutate(theta = 1:300/300)
  }) |>
  dplyr::bind_rows(.id = "layouts") |>
  ggplot(aes(x=r, y=theta, col = layouts)) +
  geom_point(alpha = .3) +
  coord_polar(theta = "y") +
  xlim(0, 3.5)

# The eye() layout generates table as an output
n <- 20
theta <- 1:n/n

dplyr::tibble(
  theta = theta,
  lbl = paste0("sample: ", sample(LETTERS, n, TRUE))
  ) |>
```

```
dplyr::bind_cols(lt = eye()(theta)) |>
ggplot(aes(x=x, y=y)) +
geom_point(aes(x=1, y=theta)) +
geom_point() +
geom_segment(aes(x=1, xend=x, y=theta, yend=y), linewidth=.2) +
geom_label(aes(label=lbl, hjust=dplyr::if_else(theta > 0.5, 1, 0)),
 nudge_x =.2) +
coord_polar(theta = "y") +
xlim(0, 5) +
ylim(0, 1)
```

---

packing *Arrange data to distribute small values*

Description

Arrange data to distribute small values further apart from each other

Usage

```
packing(.data, value, level = NULL)
```

Arguments

.data	A data frame, data frame extension (e.g. a tibble), or a lazy data frame (e.g. from dbplyr or dtplyr).
value	A .data field which contains values to distribute
level	A .data grouping field for distribution

Value

An object of the same type as .data.

Examples

```
# Create an example
n <- 20
df <- dplyr::tibble(
 lvl1 = sample(LETTERS[1:5], n, TRUE),
 lvl2 = sample(LETTERS[6:24], n, TRUE),
 value = sample(1:20, n, TRUE)
 )

# Arrange all values
packing(df, value)

# Arrange values within values
packing(df, value, lvl1)
```

pins *Connecting labels with donut segments*

Description

The set of functions served to connect text or labels with donut segments

- `geom_pin_line()` - builds curved line to linl label with donut segment
- `geom_pin_head()` - builds stylish point heads for pins
- `geom_pin()` - handy wrapper for `geom_pin_line()` and `geom_pin_head()`

Usage

```
geom_pin_line(
  mapping = NULL,
  data = NULL,
  stat = "pin",
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  r = 1.5,
  cut = 0.1,
  layout = circle(),
  ...
)

geom_pin_head(
  mapping = NULL,
  data = NULL,
  stat = "point",
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  r = 1.5,
  cut = 0.1,
  layout = circle(),
  ...
)

geom_pin(..., head = 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, either as a ggproto Geom subclass or as a string naming the stat stripped of the stat_ prefix (e.g. "count" rather than "stat_count")
position	Position adjustment, either as a string naming the adjustment (e.g. "jitter" to use position_jitter), or the result of a call to a position adjustment function. Use the latter if you need to change the settings of the adjustment.
na.rm	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders().
r	The radius where donut is placed
cut	Sets additional two-sided gap for pins
layout	The layout function to effectively display text and labels. Obviously it's better to have the same as for geom_label_ext or geom_text_ext
...	Parameters to be passed to geom_pin_line() and geom_pin_head()
head	Boolean - defines whether to add pin head

Format

An object of class StatPinLine (inherits from Stat, ggproto, gg) of length 3.

An object of class StatPinHead (inherits from Stat, ggproto, gg) of length 3.

Value

None

See Also

layouts, donut_label

Examples

```
n <- 30
set.seed(2021)
df <- dplyr::tibble(
  lvl1 = sample(LETTERS[1:5], n, TRUE),
  lvl2 = sample(LETTERS[6:24], n, TRUE),
  value = sample(1:20, n, TRUE),
  highlight_ext = sample(c(FALSE,TRUE), n, TRUE, c(.9, .1))) |>
  dplyr::mutate(highlight_int = dplyr::if_else(lvl1 == "A", TRUE, FALSE))

# Starting plot with doubled donuts and annotations for internal one
p <- dplyr::group_by(df, lvl1, lvl2, highlight_ext, highlight_int) |>
  dplyr::summarise(value = sum(value), .groups = "drop") |>
  packing(value, lvl1) |>
  ggplot(aes(value = value, fill = lvl1)) +
  geom_donut_int(aes(highlight = highlight_int), alpha=.5, r_int = .25) +
  geom_label_int(aes(label = "Sum {fill}:\n{.sum}-{scales::percent(.prc)}"),
                 alpha = .6, col = "white", size = 3, r=1.2) +
  geom_donut_ext(aes(opacity = lvl2, highlight = highlight_ext)) +
  scale_fill_viridis_d(option = "inferno", begin = .1, end = .7) +
  guides(alpha = guide_legend(ncol = 2), fill = guide_legend(ncol = 2)) +
  theme_void() +
  theme(legend.position = "none")

p + coord_radial(theta = "y", expand = FALSE, rotate_angle = FALSE)

# Add labels to external donut as percent inside group
p + coord_radial(theta = "y", expand = FALSE, rotate_angle = FALSE) +
  geom_label_ext(aes(label = paste0(lvl2, ": {scales::percent(.prc_grp)}")),
                 show.legend = FALSE, size=3, col="white") +
  geom_pin(size = .5, linewidth=.1, show.legend = FALSE, cut = .2)

# Leverage tv() layout
p + coord_radial(theta = "y", expand = FALSE, rotate_angle = FALSE) +
  geom_label_ext(aes(label = paste0(lvl2, ":{scales::percent(.prc_grp)}")),
                 show.legend = FALSE, size=3, col="white",
                 layout = tv(thinner = TRUE, thinner_gap = .15)) +
  geom_pin(size = .5, linewidth=.1, show.legend = FALSE, cut = .2,
           layout = tv(thinner = TRUE, thinner_gap = .15))

# Leverage another layout
p + coord_radial(theta = "y", expand = FALSE, rotate_angle = FALSE) +
  geom_label_ext(aes(label = paste0(lvl2, ": {scales::percent(.prc_grp)}")),
                 show.legend = FALSE, size=3, col="white", layout = eye()) +
  geom_pin(size = .5, linewidth=.1, show.legend = FALSE, layout = eye())
```

Index