Package: biscale (via r-universe)

September 5, 2024

Type Package

Title Tools and Palettes for Bivariate Thematic Mapping

Version 1.1.0.9000

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 scale functions for 'ggplot2' calls that adds those palettes to maps. Tools for creating bivariate legends are also included.

Depends R (>= 3.5)

License GPL-3

URL https://chris-prener.github.io/biscale/

BugReports https://github.com/chris-prener/biscale/issues

Encoding UTF-8

LazyData true

Imports classInt, ggplot2

RoxygenNote 7.2.0

Suggests covr, cowplot, knitr, rmarkdown, showtext, sf, testthat

VignetteBuilder knitr

Repository https://chris-prener.r-universe.dev

RemoteUrl https://github.com/chris-prener/biscale

RemoteRef HEAD

RemoteSha 4457dd13392a89a66d93050b28f9c7907e78f63f

Contents

`bi_class` *Create Classes for Bivariate Maps*

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

```
bi_class(.data, x, y, style, dim = 3, keep_factors = FALSE, dig_lab = 3)
```

Arguments

`.data`	A data frame, tibble, or `sf` object
`x`	The `x` variable, either a numeric (including double and integer classes) or factor
`y`	The `y` variable, either a numeric (including double and integer classes) or factor
`style`	A string identifying the style used to calculate breaks. Currently supported styles are `"quantile"`, `"equal"`, `"fisher"`, and `"jenks"`. If both `x` and `y` are factors, this argument can be omitted. Note that older versions of `biscale` used `"quantile"` as the default for this argument. Now that `bi_class` accepts factors, this argument no longer as a default and older code will error.
`dim`	The dimensions of the palette. To use the built-in palettes, this value must be either `2`, `3`, or `4`. A value of `3`, for example, would be used to create a three-by-three bivariate map with a total of 9 classes. If you are using a custom palette, this value may be larger (though these maps can be very hard to interpret). If you are using pre-made factors, both factors must have the same number of levels as this value.

`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.
`dig_lab`	An integer that is passed to `base::cut()`

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)
```

`bi_class_breaks` *Return Breaks*

Description

This function can be used to return a list containing vectors of either the ranges of values included in each category of `x` and `y` or, alternatively, the individual break values including the minimum and maximum values. This function supports simplified reporting as well as more descriptive legends.

Usage

```
bi_class_breaks(.data, x, y, style, dim = 3, clean_levels = TRUE,
    dig_lab = 3, si_levels = FALSE, split = FALSE)
```

Arguments

`.data`	A data frame, tibble, or `sf` object
`x`	The `x` variable, either a numeric (including double and integer classes) or factor
`y`	The `y` variable, either a numeric (including double and integer classes) or factor

`style` A string identifying the style used to calculate breaks. Currently supported styles are `"quantile"` (default), `"equal"`, `"fisher"`, and `"jenks"`. If both `x` and `y` are factors, this argument can be omitted.

`dim` The dimensions of the palette. To use the built-in palettes, this value must be either `2`, `3`, or `4`. A value of `3`, for example, would be used to create a three-by-three bivariate map with a total of 9 classes.

If you are using a custom palette, this value may be larger (though these maps can be very hard to interpret).

If you are using pre-made factors, both factors must have the same number of levels as this value.

`clean_levels` A logical scalar; if `TRUE` (default), the brackets and parentheses will be stripped from the output. If `FALSE` (default), the levels will be returned with brackets and parentheses. If `split` is `TRUE` and `clean_levels` is `FALSE`, the `clean_levels` argument will be overridden.

`dig_lab` An integer that is passed to `base::cut()`; it determines the number of digits used in formatting break numbers. It can either be a scalar or a vector. If it is a scalar, the value will be applied to both the `x` and `y` variables. If it is a vector, the first element will be applied to the `x` variable and the second will be applied to the `y` variable.

`si_levels` A logical scalar or vector of length 2 that where `TRUE`, and taking into account `dig_lab` (default = 3), rounds the level(s) and applies one of a few selected SI prefixes, if appropriate. Affects either or both the display of the `x` and `y` variables based on the same syntax as the `dig_lab` parameter. Defaults to `FALSE` (no adjustment to either variable).

`split` A logical scalar; if `FALSE` (default), the range of values for each factor level (corresponds to `dim`) will be returned for both the `x` and `y` variables. If `TRUE`, the individual values for each break (including the minimum and maximum values) will be returned.

Value

A list where `bi_x` is a vector containing the breaks for the `x` variable and `bi_y` is a vector containing the breaks for the `y` variable.

Examples

```
# return ranges for each category of x and y
bi_class_breaks(stl_race_income, style = "quantile", x = pctWhite, y = medInc,
    dim = 4, dig_lab = c(4, 5), split = FALSE)

# ranges can be returned with brackets and parentheses
bi_class_breaks(stl_race_income, style = "quantile", x = pctWhite, y = medInc,
    clean_levels = FALSE, dim = 4, dig_lab = 3, split = FALSE)

# return breaks for each category of x and y
bi_class_breaks(stl_race_income, style = "quantile", x = pctWhite, y = medInc,
    dim = 4, dig_lab = c(4, 5), split = TRUE)
```

```
# show SI prefix
bi_class_breaks(stl_race_income, style = "quantile", x = pctWhite, y = medInc,
    dim = 4, dig_lab = c(4, 5), si_levels = c(y = TRUE), split = TRUE)

# optionally name vector for dig_lab for increased clarity of code
bi_class_breaks(stl_race_income, style = "quantile", x = pctWhite, y = medInc,
    dim = 4, dig_lab = c(x = 4, y = 5), split = TRUE)

# scalars can also be used for dig_lab, though results may be less optimal
bi_class_breaks(stl_race_income, style = "quantile", x = pctWhite, y = medInc,
    dim = 4, dig_lab = 3, split = TRUE)
```

`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 = 10, flip_axes = FALSE,
    rotate_pal = FALSE, pad_width = NA, pad_color = "#ffffff",
    breaks = NULL, arrows = TRUE, base_family = "sans")
```

Arguments

`pal`	A palette name or a vector containing a custom palette. See the help file for `bi_pal` for complete list of built-in palette names. If you are providing a custom palette, it must follow the formatting described in the 'Advanced Options' vignette.
`dim`	The dimensions of the palette. To use the built-in palettes, this value must be either `2`, `3`, or `4`. A value of `3`, for example, would be used to create a three-by-three bivariate map with a total of 9 classes. If you are using a custom palette, this value may be larger (though these maps can be very hard to interpret). See the 'Advanced Options' vignette for details on the relationship between `dim` values and palette size.
`xlab`	Text for desired x axis label on legend
`ylab`	Text for desired y axis label on legend
`size`	A numeric scalar; size of axis labels
`flip_axes`	A logical scalar; if `TRUE`, the axes of the palette will be flipped. If `FALSE` (default), the palette will be displayed on its original axes. Custom palettes with 'dim' greater than 4 cannot take advantage of flipping axes.

rotate_pal	A logical scalar; if TRUE, the palette will be rotated 180 degrees. If FALSE (default), the palette will be displayed in its original orientation. Custom palettes with 'dim' greater than 4 cannot take advantage of palette rotation.
pad_width	An optional numeric scalar; controls the width of padding between tiles in the legend
pad_color	An optional character scalar; controls the color of padding between tiles in the legend
breaks	An optional list created by bi_class_breaks. Depending on the options selected when making the list, labels will placed showing the corresponding range of values for each axis or, if split = TRUE, showing the individual breaks.
arrows	A logical scalar; if TRUE (default), directional arrows will be added to both the x and y axes of the legend. If you want to suppress these arrows, especially if you are supplying breaks to create a more detailed legend, this parameter can be set of FALSE.
base_family	A character string; by default, it is set to "sans", which has been the font used in biscale since its initial release. If you are using non-Latin characters, you may need to set base_family = "" to get your characters to display. Other options include "mono" and "serif". See the Breaks and Legends vignette for details.

Value

A ggplot object with a bivariate legend.

See Also

bi_pal

Examples

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

## print legend
legend

# sample 3x3 legend with breaks
## create vector of breaks
break_vals <- bi_class_breaks(stl_race_income, style = "quantile",
    x = pctWhite, y = medInc, dim = 3, dig_lab = c(x = 4, y = 5),
    split = TRUE)

## create legend
```

```
legend <- bi_legend(pal = "GrPink",
                    dim = 3,
                    xlab = "Higher % White ",
                    ylab = "Higher Income ",
                    size = 16,
                    breaks = break_vals,
                    arrows = FALSE)

## print legend
legend

# sample 3x3 legend with Chinese characters
## set language preference
showtext::showtext_auto()

## create legend
legend <- bi_legend(pal = "GrPink",
                    dim = 3,
                    xlab = "  ",
                    ylab = "  ",
                    size = 16,
                    base_family = "")

## print legend
legend
```

`bi_pal` *Preview Palettes 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, flip_axes = FALSE, rotate_pal = FALSE)
```

Arguments

`pal`	A palette name or a vector containing a custom palette. If you are providing a palette name, it must be one of: `"Bluegill"`, `"BlueGold"`, `"BlueOr"`, `"BlueYl"`, `"Brown"`/`"Brown2"`, `"DkBlue"`/`"DkBlue2"`, `"DkCyan"`/`"DkCyan2"`, `"DkViolet"`/`"DkViolet2"`, `"GrPink"`/`"GrPink2"`, `"PinkGrn"`, `"PurpleGrn"`, or `"PurpleOr"`. Pairs of palettes, such as `"GrPink"`/`"GrPink2"`, are included for legacy support. The numbered palettes support four-by-four bivarite maps while the un-numbered ones, which were the five included in the original release of the package, only support two-by-two and three-by-three maps.

If you are providing a custom palette, it must follow the formatting described in the 'Advanced Options' vignette.

`dim` The dimensions of the palette. To use the built-in palettes, this value must be either `2`, `3`, or `4`. A value of `3`, for example, would be used to create a three-by-three bivariate map with a total of 9 classes.

If you are using a custom palette, this value may be larger (though these maps can be very hard to interpret). See the 'Advanced Options' vignette for details on the relationship between `dim` values and palette size.

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

`flip_axes` A logical scalar; if `TRUE` the axes of the palette will be flipped. If `FALSE` (default), the palette will be displayed on its original axes. Custom palettes with 'dim' greater than 4 cannot take advantage of flipping axes.

`rotate_pal` A logical scalar; if `TRUE` the palette will be rotated 180 degrees. If `FALSE` (default), the palette will be displayed in its original orientation. Custom palettes with 'dim' greater than 4 cannot take advantage of palette rotation.

Details

The `"Brown"`, `"DkBlue"`, `"DkCyan"`, and `"GrPink"` palettes were made by Joshua Stevens. The `"DkViolet"` palette was made by Timo Grossenbacher and Angelo Zehr. Many of the new palettes were inspired by Branson Fox's earlier work to expand `biscale`.

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

```
# 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)

# custom palette
custom_pal <- c(
    "1-1" = "#cabed0", # low x, low y
    "2-1" = "#ae3a4e", # high x, low y
    "1-2" = "#4885c1", # low x, high y
    "2-2" = "#3f2949" # high x, high y
```

```
)

bi_pal(pal = custom_pal, dim = 2, preview = FALSE)
```

`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

```
bi_scale_color(pal, dim = 3, flip_axes = FALSE, rotate_pal = FALSE, ...)
```

Arguments

`pal`	A palette name or a vector containing a custom palette. See the help file for `bi_pal` for complete list of built-in palette names. If you are providing a custom palette, it must follow the formatting described in the 'Advanced Options' vignette.
`dim`	The dimensions of the palette. To use the built-in palettes, this value must be either `2`, `3`, or `4`. A value of `3`, for example, would be used to create a three-by-three bivariate map with a total of 9 classes. If you are using a custom palette, this value may be larger (though these maps can be very hard to interpret). See the 'Advanced Options' vignette for details on the relationship between `dim` values and palette size.
`flip_axes`	A logical scalar; if `TRUE` the axes of the palette will be flipped. If `FALSE` (default), the palette will be displayed on its original axes. Custom palettes with 'dim' greater than 4 cannot take advantage of flipping axes.
`rotate_pal`	A logical scalar; if `TRUE` the palette will be rotated 180 degrees. If `FALSE` (default), the palette will be displayed in its original orientation. Custom palettes with 'dim' greater than 4 cannot take advantage of palette rotation.
`...`	Arguments to pass to `scale_color_manual`

Value

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

See Also

bi_pal

Examples

```
# load dependencies
library(ggplot2)

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

# create map
plot <- ggplot() +
  geom_sf(data = data, aes(color = bi_class), size = 2, show.legend = FALSE) +
  bi_scale_color(pal = "GrPink", dim = 3)
```

---

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

```
bi_scale_fill(pal, dim = 3, flip_axes = FALSE, rotate_pal = FALSE, ...)
```

Arguments

pal	A palette name or a vector containing a custom palette. See the help file for bi_pal for complete list of built-in palette names. If you are providing a custom palette, it must follow the formatting described in the 'Advanced Options' vignette.
dim	The dimensions of the palette, either 2 for a two-by-two palette, 3 for a three-by-three palette, or 4 for a four-by-four palette.
flip_axes	A logical scalar; if TRUE the axes of the palette will be flipped. If FALSE (default), the palette will be displayed on its original axes.
rotate_pal	A logical scalar; if TRUE the palette will be rotated 180 degrees. If FALSE (default), the palette will be displayed in its original orientation
...	Arguments to pass to scale_fill_manual

Value

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

See Also

bi_pal

Examples

```
# load dependencies
library(ggplot2)

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

# create map
plot <- 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

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

# add breaks, 3x3
```

```
data <- bi_class(stl_race_income, x = pctWhite, y = medInc, style = "quantile", 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. This version of the sample data are stored as polygon data.

Usage

```
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

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

`stl_race_income_point` *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. This version of the sample data are stored as point data.

Usage

```
data(stl_race_income_point)
```

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

```
str(stl_race_income_point)
head(stl_race_income_point)
summary(stl_race_income_point$medInc)
```

Index