Package ‘biscale’

May 6, 2020

Type Package
Title Tools and Palettes for Bivariate Thematic Mapping
Version 0.2.0
Description Provides a 'ggplot2' centric approach to bivariate mapping. This is a technique that maps two quantities simultaneously rather than the single value that most thematic maps display. The package provides a suite of tools for calculating breaks using multiple different approaches, a selection of palettes appropriate for bivariate mapping and a scale function for 'ggplot2' calls that adds those palettes to maps. A tool for creating bivariate legends is also included.

Depends R (>= 3.4)
License GPL-3

URL https://github.com/slu-openGIS/biscale

BugReports https://github.com/slu-openGIS/biscale/issues

Encoding UTF-8

LazyData true
Imports classInt, dplyr, ggplot2, glue, rlang, sf, stats, tidyr

RoxygenNote 7.1.0

Suggests covr, knitr, rmarkdown, testthat

VignetteBuilder knitr

NeedsCompilation no

Author Christopher Prener [aut, cre] (https://orcid.org/0000-0002-4310-9888),
    Timo Grossenbacher [aut],
    Angelo Zehr [aut],
    Joshua Stevens [ctb]

Maintainer Christopher Prener <chris.prener@slu.edu>

Repository CRAN

Date/Publication 2020-05-06 05:20:03 UTC
bi_class

Description

Creates mapping classes for a bivariate map. These data will be stored in a new variable named `bi_class`, which will be added to the given data object.

Usage

```r
bi_class(.data, x, y, style = "quantile", dim = 3, keep_factors = FALSE)
```

Arguments

- `.data` A data frame, tibble, or `sf` object
- `x` The x variable
- `y` The y variable
- `style` A string identifying the style used to calculate breaks. Currently supported styles are "quantile" (default), "equal", "fisher", and "jenks".
- `dim` The dimensions of the palette, either 2 for a two-by-two palette or 3 for a three-by-three palette.
- `keep_factors` A logical scalar; if TRUE, the intermediate factor variables created as part of the calculation of `bi_class` will be retained. If FALSE (default), they will not be returned.

Value

A copy of `.data` with a new variable `bi_class` that contains combinations of values that correspond to an observations values for `x` and `y`. This is the basis for applying a bivariate color palette.
Examples

# quantile breaks, 2x2
data <- bi_class(stl_race_income, x = pctWhite, y = medInc, style = "quantile", dim = 2)

# summarize quantile breaks, 2x2
table(data$bi_class)

# quantile breaks, 3x3
data <- bi_class(stl_race_income, x = pctWhite, y = medInc, style = "quantile", dim = 3)

# summarize quantile breaks, 3x3
table(data$bi_class)

# equal breaks
data <- bi_class(stl_race_income, x = pctWhite, y = medInc, style = "equal", dim = 3)

# summarize equal breaks, 3x3
table(data$bi_class)

# fisher breaks
data <- bi_class(stl_race_income, x = pctWhite, y = medInc, style = "fisher", dim = 3)

# summarize fisher breaks, 3x3
table(data$bi_class)

# jenks breaks
data <- bi_class(stl_race_income, x = pctWhite, y = medInc, style = "jenks", dim = 3)

# summarize jenks breaks, 3x3
table(data$bi_class)

bi_legend

Create Object for Drawing Legend

Description

Creates a ggplot object containing a legend that is specific to bivariate mapping.

Usage

bi_legend(pal, dim = 3, xlab, ylab, size)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pal</td>
<td>A palette name; one of &quot;Brown&quot;, &quot;DkBlue&quot;, &quot;DkCyan&quot;, &quot;DkViolet&quot;, or &quot;GrPink&quot;.</td>
</tr>
<tr>
<td>dim</td>
<td>The dimensions of the palette, either 2 for a two-by-two palette or 3 for a three-by-three palette.</td>
</tr>
<tr>
<td>xlab</td>
<td>Text for desired x axis label on legend</td>
</tr>
</tbody>
</table>
y稳妥lab  Text for desired y axis label on legend
size  Size of axis labels

Value
A ggplot object with a bivariate legend.

Examples

# construct 2x2 legend
legend <- bi_legend(pal = "GrPink",
            dim = 2,
            xlab = "Higher % White ",
            ylab = "Higher Income ",
            size = 16)

# print legend
legend

# construct 3x3 legend
legend <- bi_legend(pal = "GrPink",
            dim = 3,
            xlab = "Higher % White ",
            ylab = "Higher Income ",
            size = 16)

# print legend
legend

bi_pal

Palette Preview and Hex Values

Description
Prints either a visual preview of each palette or the associated hex values.

Usage
bi_pal(pal, dim = 3, preview = TRUE)

Arguments

pal  A palette name; one of "Brown", "DkBlue", "DkCyan", "DkViolet", or "GrPink".

dim  The dimensions of the palette, either 2 for a two-by-two palette or 3 for a three-by-three palette.

preview  A logical scalar; if TRUE (default), an image preview will be generated. If FALSE, a vector with hex color values will be returned.
**bi_pal**

**Details**

The "Brown", "DkBlue", "DkCyan", and "GrPink" palettes were made by Joshua Stevens. The "DkViolet" palette was made by Timo Grossenbacher and Angelo Zehr.

**Value**

If `preview = TRUE`, an image preview of the legend will be returned. Otherwise, if `preview = FALSE`, a named vector with class values for names and their corresponding hex color values.

**Examples**

```r
# brown palette, 2x2 preview
bi_pal(pal = "Brown", dim = 2)

# brown palette, 2x2 hex values
bi_pal(pal = "Brown", dim = 2, preview = FALSE)

# brown palette, 3x3 preview
bi_pal(pal = "Brown", dim = 3)

# brown palette, 3x3 hex values
bi_pal(pal = "Brown", dim = 3, preview = FALSE)

# dark blue palette, 2x2 preview
bi_pal(pal = "DkBlue", dim = 2)

# dark blue palette, 2x2 hex values
bi_pal(pal = "DkBlue", dim = 2, preview = FALSE)

# dark blue palette, 3x3 preview
bi_pal(pal = "DkBlue", dim = 3)

# dark blue palette, 3x3 hex values
bi_pal(pal = "DkBlue", dim = 3, preview = FALSE)

# dark cyan palette, 2x2
bi_pal(pal = "DkCyan", dim = 2)

# dark cyan palette, 2x2 hex values
bi_pal(pal = "DkCyan", dim = 2, preview = FALSE)

# dark cyan palette, 3x3
bi_pal(pal = "DkCyan", dim = 3)

# dark cyan palette, 3x3 hex values
bi_pal(pal = "DkCyan", dim = 3, preview = FALSE)

# dark violet palette, 2x2
bi_pal(pal = "DkViolet", dim = 2)

# dark violet palette, 2x2 hex values
bi_pal(pal = "DkViolet", dim = 2, preview = FALSE)
```
# dark violet palette, 3x3
bi_pal(pal = "DkViolet", dim = 3)

# dark violet palette, 3x3 hex values
bi_pal(pal = "DkViolet", dim = 3, preview = FALSE)

# gray pink palette, 2x2
bi_pal(pal = "GrPink", dim = 2)

# gray pink palette, 2x2 hex values
bi_pal(pal = "GrPink", dim = 2, preview = FALSE)

# gray pink palette, 3x3
bi_pal(pal = "GrPink", dim = 3)

# gray pink palette, 3x3 hex values
bi_pal(pal = "GrPink", dim = 3, preview = FALSE)

---

**bi_pal_manual**  
*Create Manual Palette*

**Description**

A function for structuring manual bi-variate palettes. All values must be entered as six-digit hex values (e.g. #000000) and must be preceded by the number symbol. Short forms of hex values (e.g. #000) are not accepted as valid inputs. For two-by-two palettes, only the 'val_1_1', 'val_1_2', 'val_2_1', and 'val_2_2' parameters are required. For three-by-three palettes, all parameters are required.

**Usage**

```r
bi_pal_manual(val_1_1, val_1_2, val_1_3, val_2_1, val_2_2, val_2_3,
              val_3_1, val_3_2, val_3_3, preview = FALSE)
```

**Arguments**

- `val_1_1`  
  A hex value for cell 1-1
- `val_1_2`  
  A hex value for cell 1-2
- `val_1_3`  
  A hex value for cell 1-3
- `val_2_1`  
  A hex value for cell 2-1
- `val_2_2`  
  A hex value for cell 2-2
- `val_2_3`  
  A hex value for cell 2-3
- `val_3_1`  
  A hex value for cell 3-1
- `val_3_2`  
  A hex value for cell 3-2
- `val_3_3`  
  A hex value for cell 3-3
- `preview`  
  A logical scalar; if TRUE (default), an image preview will be generated. If FALSE, a vector with hex color values will be returned.
bi_scale_color

Examples

```r
custom_pal <- bi_pal_manual(val_1_1 = "#E8E8E8", val_1_2 = "#73AE80", val_2_1 = "#6C83B5", val_2_2 = "#2A5A5B")
```

---

### bi_scale_color

**Apply Bivariate Color to ggplot Object**

#### Description

Applies the selected palette as the color aesthetic when `geom_sf` is used and the `bi_class` variable is given as the color in the aesthetic mapping.

#### Usage

```r
bi_scale_color(pal, dim = 3, ...)
```

#### Arguments

- **pal**: Either palette name (one of "Brown", "DkBlue", "DkCyan", "DkViolet", or "GrPink") or a custom palette object created with `bi_pal_manual`.
- **dim**: The dimensions of the palette, either 2 for a two-by-two palette or 3 for a three-by-three palette.
- **...**: Arguments to pass to `scale_fill_manual`.

#### Value

A `ggplot` object with the given bivariate palette applied to the data.

---

### bi_scale_fill

**Apply Bivariate Fill to ggplot Object**

#### Description

Applies the selected palette as the fill aesthetic when `geom_sf` is used and the `bi_class` variable is given as the fill in the aesthetic mapping.

#### Usage

```r
bi_scale_fill(pal, dim = 3, ...)
```
Arguments

pal
Either palette name (one of "Brown", "DkBlue", "DkCyan", "DkViolet", or
"GrPink") or a custom palette object created with bi_pal_manual.

dim
The dimensions of the palette, either 2 for a two-by-two palette or 3 for a three-
by-three palette.

... Arguments to pass to scale_fill_manual

Value

A ggplot object with the given bivariate palette applied to the data.

Examples

# load suggested dependencies
library(ggplot2)
library(sf)

# add breaks, 2x2
data <- bi_class(stl_race_income, x = pctWhite, y = medInc, dim = 2)

# create map
ggplot() +
  geom_sf(data = data, aes(fill = bi_class), color = "white", size = 0.1, show.legend = FALSE) +
  bi_scale_fill(pal = "GrPink", dim = 2)

# add breaks, 3x3
data <- bi_class(stl_race_income, x = pctWhite, y = medInc, dim = 3)

# create map
ggplot() +
  geom_sf(data = data, aes(fill = bi_class), color = "white", size = 0.1, show.legend = FALSE) +
  bi_scale_fill(pal = "GrPink", dim = 3)

bi_theme

Basic Theme for Bivariate Mapping

Description

A theme for creating a simple, clean bivariate map using ggplot2.

Usage

bi_theme(
  base_family = "sans",
  base_size = 24,
  bg_color = "#ffffff",
  font_color = "#000000",
  ...
)
Arguments

- **base_family**: A character string representing the font family to be used in the map.
- **base_size**: A number representing the base size used in the map.
- **bg_color**: A character string containing the hex value for the desired color of the map’s background.
- **font_color**: A character string containing the hex value for the desired color of the map’s text.
- **...**: Arguments to pass on to `ggplot2`’s theme function

Examples

```r
# load suggested dependencies
library(ggplot2)
library(sf)

# add breaks, 3x3
data <- bi_class(stl_race_income, x = pctWhite, y = medInc, dim = 3)

# create map
ggplot() +
  geom_sf(data = data, aes(fill = bi_class), color = "white", size = 0.1, show.legend = FALSE) +
  bi_scale_fill(pal = "GrPink", dim = 3) +
  bi_theme()
```

---

**stl_race_income**  
*Race and Median Income in St. Louis by Census Tract, 2017*

Description

A simple features data set containing the geometry and associated attributes for the 2013-2017 American Community Survey estimates for median household income and the percentage of white residents in St. Louis.

Usage

```r
data(stl_race_income)
```

Format

A data frame with 106 rows and 4 variables:

- **GEOID**: full GEOID string
- **pctWhite**: Percent of white residents per tract
- **medInc**: Median household income of tract
- **geometry**: simple features geometry
**Source**

tidycensus package

**Examples**

```r
str(stl_race_income)
head(stl_race_income)
summary(stl_race_income$medInc)
```
Index

*Topic datasets
  stl_race_income, 9

bi_class, 2
bi_legend, 3
bi_pal, 4
bi_pal_manual, 6, 7, 8
bi_scale_color, 7
bi_scale_fill, 7
bi_theme, 8

geom_sf, 7
ggplot2, 8

scale_fill_manual, 7, 8
stl_race_income, 9