Package ‘ggfoundry’

July 6, 2024

Type Package
Title Shape Foundry & Geom for ‘ggplot2’
Version 0.3.1
Description A ‘ggplot2’ extension that supports arbitrary hand-crafted colourable & fillable shapes. New shapes may be feature requested via a Github issue.
License MIT + file LICENSE
Encoding UTF-8
Depends ggplot2 (>= 3.5.0), R (>= 4.1)
Imports cli, grid, grImport2, lifecycle, rlang
Suggests dplyr, forcats, ggimage, httr, knitr, rmarkdown, spelling, stringr, testthat (>= 3.0.0)
Config/testthat/edition 3
RoxygenNote 7.3.1
VignetteBuilder knitr
URL https://github.com/cgoo4/ggfoundry,
    https://cgoo4.github.io/ggfoundry/
BugReports https://github.com/cgoo4/ggfoundry/issues
Language en-GB
NeedsCompilation no
Author Carl Goodwin [aut, cre, cph]
Maintainer Carl Goodwin <carl.goodwin@quantumjitter.com>
Repository CRAN
Date/Publication 2024-07-06 15:42:02 UTC

Contents

display_palette ................................................................. 2
geom_casting .................................................................. 3
shapes_cast ................................................................. 6
display_palette

Display a palette using fillable shapes

Description

[Experimental]

Creates a visualisation of a chosen palette with each colour in the selected fillable shape.

Usage

```r
display_palette(
  fill,
  pal_name,
  colour = "grey50",
  color = colour,
  shape = c("jar", "tube")
)
```

Arguments

- `fill` The colour of the shape fill.
- `pal_name` A character string for the name of the palette.
- `colour, color` The colour of the shape outline. Defaults to mid-grey to better support a website’s light and dark mode.
- `shape` A character string for the name of the shape, e.g. "jar".

Value

A ggplot2 object.

Examples

```r
display_palette(
  c("skyblue", "lightgreen", "pink", "bisque"),
  "Custom Palette Names"
)
display_palette(
  c("#9986A5", "#79402E", "#CCBA72", "#0F0D0E", "#D9D0D3", "#8D8680"),
  "Vector of Hex Codes",
  shape = "tube",
  colour = "black"
)
display_palette(
  c("#423C29", "#333031", "#8F8988", "#D2C9CB", "#AFA7A5", "#8D8680",
  "#9986A5", "#8A666E", "#7B4638", "#976C46", "#BCA365", "#988A56"
)
**Description**

[Experimental]
Arbitrary hand-crafted colourable and fillable shapes for ggplot2.
New shapes may be feature requested via a Github issue.

**Usage**

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

**Arguments**

- `mapping`  
  Set of aesthetic mappings created by `aes()`. If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.

- `data`  
  The data to be displayed in this layer. There are three options:
  - If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`.
  - A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created.
  - A function will be called with a single argument, the plot data. The return value must be a `data.frame`, and will be used as the layer data. A function can be created from a `formula` (e.g. `~ head(.x, 10)`).

- `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 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 documentation.

... Other arguments passed on to layer()’s params argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the position argument, or aesthetics that are required can not be passed through .... Unknown arguments that are not part of the 4 categories below are ignored.

- Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, `colour = "red"` or `linewidth = 3`. The geom’s documentation has an Aesthetics section that lists the available options. The ‘required’ aesthetics cannot be passed on to the params. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data.
- When constructing a layer using a stat_*() function, the ... argument can be used to pass on parameters to the geom part of the layer. An example of this is `stat_density(geom = "area", outline.type = "both")`. The geom’s documentation lists which parameters it can accept.
- Inversely, when constructing a layer using a geom_*() function, the ... argument can be used to pass on parameters to the stat part of the layer. An example of this is `geom_area(stat = "density", adjust = 0.5)`. The stat’s documentation lists which parameters it can accept.
- The key_glyph argument of layer() may also be passed on through .... This can be one of the functions described as key glyphs, to change the display of the layer in the legend.

**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().
Details

Behind the scenes, a pair of hand-drawn vector images (outline & fill) are converted into Cairo graphics library SVG files, then into grid graphical objects (grobs) for use in a ggplot2 layer.

By default, the "violin" shape is used.

If the shape is mapped to a variable, e.g. `aes(shape = factor(cyl))`, then `scale_shape_manual()` is also required to explicitly name the desired shapes as a character vector (see examples). This is because standard shapes are associated with a number, e.g. a circle is 19, whereas `geom_casting()` shapes are associated only with character strings.

In addition to the supported aesthetics below, `nudge_x`, `nudge_y`, `hjust` and `vjust` are also respected.

Value

A geom layer that can be added to a ggplot.

Aesthetics

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

- x
- y
- alpha
- angle
- colour
- fill
- group
- shape
- size

Learn more about setting these aesthetics in vignette("ggplot2-specs")

Examples

```r
library(ggplot2)

# "Baby violin" shape by default
p <- ggplot(mtcars, aes(wt, mpg))
p + geom_casting()

# Change shape & fill
p + geom_casting(shape = "box", fill = "lightgreen")

# Shapes mapped to a variable
ggplot(mtcars, aes(wt, mpg, fill = factor(cyl))) +
  geom_casting(aes(shape = factor(cyl))) +
  scale_shape_manual(values = c("violin", "dendro", "box"))
```
shapes_cast  Get the names of available shapes

Description

[Experimental]
Create a data frame of available shapes and associated sets. This may be filtered and used as a vector of strings in `scale_shape_manual()`.

Usage

shapes_cast()

Value

A data frame of available sets and shapes.

Examples

# Returns a data frame of available shapes
shapes_cast()


Index

* datasets
  geom_casting, 3

aes(), 3

borders(), 4

cast_layers (geom_casting), 3
cast_shape (geom_casting), 3
display_palette, 2

fortify(), 3

geom_casting, 3
GeomCasting (geom_casting), 3
ggplot(), 3

key glyphs, 4

layer position, 4
layer stat, 4
layer(), 4

shapes_cast, 6