Package ‘brickr’

April 6, 2020

Title Emulate LEGO Bricks in 2D and 3D

Version 0.3.2

Description Generate digital LEGO models using 'tidyverse' functions.
Convert image files into 2D and 3D LEGO mosaics, complete with piece counts and instructions.
Render 3D models using simple data frame instructions.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 3.0.2)

Imports ggplot2, magrittr, dplyr, tidyr, purrr, scales, farver, colorspace, rgl

Suggests knitr, rmarkdown, gridExtra, png, jpeg, tibble, raster, stringr

RoxygenNote 7.0.2

URL https://github.com/ryantimpe/brickr

BugReports https://github.com/ryantimpe/brickr/issues

Collate 'brickr.R' 'bricks-from-mosaic.R' 'bricks-from-tables.R'
'build-bricks-rgl.R' 'build-instructions.R' 'build-mosaic.R'
'collect-bricks.R' 'colors-and-themes.R' 'data.R'
'geom-brick-rect.R' 'image-to-mosaic.R'
'image-to-prep-mosaic.R' 'piece-count.R' 'utils-pipe.R'

VignetteBuilder knitr

NeedsCompilation no

Author Ryan Timpe [aut, cre]

Maintainer Ryan Timpe <ryan.timpe@gmail.com>

Repository CRAN

Date/Publication 2020-04-06 17:42:06 UTC
**R topics documented:**

- **brickr** .................................................... 2
- **bricks_from_coords** ................................. 2
- **bricks_from_excel** ................................. 4
- **bricks_from_mosaic** .............................. 6
- **bricks_from_table** ............................... 7
- **build_bricks** ......................................... 10
- **build_colors** ......................................... 11
- **build_instructions** ............................... 12
- **build_mosaic** ......................................... 13
- **build_pieces** ......................................... 14
- **build_pieces_table** .............................. 15
- **GeomBrick** ........................................... 16
- **image_to_mosaic** .................................. 16
- **lego_colors** .......................................... 18

**Index**

```
brickr  'brickr' package
```

**Description**

Emulate LEGO Bricks in 2D and 3D

```
bricks_from_coords  Create a 3D model object from a long coordinate data frame
```

**Description**

Convert a data frame with x, y, z & Color columns into a 3D object

**Usage**

```r
bricks_from_coords(
  coord_table,
  use_bricks = NULL,
  increment_level = 0,
  min_level = 1,
  max_level = Inf,
  increment_x = 0,
  max_x = Inf,
  increment_y = 0,
  max_y = Inf,
  exclude_color = NULL,
  exclude_level = NULL
)
```
**bricks_from_coords**

Arguments

- **coord_table**
  A data frame of a 3D brick model design. Contains 'x', 'y', and 'z' (vertical height) dimensions, as well as 'Color' from official LEGO color names. See `build_colors`. Optional column 'piece_type' for shapes other than rectangular bricks. Optional column 'mid_Level' with values 0, 1, or 2 (default 0) for 1-height placement of bricks.

- **use_bricks**
  Array of brick sizes to use in mosaic. Defaults to c('4x2','3x2','2x2','3x1','2x1','1x1').

- **increment_level**
  Default '0'. Use in animations. Shift Level/z dimension by an integer.

- **min_level**
  Default '1'. Use in animations. Any Level/z values below this value will be cut off.

- **max_level**
  Default 'Inf'. Use in animations. Any Level/z values above this value will be cut off.

- **increment_x**
  Default '0'. Use in animations. Shift x dimension by an integer.

- **max_x**
  Default 'Inf'. Use in animations. Any x values above this value will be cut off.

- **increment_y**
  Default '0'. Use in animations. Shift y dimension by an integer.

- **max_y**
  Default 'Inf'. Use in animations. Any y values above this value will be cut off.

- **exclude_color**
  Numeric array of color ID numbers to exclude.

- **exclude_level**
  Numeric array of Level/z dimensions to exclude.

Value

A list with elements Img_lego to pass to `build_bricks`.

See Also

Other 3D Models: `bricks_from_excel()`, `bricks_from_mosaic()`, `bricks_from_table()`, `build_bricks()`

Examples

```r
#This is a 1x4 yellow brick
brick <- data.frame(
  x = 1:4,
  y = 1, z=1,
  color = "Bright yellow",
  stringsAsFactors=FALSE)

brick %>%
  bricks_from_coords() %>%
  build_bricks()
rgl::clear3d()

#This is a lot of bricks
bricks <- expand.grid(
  x = 1:4,
  y = 1:2,
```
z = 1:3)
#Color them in sets of these 3 options
bricks$color <- rep(rep(c("Bright yellow", "Bright red", "Tr. green"), each=4), 2)

bricks %>%
  bricks_from_coords() %>%
  build_bricks()

rgl::clear3d()

#Use different brick shapes by added a 'piece_type' column
bricks$piece_type <- "c1"  #Make all the pieces cylinders

bricks %>%
  bricks_from_coords() %>%
  build_bricks()

rgl::clear3d()

---

**bricks_from_excel**

Convert an Excel 'brickr' template into a 3D object

**Description**

Build a 3D model from an Excel template. A single data frame includes both the instructions and the color guides.

**Usage**

```r
bricks_from_excel(
  excel_table,  # Required
  piece_table = NULL,  # Optional
  use_bricks = NULL,  # Optional
  repeat_levels = 1,  # Optional
  increment_level = 0,  # Optional
  min_level = 1,  # Optional
  max_level = Inf,  # Optional
  increment_x = 0,  # Optional
  max_x = Inf,  # Optional
  increment_y = 0,  # Optional
  max_y = Inf,  # Optional
  exclude_color = NULL,  # Optional
  exclude_level = NULL  # Optional
)
```
## bricks_from_excel

### Arguments

- **excel_table**: Sheet imported from a brickr Excel template to build model. This differs slightly from `bricks_from_table` because a single data frame has both the brick coordinates and color table.
- **piece_table**: Sheet identical in shape to `excel_table` with piece shape IDs.
- **use_bricks**: Array of brick sizes to use in mosaic. Defaults to `c(4x2', '3x2', '2x2', '3x1', '2x1', '1x1')`.
- **repeat_levels**: How many times to repeat a level. Can save time in model planning. Default is 1.
- **increment_level**: Default '0'. Use in animations. Shift Level/z dimension by an integer.
- **min_level**: Default '1'. Use in animations. Any Level/z values below this value will be cut off.
- **max_level**: Default 'Inf'. Use in animations. Any Level/z values above this value will be cut off.
- **increment_x**: Default '0'. Use in animations. Shift x dimension by an integer.
- **max_x**: Default 'Inf'. Use in animations. Any x values above this value will be cut off.
- **increment_y**: Default '0'. Use in animations. Shift y dimension by an integer.
- **max_y**: Default 'Inf'. Use in animations. Any y values above this value will be cut off.
- **exclude_color**: Numeric array of color ID numbers to exclude.
- **exclude_level**: Numeric array of Level/z dimensions to exclude.

### Value

A list with elements `Img_lego` to pass to `build_bricks`.

### See Also

Other 3D Models: `bricks_from_coords()`, `bricks_from_mosaic()`, `bricks_from_table()`, `build_bricks()`

### Examples

```r
#Demo table in same format as Excel template
#This creates a 1x3 red brick.
demo_excel <- tibble::tribble(
  ~Level, ~"1", ~"2", ~"3", ~user_color, ~LEGO_color,
  "A", 1, 1, 1, 1, "Bright red"
)

demo_excel %>%
bricks_from_excel() %>%
build_bricks()

rgl::clear3d()

#To change the pieces, import a second table in the same shape, but with piece IDs.
```
demo_pieces <- tibble::tribble(
  ~Level, ~"1", ~"2", ~"3",
  "A", "w4", "c1", "w2"
)

demo_excel %>%
  bricks_from_excel(piece_table = demo_pieces) %>%
  build_bricks()

rgl::clear3d()

---

bricks_from_mosaic  
*Convert a 2D LEGO mosaic into a 'brickr' 3D object*

**Description**

Stacks LEGO plates to create a 3D version of the 2D brick mosaics. Height of bricks determined by brightness of color.

**Usage**

```r
bricks_from_mosaic(mosaic_list, mosaic_height = 6, highest_el = "light")
```

**Arguments**

- `mosaic_list`  
  List output from `image_to_bricks()`. Contains an element `Img_lego`.
- `mosaic_height`  
  Number of layers in the 3D image.
- `highest_el`  
  Brick height is determined by brightness of color. Use `highest_el = 'dark'` for darkest bricks to have `mosaic_height`.

**Value**

A list with elements `Img_lego` to pass to `build_bricks`.

**See Also**

Other 3D Models: `bricks_from_coords()`, `bricks_from_excel()`, `bricks_from_table()`, `build_bricks()`

**Examples**

```r
# Import a jpeg or png
demo_file <- system.file("extdata", "demo_img.jpg", 
  package = "brickr", mustWork = TRUE)
demo_image <- jpeg::readJPEG(demo_file)

#Begin with a 24x24 mosaic object
mosaic <- demo_image %>%
```
## bricks_from_table

**Convert a table into a 'brickr' 3D object**

### Description

Create a 3D brick object from a data frame. Left-most column is level/height/z dimension, with rows as Y axis and columns as X axis.

### Usage

```r
bricks_from_table(
  matrix_table,
  color_guide = brickr::lego_colors,
  piece_matrix = NULL,
  use_bricks = NULL,
  .re_level = TRUE,
  increment_level = 0,
  min_level = 1,
  max_level = Inf,
  increment_x = 0,
  max_x = Inf,
  increment_y = 0,
  max_y = Inf,
  exclude_color = NULL,
  exclude_level = NULL
)
```
Arguments

- **matrix_table**: A data frame of a 3D brick model design. Left-most column is level/height/z dimension, with rows as Y axis and columns as X axis. See example. Use `tribble` for ease.

- **color_guide**: A data frame linking numeric `.value` in `matrix_table` to official LEGO color names. Defaults to data frame `lego_colors`.

- **piece_matrix**: A data frame in same shape as `matrix_table` with piece shape IDs.

- **use_bricks**: Array of brick sizes to use in mosaic. Defaults to `c("4x2","2x2","3x1","2x1","1x1")`. ‘1x1’ will always be considered.

- **.re_level**: Logical to reassign the Level/z dimension to layers in alphanumeric order. Set to FALSE to explicitly provide levels.

- **increment_level**: Default ‘0’. Use in animations. Shift Level/z dimension by an integer.

- **min_level**: Default ‘1’. Use in animations. Any Level/z values below this value will be cut off.

- **max_level**: Default ‘Inf’. Use in animations. Any Level/z values above this value will be cut off.

- **increment_x**: Default ‘0’. Use in animations. Shift x dimension by an integer.

- **max_x**: Default ‘Inf’. Use in animations. Any x values above this value will be cut off.

- **increment_y**: Default ‘0’. Use in animations. Shift y dimension by an integer.

- **max_y**: Default ‘Inf’. Use in animations. Any y values above this value will be cut off.

- **exclude_color**: Numeric array of color ID numbers to exclude.

- **exclude_level**: Numeric array of Level/z dimensions to exclude.

Value

A list with elements `Img_lego` to pass to `build_bricks`.

See Also

Other 3D Models: `bricks_from_coords()`, `bricks_from_excel()`, `bricks_from_mosaic()`, `build_bricks()`

Examples

```r
#This is a 4x2 brick. One level high, 2 x-values (columns), 4 y-values (rows).
brick <- data.frame(
  Level="A",
  X1 = rep(3,4), #The number 3 is the brickID for 'bright red'
  X2 = rep(3,4)
)

brick %>%
  bricks_from_table() %>%
  build_bricks()
```
# Build on top of each other by changing the Level value.
# This example builds a blue 2x2 brick on top of a red 2x2 brick.
brick <- data.frame(
  Level = c("A", "A", "B", "B"),
  X1 = c(3, 3, 4, 4), # 3 is red, 4 is blue
  X2 = c(3, 3, 4, 4)
)

brick %>%
  bricks_from_table() %>%
  build_bricks()

rgl::clear3d()

# Provide an additional piece_matrix argument to change the default brick shape.
pieces <- data.frame(
  Level = c("A", "A", "B", "B"),
  X1 = c("b", "b", "p", "p"), # b is brick (default), p is plate
  X2 = c("b", "b", "p", "p")
)

brick %>%
  bricks_from_table(piece_matrix = pieces) %>%
  build_bricks()

rgl::clear3d()

# Provide a custom table of colors
custom_colors <- data.frame(
  .value = c(3, 4),
  Color = c("Bright orange", "Dark green")
)

brick %>%
  bricks_from_table(color_guide = custom_colors) %>%
  build_bricks()

rgl::clear3d()

# Limit the size of bricks used in the model with use_bricks

brick %>%
  bricks_from_table(use_bricks = "2x1") %>% # Only use 2x1 bricks.
  build_bricks()

rgl::clear3d()
**Description**

Render the output of any of the bricks_from_* functions as a 3D model. Opens an 'rgl' window.

**Usage**

```r
build_bricks(
    brick_list,
    background_color = "white",
    rgl_lit = TRUE,
    outline_bricks = FALSE,
    trans_alpha = 0.5,
    view_levels = NULL
)
```

**Arguments**

- `brick_list`: List output from a bricks_from_* function. Contains an element `Img_lego`.
- `background_color`: Default 'white'. Color of the background.
- `rgl_lit`: Default 'TRUE'. Include RGL lighting features in rendering.
- `outline_bricks`: Default 'FALSE'. Include black outlines around brick edges. Set to 'TRUE' and `rgl_lit='FALSE'` for cartoon-looking bricks.
- `trans_alpha`: Default 0.5. Alpha level for transparent bricks.
- `view_levels`: Numeric array of Levels/z values to display. Leave as 'NULL' to include all.

**Value**

3D brick model rendered in the 'rgl' package.

**See Also**

Other 3D Models: `bricks_from_coords()`, `bricks_from_excel()`, `bricks_from_mosaic()`, `bricks_from_table()`

**Examples**

```r
#This is a brick
brick <- data.frame(
    Level="A",
    X1 = rep(3,4), #The number 3 is the brickrID for 'bright red'
    X2 = rep(3,4)
)
```
#Convert the dataframe to a list object that can be rendered
brick_object <- brick %>%
  bricks_from_table()

#Render it
brick_object %>%
  build_bricks()

rgl::clear3d()

#Combine the option rgl_lit=FALSE & outline_bricks=TRUE
# This makes the rendering look like a drawing
brick_object %>%
  build_bricks(outline_bricks = TRUE, rgl_lit = FALSE,
               background_color = "#99e7ff")

rgl::clear3d()

---

## build_colors

**Display available brick colors**

**Description**

Generates a plot of available brick colors. These names must be used exactly when creating custom name lists. There are 41 solid brick color names and 13 transparent colors. Transparent colors are not used in mosaics.

**Usage**

```
build_colors(.names_only = FALSE, include_transparent = TRUE)
```

**Arguments**

- `.names_only` Return an array of the 41 solid brick color names and 13 transparent colors. Does not plot.
- `include_transparent` Include transparent colors in the plot output.

**Details**

Use `.names_only = TRUE` to get a list of color names.

**Value**

An array or `ggplot` of brick colors & ID numbers.

**See Also**

Other Resources: `build_instructions()`, `build_pieces_table()`, `build_pieces()`
Build Instructions

Create instruction manual for a 2D mosaic or 3D model

Examples

# Generate plot of colors
build_colors(include_transparent = FALSE)

# Print list of colors
build_colors(TRUE)

Description

Render faceted plot of instructions for 2D mosaics or 3D model objects. For mosaics, can specify the number of steps.

Usage

build_instructions(brickr_obj, num_steps = 6)

Arguments

brickr_obj     brickr mosaic or 3D model object.
num_steps     Number of discrete steps in instruction manual, for mosaics only.

Details

Instructions for 2D mosaics are split into sections beginning at the bottom of the image. This makes it easier to follow each row when building an actual brick mosaic.

3D model instructions are displayed one Level (z value) at a time. The current model level is clearly displayed, while the previous level is shown as transparent.

Value

A single plot object of steps to build brickr model or mosaic.

See Also

Other Resources: build_colors(), build_pieces_table(), build_pieces()

Examples

# Import a jpeg or png
demo_file <- system.file("extdata", "demo_img.jpg",
                        package = "brickr", mustWork = TRUE)
demo_image <- jpeg::readJPEG(demo_file)

# Create a mosaic object
mosaic <- demo_image %>%
  image_to_mosaic(img_size = 24)

# Rather than drawing the mosaic, use build_instructions() to draw instructions
mosaic %>%
  build_instructions()

# Change the number of steps for more detail
mosaic %>%
  build_instructions(num_steps = 9)

---

build_mosaic  
Display 2D LEGO mosaic as a plot image

Description
Render a plot image of the 2D brick mosaic with optional title.

Usage
build_mosaic(brick_obj, title = NULL)

Arguments
- brick_obj: List output from image_to_bricks(). Contains an element Img_lego.
- title: Optional title to include above plotted mosaic.

Value
A single plot object to display 2D mosaic.

See Also
Other Mosaics: image_to_mosaic()

Examples

# Import a jpeg or png
demo_file <- system.file("extdata", "demo_img.jpg",  
  package = "brickr", mustWork = TRUE)
demo_image <- jpeg::readJPEG(demo_file)
```r
# Build a very small 12x12 mosaic.

demo_image %>%
  image_to_mosaic(12) %>%
  build_mosaic()

# Build a mosaic in the default size of 48x48 studs with title

demo_image %>%
  image_to_mosaic() %>%
  build_mosaic("Demo mosaic")
```

<table>
<thead>
<tr>
<th>build_pieces</th>
<th>Display bricks required to build model or mosaic</th>
</tr>
</thead>
</table>

**Description**

Create a chart of brick colors and sizes used in a brick mosaic or model.

**Usage**

```r
build_pieces(brick_obj)
```

**Arguments**

- `brick_obj`: `brickr` mosaic or 3D model object.

**Value**

Plot object of required bricks by color and size.

**See Also**

Other Resources: `build_colors()`, `build_instructions()`, `build_pieces_table()`

**Examples**

```r
# Import a jpeg or png

demo_file <- system.file("extdata", "demo_img.jpg",
                          package = "brickr", mustWork = TRUE)

demo_image <- jpeg::readJPEG(demo_file)

# Create a mosaic object

mosaic <- demo_image %>%
  image_to_mosaic(img_size = 24)

# Rather than drawing the mosaic, use build_pieces_table() to draw piece chart
```
build_pieces_table

mosaic %>%
  build_pieces()

---

**build_pieces_table**  
*Generate required bricks as a data frame*

**Description**

Create a dataframe of brick colors and sizes used in a brick mosaic or model.

**Usage**

```r
build_pieces_table(brick_obj)
```

**Arguments**

- `brick_obj`: brickr mosaic or 3D model object.

**Value**

Data frame of piece counts by LEGO color name and size.

**See Also**

Other Resources: `build_colors()`, `build_instructions()`, `build_pieces()`

**Examples**

```r
# Import a jpeg or png
demo_file <- system.file("extdata", "demo_img.jpg", 
                          package = "brickr", mustWork = TRUE)
demo_image <- jpeg::readJPEG(demo_file)
#Create a mosaic oject
mosaic <- demo_image %>%
  image_to_mosaic(img_size = 24)

#Rather than drawing the mosaic, use build_pieces_table() to produce piece table
mosaic %>%
  build_pieces_table()
```
**GeomBrick**

**GeomBrick**

**Description**

ggproto for brickr geoms

**image_to_mosaic**

Create a 2D LEGO mosaic from an image array

**Description**

Generate brick mosaics from an image or matrix with customization options.

**Usage**

```r
image_to_mosaic(
  img,
  img_size = 48,
  color_table = NULL,
  method = "cie94",
  color_palette = c("universal", "generic", "special"),
  dithering = FALSE,
  contrast = 1,
  use_bricks = NULL,
  brightness = 1,
  warhol = 1:3
)
```

**Arguments**

- **img**
  Image matrix to convert into mosaic. Usually from `readJPEG` or `readPNG`.

- **img_size**
  Size of output image in pixel, where one pixel = one 'brick'. Use a single value (e.g. 48) for a square image with 48 pixels on each side. Use an array of two values for a rectangular image `c(width, height)`.

- **color_table**
  Defaults to `lego_colors`. Data frame of brick colors to map onto image. Must contain Name and R, G, B channels. See attached data `lego_colors` as examples.

- **method**
  The method to use for comparison. Options are 'euclidean', 'cie1976', 'cie94', 'cie2000', or 'cmc'. See `compare_colour`.

- **color_palette**
  Brick color rarity to use. Defaults to all colors: 'universal' (most common), 'generic', and 'special' (least common). This is useful when trying to build the mosaic out of real bricks. Use "bw" for only grayscale bricks. Ignored if a `color_table` is supplied.
**image_to_mosaic**

- **dithering**  Improves color of large, photo-realistic mosaics.
- **contrast**  For theme = "bw". A value >1 will increase the contrast of the image while a positive value <1 will decrease the contrast.
- **use_bricks**  Array of brick sizes to use in mosaic. Defaults to c('4x2', '2x2', '3x1', '2x1', '1x1').
- **brightness**  A value >1 will increase the brightness of the image while a positive value <1 will decrease the brightness.
- **warhol**  Array of values c(1, 2, 3) associated with R, G, B color channels. Swap values in array to swap color channels for a fun visual effect.

**Value**

A list with element `Img_lego` containing a data frame of the x- & y-coordinates, R, G, B channels, and mapped color of each brick (pixel).

**See Also**

Other Mosaics: `build_mosaic()`

**Examples**

```r
# Import a jpeg or png
demo_file <- system.file("extdata", "demo_img.jpg", 
  package = "brickr", mustWork = TRUE)
demo_image <- jpeg::readJPEG(demo_file)

#Create a 24x24 mosaic

demo_image %>%
  image_to_mosaic(img_size = 24) %>%
  build_mosaic()

#Only use the two more common tiers of colors

demo_image %>%
  image_to_mosaic(img_size = 24,
    color_palette = c("universal", "generic")) %>%
  build_mosaic()

#Be more prescriptive with colors using 'color_table'.
# Here, we prevent all blues from being used
lego_colors_no_blue = lego_colors %>%
  dplyr::filter(!grepl("[Bb]lue|[Aa]zur", Color))
demo_image %>%
  image_to_mosaic(img_size = 24,
    color_table = lego_colors_no_blue) %>%
  build_mosaic()
```
# Color matching is done with the 'farver' package. There are different methods.
# Change the method to euclidean (RGB distance matching)

```r
demo_image %>%
  image_to_mosaic(img_size = 24,
                  method = "euclidean") %>%
  build_mosaic()
```

# Change the default size of bricks to use.
# In this extreme example, use 4x6s for the background, 1x1 everywhere else

```r
demo_image %>%
  image_to_mosaic(img_size = 24,
                  use_bricks = c("4x6")) %>%
  build_mosaic()
```

---

**lego_colors**

*Brickr colors available for mosaics & 3D models*

**Description**

A dataset containing the 54 colors available in 'brickr', along with metadata.

**Usage**

`lego_colors`

**Format**

A data frame with 54 rows and 10 variables:

- **brickrID**: integer, simple color number for use in mosaic creation
- **Color**: color name
- **LEGONo**: integer, color number according to The LEGO Group
- **Palette**: Name of palette, either Universal, Generic, or Special
- **R_lego**: Red channel, (0-1)
- **G_lego**: Green channel, (0-1)
- **B_lego**: Blue channel, (0-1)
- **Trans_lego**: Transparent color, TRUE or FALSE
- **hex**: Color hex code
- **lum**: Color brightness, (0-1)

**Source**

[https://brickarchitect.com/color/](https://brickarchitect.com/color/)
Index

*Topic datasets
  GeomBrick, 16
  lego_colors, 18

brickr, 2
bricks_from_coords, 2, 5, 6, 8, 10
bricks_from_excel, 3, 4, 6, 8, 10
bricks_from_mosaic, 3, 5, 6, 8, 10
bricks_from_table, 3, 5, 6, 7, 10
build_bricks, 3, 5, 6, 8, 10
build_colors, 3, 11, 12, 14, 15
build_instructions, 11, 12, 14, 15
build_mosaic, 13, 17
build_pieces, 11, 12, 14, 15
build_pieces_table, 11, 12, 14, 15
compare_colour, 16

GeomBrick, 16

image_to_mosaic, 13, 16

lego_colors, 18

readJPEG, 16
readPNG, 16

tribble, 8