

Package ‘waffle’

January 7, 2017

Type Package

Title Create Waffle Chart Visualizations in R

Version 0.7.0

Date 2017-01-07

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.

URL <https://github.com/hrbrmstr/waffle/tree/cran>

BugReports <https://github.com/hrbrmstr/waffle/issues>

Suggests testthat

Depends R (>= 3.2.0), ggplot2 (>= 2.0.0)

License GPL (>= 2)

Imports RColorBrewer, grid, gridExtra, gtable, extrafont

RoxygenNote 5.0.1

NeedsCompilation no

Author Bob Rudis [aut, cre],
Dave Gandy [aut] (Font Awesome)

Repository CRAN

Date/Publication 2017-01-07 15:59:43

R topics documented:

waffle-package	2
fa_grep	2
fa_list	2
iron	3
waffle	3

Index	6
--------------	----------

waffle-package	<i>A package to make waffle charts (square pie charts) in R.</i>
----------------	--

Description

For glyphs:

Font Awesome by Dave Gandy - <http://fontawesome.io>

License: SIL OFL 1.1

URL: <http://scripts.sil.org/OFL>

fa_grep	<i>Search FontAwesome names for a pattern</i>
---------	---

Description

Search FontAwesome names for a pattern

Usage

```
fa_grep(pattern)
```

Arguments

pattern pattern to search for in the names of FontAwesome fonts

fa_list	<i>List all FontAwesome names</i>
---------	-----------------------------------

Description

List all FontAwesome names

Usage

```
fa_list()
```

`iron`*Vertical, 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)
```

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

Description

Given a named vector, 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, legend_pos = "right")
```

Arguments

<code>parts</code>	named vector of values to use for the chart
<code>rows</code>	number of rows of blocks
<code>keep</code>	keep factor levels (i.e. for consistent legends across waffle plots)
<code>xlab</code>	text for below the chart. Highly suggested this be used to give the "1 sq == xyz" relationship if it's not obvious
<code>title</code>	chart title
<code>colors</code>	exactly the number of colors as values in <code>parts</code> . If omitted, Color Brewer "Set2" colors are used.
<code>size</code>	width of the separator between blocks (defaults to 2)
<code>flip</code>	flips x & y axes
<code>reverse</code>	reverses the order of the data
<code>equal</code>	by default, waffle uses <code>coord_equal</code> ; this can cause layout problems, so you can use this to disable it if you are using <code>ggsave</code> or <code>knitr</code> to control output sizes (or manually sizing the chart)
<code>pad</code>	how many blocks to right-pad the grid with
<code>use_glyph</code>	use specified FontAwesome glyph
<code>glyph_size</code>	size of the FontAwesome font
<code>legend_pos</code>	position of legend

Details

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 FontAwesome 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 actually install FontAwesome and use the `extrafont` package (<https://github.com/wch/extrafont>) to be able to use the FontAwesome 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)

Examples

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

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

waffle

5

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

Index

[fa_grep](#), 2

[fa_list](#), 2

[iron](#), 3

[waffle](#), 3

[waffle-package](#), 2