Package 'ggbrick'

February 8, 2026

Type Package

Title Waffle Style Chart with a Brick Layout in 'ggplot2'

Version 0.3.2

Maintainer Daniel Oehm <danieloehm@gmail.com>

Description A new take on the bar chart. Similar to a waffle style chart but instead of squares the layout resembles a brick wall.

Depends R (>= 3.5.0)

Imports ggplot2, dplyr, purrr, glue

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

NeedsCompilation no

Author Daniel Oehm [aut, cre]

Repository CRAN

Date/Publication 2026-02-08 19:50:02 UTC

Contents

brick_row 2
build_wall 2
build_wall_waffle 3
coord_brick 3
GeomBrick 5
GeomBrick0 5
GeomWaffle 6
GeomWaffle0 6
half_brick_row 7
make_new_fill 7
robust_random 8
robust_round 8
stat_brick 9
stat_waffle 11
switch_pos 14

Index **15**

`brick_row` *Brick row*

Description

Brick row

Usage

```
brick_row(
  layer,
  bpl,
  brick_height = 1,
  brick_width = 2.5,
  gap = 0.125,
  width = 0.9,
  .geom = "brick"
)
```

Arguments

`layer`	Brick layer.
`bpl`	Number of bricks in the layer.
`brick_height`	Brick height.
`brick_width`	Brick width.
`gap`	Gap between the bricks.
`width`	Column width
`.geom`	Geom type for layering. Either 'brick' or 'brick_waffle'

`build_wall` *Build the wall*

Description

Build the wall

Usage

```
build_wall(n_bricks, height, bpl, gap = NULL, width = 0.9)
```

Arguments

n_bricks	Number of bricks
height	Height of the wall.
bpl	Bricks per layer
gap	The space between bricks.
width	Column_width

---

build_wall_waffle *Build the wall*

Description

Build the wall

Usage

```
build_wall_waffle(n_bricks, height, bpl, gap = NULL, width = 0.9)
```

Arguments

n_bricks	Number of bricks
height	Height of the wall.
bpl	Bricks per layer
gap	The space between bricks.
width	Column width.

---

coord_brick *Cartesian coordinates with fixed "aspect ratio"*

Description

A fixed scale coordinate system forces a specified ratio similar to `coord_fixed`. It holds the coordinates fixed at the right ratio to ensure each brick is of the right dimensions.

Usage

```
coord_brick(
  bricks_per_layer = 4,
  ratio = NULL,
  xlim = NULL,
  ylim = NULL,
  expand = TRUE,
  clip = "on",
  width = 0.9
)

coord_waffle(
  bricks_per_layer = 4,
  ratio = NULL,
  xlim = NULL,
  ylim = NULL,
  expand = TRUE,
  clip = "on",
  width = 0.9
)
```

Arguments

bricks_per_layer	Number of bricks per layer. Should match the `bricks_per_layer` specification in `geom_brick`. Default is `4`.
ratio	aspect ratio, expressed as `y / x`
xlim, ylim	Limits for the x and y axes.
expand	If `TRUE`, the default, adds a small expansion factor to the limits to ensure that data and axes don't overlap. If `FALSE`, limits are taken exactly from the data or `xlim`/`ylim`.
clip	Should drawing be clipped to the extent of the plot panel? A setting of `"on"` (the default) means yes, and a setting of `"off"` means no. In most cases, the default of `"on"` should not be changed, as setting `clip = "off"` can cause unexpected results. It allows drawing of data points anywhere on the plot, including in the plot margins.
width	Column width. If using a different `width` in `geom_brick` use the same `width` here to ensure correct scaling.

Value

ggplot object

Examples

```
# ensures that the ranges of axes are equal to the specified ratio by
```

```
library(ggplot2)
library(dplyr)

# create a base plot
plt <- mpg %>%
  count(class, drv) %>%
  ggplot() +
  geom_brick(aes(class, n, fill = drv), bricks_per_layer = 6)

# view the base plot
plt

# View the base plot with fixed coords
# Ensure `bricks_per_layer` matches the geom
plt %>%
  coord_brick(6)

# The same using `geom_waffle`
mpg %>%
  count(class, drv) %>%
  ggplot() +
  geom_waffle(aes(class, n, fill = drv), bricks_per_layer = 6) +
  coord_waffle(6)
```

`GeomBrick` *GeomBrick*

Description

GeomBrick

Usage

```
GeomBrick
```

Format

An object of class `GeomBrick` (inherits from `GeomRect`, `Geom`, `ggproto`, gg) of length 6.

`GeomBrick0` *GeomBrick*

Description

GeomBrick

Usage

```
GeomBrick0
```

Format

An object of class GeomBrick0 (inherits from GeomBrick, GeomRect, Geom, ggproto, gg) of length 6.

GeomWaffle *GeomBrick*

Description

GeomBrick

Usage

```
GeomWaffle
```

Format

An object of class GeomWaffle (inherits from GeomRect, Geom, ggproto, gg) of length 6.

GeomWaffle0 *GeomBrick*

Description

GeomBrick

Usage

```
GeomWaffle0
```

Format

An object of class GeomWaffle0 (inherits from GeomWaffle, GeomRect, Geom, ggproto, gg) of length 6.

half_brick_row *half brick row*

Description

half brick row

Usage

```
half_brick_row(
  layer,
  bpl,
  brick_height = 1,
  brick_width = 2.5,
  gap = 0.125,
  width = 0.9
)
```

Arguments

layer	Brick layer.
bpl	Number of bricks in the layer.
brick_height	Brick height.
brick_width	Brick width.
gap	Gap between the bricks.
width	Column_width

make_new_fill *Fill*

Description

Makes the vector for the fill aesthetic

Usage

```
make_new_fill(fill, n, val)
```

Arguments

fill	The fill vector.
n	Vector representing the number of bricks for the fill level.
val	Vector of length the same as fill of with 1 o 0.5 for whole or half bricks.

robust_random *Robust random*

Description

Ensures the half bricks are randomised in pairs to preserve the total

Usage

```
robust_random(x, val)
```

Arguments

x	x.
val	Value.

robust_round *Robust round*

Description

Robust round

Usage

```
robust_round(x, N)
```

Arguments

x	Vector of values.
N	Value to preserve sum to.

stat_brick *stat_brick*

Description

Creates a 'waffle' style chart with the aesthetic of a brick wall. Usage is similar to `geom_col` where you supply counts as the height of the bar. Each whole brick represents 1 unit. Two half bricks equal one whole brick. Where the count exceeds the number of brick layers, the number of bricks is scaled to retain the brick wall aesthetic.

Usage

```
stat_brick(
  mapping = NULL,
  data = NULL,
  geom = "rect",
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  width = 0.9,
  bricks_per_layer = 4,
  type = "ordered",
  gap = NULL,
  ...
)

geom_brick(
  mapping = NULL,
  data = NULL,
  stat = "brick",
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  bricks_per_layer = 4,
  width = 0.9,
  type = "ordered",
  gap = NULL,
  ...
)

geom_brick0(
  mapping = NULL,
  data = NULL,
  stat = "brick",
  position = "identity",
```

```
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  bricks_per_layer = 4,
  type = "ordered",
  gap = 0,
  width = 0.9,
  ...
)
```

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)).
geom	Geom
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 removes NAs from the data.
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().
width	Column width. Default value is 0.9.
bricks_per_layer	The number of bricks per layer. Default 4.
type	The type of fill ordering. one of 'ordered', 'random' or 'soft_random', Default 'ordered'
gap	The space between bricks.
...	Dots.
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")

Value

ggplot object

Aesthetics

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

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

Examples

```
library(ggplot2)
library(dplyr)
mpg %>%
  count(class, drv) %>%
  ggplot() +
  geom_brick(aes(class, n, fill = drv)) +
  coord_brick()
```

`stat_waffle` *stat_brick*

Description

Creates a 'waffle' style chart with the aesthetic of a brick wall. Usage is similar to `geom_col` where you supply counts as the height of the bar. Each whole brick represents 1 unit. Two half bricks equal one whole brick. Where the count exceeds the number of brick layers, the number of bricks is scaled to retain the brick wall aesthetic.

Usage

```
stat_waffle(
  mapping = NULL,
  data = NULL,
  geom = "rect",
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
```

```
  bricks_per_layer = 4,
  type = "ordered",
  gap = NULL,
  width = 0.9,
  ...
)

geom_waffle(
  mapping = NULL,
  data = NULL,
  stat = "waffle",
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  bricks_per_layer = 4,
  type = "ordered",
  gap = NULL,
  width = 0.9,
  ...
)

geom_waffle0(
  mapping = NULL,
  data = NULL,
  stat = "waffle",
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  bricks_per_layer = 4,
  type = "ordered",
  gap = 0,
  width = 0.9,
  ...
)
```

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)).
geom	Geom
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 removes NAs from the data.
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().
bricks_per_layer	The number of bricks per layer. Default 4.
type	The type of fill ordering. one of 'ordered', 'random' or 'soft_random', Default 'ordered'
gap	The space between bricks.
width	Column width. Default 0.9.
...	Dots.
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")

Value

ggplot object

Aesthetics

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

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

Examples

```
library(ggplot2)
library(dplyr)
mpg %>%
  count(class, drv) %>%
  ggplot() +
  geom_waffle(aes(class, n, fill = drv)) +
  coord_waffle()
```

`switch_pos` *Switch position for soft random*

Description

Switch position for soft random

Usage

```
switch_pos(x, n)
```

Arguments

x	Vector to switch values in.
n	Number to switch.

Index

∗ **datasets**
GeomBrick, 5
GeomBrick0, 5
GeomWaffle, 6
GeomWaffle0, 6

brick *(*stat_brick*)*, 9
brick_row, 2
build_wall, 2
build_wall_waffle, 3

coord_brick, 3
coord_waffle *(*coord_brick*)*, 3

geom_brick *(*stat_brick*)*, 9
geom_brick0 *(*stat_brick*)*, 9
geom_waffle *(*stat_waffle*)*, 11
geom_waffle0 *(*stat_waffle*)*, 11
GeomBrick, 5
GeomBrick0, 5
GeomWaffle, 6
GeomWaffle0, 6

half_brick_row, 7

make_new_fill, 7

robust_random, 8
robust_round, 8

stat_brick, 9
stat_waffle, 11
switch_pos, 14

waffle *(*stat_waffle*)*, 11