Package: waffle (via r-universe)

August 25, 2024

Type Package

Title Create Waffle Chart Visualizations

Version 1.0.2

Date 2023-09-30

Maintainer Bob Rudis <bob@rud.is>

Description Square pie charts (a.k.a. waffle charts) can be used to communicate parts of a whole for categorical quantities. To emulate the percentage view of a pie chart, a 10x10 grid should be used with each square representing 1% of the total. Modern uses of waffle charts do not necessarily adhere to this rule and can be created with a grid of any rectangular shape. Best practices suggest keeping the number of categories small, just as should be done when creating pie charts. Tools are provided to create waffle charts as well as stitch them together, and to use glyphs for making isotype pictograms.

Encoding UTF-8

Copyright file inst/COPYRIGHTS

Suggests knitr, rmarkdown, dplyr, ggthemes

Depends R (>= 3.5.0), ggplot2 (>= 3.1.0)

License GPL (>= 2)

Imports RColorBrewer, grid, gridExtra, gtable, extrafont, curl, stringr, stats, htmlwidgets, DT, plyr, rlang, utils

RoxygenNote 7.2.3

Repository https://hrbrmstr.r-universe.dev

RemoteUrl https://github.com/hrbrmstr/waffle

RemoteRef HEAD

RemoteSha 767875bf15f0982f5deb6ca3be1b99b830d2b074

Contents

`waffle-package` *A package to make waffle charts (square pie charts) in R.*

Description

For glyphs:

Font Awesome by Dave Gandy - http://fontawesome.io
License: SIL OFL 1.1
URL: http://scripts.sil.org/OFL

`fa5_brand` *Font Awesome 5 Brand*

Description

'fa5_brand' is shorthand for "'FontAwesome5Brands-Regular'"

Usage

```
fa5_brand
```

Format

An object of class `character` of length 1.

fa5_solid *Font Awesome 5 Solid*

Description

'fa5_solid' is shorthand for "'FontAwesome5Free-Solid'"

Usage

```
fa5_solid
```

Format

An object of class `character` of length 1.

fa_grep *Search Font Awesome glyph names for a pattern*

Description

Search Font Awesome glyph names for a pattern

Usage

```
fa_grep(pattern)
```

Arguments

`pattern`	pattern to search for in the names of Font Awesome fonts

fa_list *List all Font Awesome glyphs*

Description

List all Font Awesome glyphs

Usage

```
fa_list()
```

geom_pictogram *Pictogram Geom*

Description

There are two special/critical `aes()` mappings:

- `label` (so the geom knows which column to map the glyphs to)
- `values` (which column you're mapping the filling for the squares with)

Usage

```
geom_pictogram(
  mapping = NULL,
  data = NULL,
  n_rows = 10,
  make_proportional = FALSE,
  flip = FALSE,
  ...,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

GeomPictogram
```

Arguments

`mapping`	Set of aesthetic mappings created by `aes()` or `aes_()`. If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.
`data`	The data to be displayed in this layer. There are three options: If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`. A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created. A `function` will be called with a single argument, the plot data. The return value must be a `data.frame.`, and will be used as the layer data.
`n_rows`	how many rows should there be in the waffle chart? default is 10
`make_proportional`	compute proportions from the raw values? (i.e. each value `n` will be replaced with `n/sum(n)`); default is `FALSE`.
`flip`	If `TRUE`, flip x and y coords. n_rows then becomes n_cols. Useful to achieve waffle column chart effect. Defaults is `FALSE`.

...	other arguments passed on to layer(). These are often aesthetics, used to set an aesthetic to a fixed value, like color = "red" or size = 3. They may also be parameters to the paired geom/stat.
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().

Format

An object of class GeomPictogram (inherits from GeomText, Geom, ggproto, gg) of length 5.

---

geom_waffle *Waffle (Square pie chart) Geom*

Description

There are two special/critical aes() mappings:

- fill (so the geom knows which column to map the fills to)
- values (which column you're mapping the filling for the squares with)

Usage

```
geom_waffle(
  mapping = NULL,
  data = NULL,
  n_rows = 10,
  make_proportional = FALSE,
  flip = FALSE,
  na.rm = FALSE,
  show.legend = NA,
  radius = grid::unit(0, "npc"),
  inherit.aes = TRUE,
  ...
)

GeomWaffle

stat_waffle(
  mapping = NULL,
  data = NULL,
```

```
  geom = "waffle",
  n_rows = 10,
  make_proportional = FALSE,
  flip = FALSE,
  radius = grid::unit(0, "npc"),
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  ...
)

StatWaffle
```

Arguments

mapping	Set of aesthetic mappings created by `aes()` or `aes_()`. If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`. A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created. A `function` will be called with a single argument, the plot data. The return value must be a `data.frame.`, and will be used as the layer data.
n_rows	how many rows should there be in the waffle chart? default is 10
make_proportional	compute proportions from the raw values? (i.e. each value `n` will be replaced with `n/sum(n)`); default is `FALSE`.
flip	If `TRUE`, flip x and y coords. n_rows then becomes n_cols. Useful to achieve waffle column chart effect. Defaults is `FALSE`.
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.
radius	radius for round squares
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()`.
...	other arguments passed on to `layer()`. These are often aesthetics, used to set an aesthetic to a fixed value, like `color = "red"` or `size = 3`. They may also be parameters to the paired geom/stat.
geom	geom to use (default is "waffle")

Format

An object of class `GeomWaffle` (inherits from `GeomRtile`, `GeomRrect`, `Geom`, `ggproto`, `gg`) of length 5.

An object of class `StatWaffle` (inherits from `Stat`, `ggproto`, `gg`) of length 8.

Examples

```
data.frame(
  parts = factor(rep(month.abb[1:3], 3), levels=month.abb[1:3]),
  vals = c(10, 20, 30, 6, 14, 40, 30, 20, 10),
  fct = c(rep("Thing 1", 3), rep("Thing 2", 3), rep("Thing 3", 3))
) -> xdf

ggplot(xdf, aes(fill = parts, values = vals)) +
  geom_waffle() +
  facet_wrap(~fct)
```

`install_fa_fonts` *Install Font Awesome 5 Fonts*

Description

Install Font Awesome 5 Fonts

Usage

```
install_fa_fonts()
```

`iron` *Veritical, left-aligned layout for waffle plots*

Description

Left-align the waffle plots by x-axis. Use the `pad` parameter in `waffle` to pad each plot to the max width (num of squares), otherwise the plots will be scaled.

Usage

```
iron(...)
```

Arguments

`...`	one or more waffle plots

Examples

```
parts <- c(80, 30, 20, 10)
w1 <- waffle(parts, rows=8)
w2 <- waffle(parts, rows=8)
w3 <- waffle(parts, rows=8)
# print chart
## iron(w1, w2, w3)
```

`scale_label_pictogram` *Used with geom_pictogram() to map Font Awesome fonts to labels*

Description

Used with geom_pictogram() to map Font Awesome fonts to labels

Usage

```
scale_label_pictogram(..., values, aesthetics = "label")
```

Arguments

`...`	dots
`values`	values
`aesthetics`	aesthetics

`theme_enhance_waffle` *Waffle chart theme cruft remover that can be used with any other theme*

Description

Removes:

Usage

```
theme_enhance_waffle()
```

Details

- panel grid
- all axis text
- all axis ticks
- all axis titles

`waffle` *Make waffle (square pie) charts*

Description

Given a named vector or a data frame, this function will return a ggplot object that represents a waffle chart of the values. The individual values will be summed up and each that will be the total number of squares in the grid. You can perform appropriate value transformation ahead of time to get the desired waffle layout/effect.

Usage

```
waffle(
  parts,
  rows = 10,
  keep = TRUE,
  xlab = NULL,
  title = NULL,
  colors = NA,
  size = 2,
  flip = FALSE,
  reverse = FALSE,
  equal = TRUE,
  pad = 0,
  use_glyph = FALSE,
  glyph_size = 12,
  glyph_font = "Font Awesome 5 Free Solid",
  glyph_font_family = "FontAwesome5Free-Solid",
  legend_pos = "right"
)
```

Arguments

`parts`	named vector of values or a data frame to use for the chart
`rows`	number of rows of blocks
`keep`	keep factor levels (i.e. for consistent legends across waffle plots)
`xlab`	text for below the chart. Highly suggested this be used to give the "1 sq == xyz" relationship if it's not obvious
`title`	chart title
`colors`	exactly the number of colors as values in `parts`. If omitted, Color Brewer "Set2" colors are used.
`size`	width of the separator between blocks (defaults to `2`)
`flip`	flips x & y axes
`reverse`	reverses the order of the data

`equal` by default, waffle uses `coord_equal`; this can cause layout problems, so you an use this to disable it if you are using ggsave or knitr to control output sizes (or manually sizing the chart)

`pad` how many blocks to right-pad the grid with

`use_glyph` use specified glyph; if using built-in Font Awesome, can be the glyph name; otherwise, it must be the unicode glyph from the custom font the caller is using.

`glyph_size` size of the Font Awesome font

`glyph_font, glyph_font_family`
if `use_glyph` is not `FALSE`, the `gplyph_font` will be looked up in the font database and the `glpyph_font_family` used as the `family` parameter to ggplot for font display since fonts in R, Python and anythign that relies on legacy font C libraries are woefully messed up. You may need to adjust either of these "font" parameters depending on your system & OS version due to the fact that font names are different even between OS versions (sometimes).

The package comes with Font Awesome and helpers for it. Use of any other fonts requires the caller to be familiar with using fonts in R. NOT ALL FONTS will work with ggplot2 and definitely not under all graphics devices for ggplot2.

`legend_pos` position of legend

Details

If a data frame is used, the first two columns should contain the desired names and the values, respectively.

If the vector is not named or only partially named, capital letters will be used instead.

It is highly suggested that you limit the number of elements to plot, just like you should if you ever got wasted and decided that a regular pie chart was a good thing to create and then decide to be totally evil and make one to pollute this beautiful world of ours.

Chart title and x-axis labels are optional, especially if you'll just be exporting to another program for use/display.

If you specify a string (vs `FALSE`) to `use_glyph` the function will map the input to a Font Awesome glyph name and use that glyph for the tile instead of a block (making it more like an isotype pictogram than a waffle chart). You'll need to install Font Awesome 5 and use the `extrafont` package to be able to use Font Awesome 5 glyphs. Sizing is also up to the user since fonts do not automatically scale with graphic resize.

Glyph idea inspired by Ruben C. Arslan (@_r_c_a)

Note

You MUST use the Font Awesome 5 fonts bundled with the package. See `install_fa_fonts()`.

Examples

```
parts <- c(80, 30, 20, 10)
waffle(parts, rows=8)
```

```
parts <- data.frame(
  names = LETTERS[1:4],
  vals = c(80, 30, 20, 10)
)

waffle(parts, rows=8)

# library(extrafont)
# waffle(parts, rows=8, use_glyph="shield")

parts <- c(One=80, Two=30, Three=20, Four=10)
chart <- waffle(parts, rows=8)
# print(chart)
```

Index