Package 'choroplethr'

June 26, 2025

Title Create Color-Coded Choropleth Maps in R

Description Easily create color-coded (choropleth) maps in R. No knowledge of cartography or shapefiles needed; go directly from your geographically identified data to a highly customizable map with a single line of code! Supported geographies: U.S. states, counties, and census tracts, world countries and sub-country regions (e.g., provinces, prefectures, etc.). One of the suggested packages, rnaturalearthhires, is not available on CRAN owing to its larger filesize (40MB). It can be installed from GitHub using remotes::install_github(``https://github.com/ropensci/rnaturalearthhires''). This package contains higher resolution sub-country maps and is only needed for the choropleth_admin1() function.

Version 5.0.0

Maintainer Zhaochen He <zhaochen.he@cnu.edu>

URL https://github.com/eastnile/choroplethr

License BSD_3_clause + file LICENSE

Imports Hmisc, stringr, ggplot2 (>= 2.0.0), dplyr, R6, ggrepel, tigris (>= 1.0), sf, tidycensus, rnaturalearth

Suggests testthat (>= 3.0.0), rnaturalearthhires

Depends R (>= 3.5.0)

Collate 'acs.R' 'admin1.R' 'choropleth.R' 'country.R' 'county.R' 'data.R' 'get_usa_demographics.R' 'init.R' 'internal-docs.R' 'state.R' 'tract.R'

RoxygenNote 7.3.2

Encoding UTF-8

LazyData true

LazyDataCompression xz

Config/testthat/edition 3

NeedsCompilation no

Author Ari Lamstein [aut],
Zhaochen He [ctb, cre],
Brian Johnson [ctb],
Trulia, Inc. [cph]

Repository CRAN

Date/Publication 2025-06-26 21:30:06 UTC

Contents

`admin1_choropleth` *Create a choropleth map using regional data at the sub-country level*

Description

This function can be used to plot regional data at the first sub-level of administration (ie., state, province, prefecture, etc.) for one or more countries. Use get_admin1_map() for an object which can help you coerce your region names into the required format; see below for an example with Japanese data.

Usage

```
admin1_choropleth(
  df,
  geoid.name = "region",
  geoid.type = "auto",
  value.name = "value",
  num_colors = 7,
  color.max = NULL,
  color.min = NULL,
  na.color = "grey",
  custom.colors = NULL,
  nbreaks = 5,
  zoom = NULL,
  country_zoom = NULL,
  projection = "cartesian",
  limits_lat = NULL,
  limits_lon = NULL,
  reproject = TRUE,
  whitespace = TRUE,
  border_color = "grey15",
  border_thickness = 0.2,
  background_color = "white",
  gridlines = FALSE,
  latlon_ticks = FALSE,
  label = NULL,
  label_text_size = 3,
  label_text_color = "black",
  label_box_color = "white",
  ggrepel_options = NULL,
  legend = NULL,
```

```
  legend_position = "right",
  title = NULL,
  return = "plot"
)
```

Arguments

df	A dataframe containing regional data at the sub-country level for one or more countries.
geoid.name	The variable that identifies each administrative region
geoid.type	How the variable given by geoid.name specifies each country. The allowed geoid.type are given by the columns "adm1_code", "diss_me", "ne_id" in the output of get_admin1_map(); use this output to match the names of your regions to the correct geoid. If "auto", the function will try to automatically determine geoid.type.
value.name	The name of the variable you wish to plot.
num_colors	The number of colors you want in your graph when plotting continuous data. If num_colors > 1, the variable in question will be divided into quantiles and converted into a factor with that many levels. If num_colors = 1, a continuous color gradient will be used; if num_colors = 0, a diverging color gradient will be used (useful for visualizing negative and positive numbers). Use color.max and color.min to control the range of colors displayed. num_colors is ignored when plotting categorical data.
color.max	The color of the highest value in your data. Ignored if the plotted variable is categorical.
color.min	The color of the lowest value in your data. Ignored if the plotted variable is categorical.
na.color	The color you want to assign for regions with missing data
custom.colors	A vector of valid R color terms of the to use for the map when plotting factor variables. The length of this vector must match the number of levels in your factor variable, or num_colors for a continuous variable that will be discretized by the function, and the order should match the order of the levels of in your factor variable.
nbreaks	The number of breaks you wish to show in the legend when using a continuous color scale. Ignored if num_colors > 1.
zoom	An optional vector of regions to zoom in on, written in the same manner as geoid.name.
country_zoom	An optional vector of countries to zoom in on, written as they appear in the "adm0_a3" column of the object returned from get_tract_map().
projection	One of the following: "cartesian", "mercator", "robinson", or "albers", for equirectangular, Mercator, Robinson, and Albers Equal Area projections, respectively. When using the Mercator projection for world maps, setting limits_lon is recommended to prevent exaggeration of the size of Antarctica.
limits_lat	A length two vector giving the minimum and maximum latitude you wish to include in your map.

limits_lon
: A length two vector giving the minimum and maximum longitude you wish to include in your map.

reproject
: If TRUE, the map will be cropped and centered prior to applying the projection. This will generally result in a better figure when using the Robinson and Albers, but may lead to countries near the edge of the map being occluded.

whitespace
: Add some blank space to the sides of your map? For some projections, this must be set to FALSE in order for lat/lon ticks and display correctly.

border_color
: The color of the borders on your map

border_thickness
: The thickness of the borders on your map

background_color
: The background color of your map

gridlines
: Should gridlines appear on your map?

latlon_ticks
: Should lat/lon tick marks appear on the edge of your map?

label
: The name of variable you wish to use to label your map; must be one of the variables that appears in the spatial dataframe just prior plotting (use return = 'sf' to see this dataframe), and in general, can be any of the allowed geoid.type. This function uses ggplot2::geom_label_repel to create the labels and ensure that they do not overlap.

label_text_size
: The size of the text that will appear in each label

label_text_color
: The color of the text that will appear in each label

label_box_color
: The color of the box around each label

ggrepel_options
: A list containing additional arguments to be passed to geom_label_repel (see ?ggplot2::geom_label_repel)

legend
: A title for your legend; if NULL, value.name will be used.

legend_position
: The position of your legend relative to the rest of the map; can be "top", "bottom", "left", or "right".

title
: A title for your plot; if NULL, no title will be added.

return
: If "plot", the function will return the requested map as a ggplot object. If "sf", the function will return the spatial dataframe used to draw the map (useful if you wish to customize the map yourself).

Details

Note: This function requires the package rnatrualearthhires, which is not available on CRAN due to the filesize of the map being large. You can install it using: remotes::install_github("ropensci/rnaturalearthhires")

Examples

```
library(dplyr)
# Our Japanese data is at the prefecture level, with names in English lower case.
df_japan_census = choroplethr::df_japan_census
# We match our data to one of the geoids ("adm1_code", "diss_me", or "ne_id" )
# in the output of get_admin1_map().

if (requireNamespace("rnaturalearthhires")) {
  admin1_lookup = get_admin1_map()
  # The "name_en" variable is very close to how the prefectures are named in our data.
 admin1_lookup = admin1_lookup[admin1_lookup$admin == 'Japan', c('adm1_code', 'name_en')]
  admin1_lookup$name_lower = tolower(admin1_lookup$name_en)
  admin1_lookup$name_lower = iconv(admin1_lookup$name_lower,
                            from = "UTF-8", to = "ASCII//TRANSLIT") # Remove accent marks
  admin1_lookup$name_lower = gsub(pattern = ' prefecture', replacement = '',
                                  x = admin1_lookup$name_lower)
  # We merge in admin1_code after making name_en resemble our data.
 data_prepped = left_join(df_japan_census, admin1_lookup[, c('adm1_code', 'name_lower')],
                          by = join_by(region == name_lower))
  admin1_choropleth(data_prepped, geoid.name = 'adm1_code', value.name = 'pop_2010',
                    country_zoom = 'JPN', num_colors = 4) # Create the map
}
```

`Choropleth` *The base Choropleth object.*

Description

The base Choropleth object.

The base Choropleth object.

Methods

Public methods:

- `Choropleth$new()`
- `Choropleth$set_zoom()`
- `Choropleth$get_ggscale()`
- `Choropleth$get_projection()`
- `Choropleth$render()`
- `Choropleth$clone()`

Method `new()`:

Usage:

```
Choropleth$new(
  ref.regions,
  ref.regions.name,
  map.df,
  geoid.all,
  user.df,
  geoid.name,
  geoid.type,
  value.name,
  num_colors,
  label_col
)
```

Method `set_zoom()`**:**

Usage:

```
Choropleth$set_zoom(zoom)
```

Method `get_ggscale()`**:**

Usage:

```
Choropleth$get_ggscale(
  choropleth.df = self$choropleth.df,
  respect_zoom = TRUE,
  custom.colors,
  color.min,
  color.max,
  na.color,
  nbreaks
)
```

Method `get_projection()`**:**

Usage:

```
Choropleth$get_projection(
  choropleth.df = self$choropleth.df,
  respect_zoom = TRUE,
  projection_name,
  ignore_latlon,
  limits_lat,
  limits_lon,
  reproject,
  whitespace
)
```

Method `render()`**:**

Usage:

```
Choropleth$render(
  choropleth.df = self$choropleth.df,
  ggscale,
  projection,
```

```
  respect_zoom = TRUE,
  occlude_latlon_limits,
  border_color,
  border_thickness,
  background_color,
  gridlines,
  latlon_ticks,
  label,
  label_text_size,
  label_text_color,
  label_box_color,
  ggrepel_options,
  legend,
  legend_position,
  title,
  addl_gglayer
)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Choropleth$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`congress116.regions` *A data.frame containing geographic metadata about the Congressional Districts of the 116th US Congress*

Description

Column region is how the Census Bureau refers to the geography. Note that this region is a 4-character string, and so has a leading 0 if necessary. The first two characters are the state FIPS code, and the second two characters are the district ID. States that only have 1 district (i.e. a representative "at large") have district 00. All other states start at 01.

Usage

```
data(congress116.regions)
```

continental_us_states *A vector of the names of US Continental US States.*

Description

A vector of the names of US Continental US States.

Usage

```
data(continental_us_states)
```

Author(s)

Ari Lamstein

country.map *An sf containing geometry data for countries of the world*

Description

An sf containing geometry data for countries of the world

Usage

```
data(country.map)
```

References

Data obtained using the ne_countries function from rnaturalearth; https://github.com/ropensci/rnaturalearth, https://www.naturalearthdata.com/

country.regions *Supported regions for world countries*

Description

Supported regions for world countries

Usage

```
data(country.regions)
```

`country_choropleth` *Create a choropleth map using country-level data*

Description

See choroplethr::country.regions for an object which can help you coerce your country names into the required format; the allowed geoid for this function are columns name.proper, name.lower, iso_a3, and iso_a2 which appear at the beginning of this object.

Usage

```
country_choropleth(
  df,
  geoid.name = "region",
  geoid.type = "auto",
  value.name = "value",
  num_colors = 7,
  color.max = NULL,
  color.min = NULL,
  na.color = "grey",
  custom.colors = NULL,
  nbreaks = 5,
  zoom = NULL,
  continent_zoom = NULL,
  projection = "cartesian",
  limits_lat = NULL,
  limits_lon = NULL,
  reproject = TRUE,
  border_color = "grey15",
  border_thickness = 0.2,
  background_color = "white",
  gridlines = FALSE,
  latlon_ticks = FALSE,
  whitespace = TRUE,
  label = NULL,
  label_text_size = 3,
  label_text_color = "black",
  label_box_color = "white",
  ggrepel_options = NULL,
  legend = NULL,
  legend_position = "right",
  title = NULL,
  return = "plot"
)
```

Arguments

`df`	A dataframe containing country level data

`geoid.name` The variable that identifies each country

`geoid.type` How the variable given by geoid.name specifies each country. The allowed geoid.type are given by the columns name.proper, name.lower, iso_a3, and iso_a2 in choroplethr::country.regions. If "auto", the function will try to automatically determine geoid.type.

`value.name` The name of the variable you wish to plot.

`num_colors` The number of colors you want in your graph when plotting continuous data. If num_colors > 1, the variable in question will be divided into quantiles and converted into a factor with that many levels. If num_colors = 1, a continuous color gradient will be used; if num_colors = 0, a diverging color gradient will be used (useful for visualizing negative and positive numbers). Use color.max and color.min to control the range of colors displayed. num_colors is ignored when plotting categorical data.

`color.max` The color of the highest value in your data. Ignored if the plotted variable is categorical.

`color.min` The color of the lowest value in your data. Ignored if the plotted variable is categorical.

`na.color` The color you want to assign for regions with missing data

`custom.colors` A vector of valid R color terms of the to use for the map when plotting factor variables. The length of this vector must match the number of levels in your factor variable, or num_colors for a continuous variable that will be discretized by the function, and the order should match the order of the levels of in your factor variable.

`nbreaks` The number of breaks you wish to show in the legend when using a continuous color scale. Ignored if num_colors > 1.

`zoom` An optional vector of countries to zoom in on, written in the same manner as geoid.name.

`continent_zoom` Zoom in on a particular continent; to see which countries belong to which continent, see choroplethr::country.regions

`projection` One of the following: "cartesian", "mercator", "robinson", or "albers", for equirectangular, Mercator, Robinson, and Albers Equal Area projections, respectively. When using the Mercator projection for world maps, setting limits_lon is recommended to prevent exaggeration of the size of Antarctica.

`limits_lat` A length two vector giving the minimum and maximum latitude you wish to include in your map.

`limits_lon` A length two vector giving the minimum and maximum longitude you wish to include in your map.

`reproject` If TRUE, the map will be cropped and centered prior to applying the projection. This will generally result in a better figure when using the Robinson and Albers, but may lead to countries near the edge of the map being occluded.

`border_color` The color of the borders on your map

`border_thickness` The thickness of the borders on your map

background_color
: The background color of your map

gridlines
: Should gridlines appear on your map?

latlon_ticks
: Should lat/lon tick marks appear on the edge of your map?

whitespace
: Add some blank space to the sides of your map? For some projections, this must be set to FALSE in order for lat/lon ticks and display correctly.

label
: The name of variable you wish to use to label your map; must be one of the variables that appears in the spatial dataframe just prior plotting (use return = 'sf' to see this dataframe), and in general, can be any of the allowed geoid.type. This function uses ggplot2::geom_label_repel to create the labels and ensure that they do not overlap.

label_text_size
: The size of the text that will appear in each label

label_text_color
: The color of the text that will appear in each label

label_box_color
: The color of the box around each label

ggrepel_options
: A list containing additional arguments to be passed to geom_label_repel (see ?ggplot2::geom_label_repel)

legend
: A title for your legend; if NULL, value.name will be used.

legend_position
: The position of your legend relative to the rest of the map; can be "top", "bottom", "left", or "right".

title
: A title for your plot; if NULL, no title will be added.

return
: If "plot", the function will return the requested map as a ggplot object. If "sf", the function will return the spatial dataframe used to draw the map (useful if you wish to customize the map yourself).

Examples

```
# Create a choropleth map using country level data:
data(df_country_demographics)
country_choropleth(df_country_demographics, geoid.name = 'region', geoid.type = 'iso_a3',
                   value.name = 'gdp',
                   title = "GDP of Countries in the World", legend = 'GDP (millions)')

# Use a divergent continuous color scale and customize map appearance:
country_choropleth(df_country_demographics, geoid.name = 'region', geoid.type = 'iso_a3',
                   value.name = 'gdp', num_colors = 0, border_color = 'grey',
                   color.max = 'gold', color.min = 'navyblue',
                   projection = 'robinson', latlon_ticks = TRUE,
                   gridlines = TRUE, whitespace = FALSE,
                   background_color = 'azure',
                   title = "GDP of Countries in the World", legend = 'GDP (millions)')

# Zoom in on South America:
```

```
country_choropleth(df_country_demographics, geoid.name = 'region', geoid.type = 'iso_a3',
                   value.name = 'gdp', num_colors = 0, border_color = 'grey',
                   continent_zoom = 'South America',
                   color.max = 'gold', color.min = 'navyblue',
                   projection = 'robinson', latlon_ticks = TRUE,
                   gridlines = TRUE, whitespace = FALSE,
                   background_color = 'azure',
                   title = "GDP of Countries in the World", legend = 'GDP (millions)',
                   label = 'iso_a2', label_text_size = 5)
```

`county.map.2015` *An sf containing geometry data for US counties in 2015*

Description

An sf containing geometry data for US counties in 2015

Usage

```
data(county.map.2015)
```

References

obtained using tigris::counties()

`county.map.2024` *An sf containing geometry data for US counties in 2024*

Description

An sf containing geometry data for US counties in 2024

Usage

```
data(county.map.2024)
```

References

obtained using tigris::counties()

county.regions.2015 *Supported regions for US counties in 2015*

Description

Supported regions for US counties in 2015

Usage

```
data(county.regions.2015)
```

county.regions.2024 *Supported regions for US counties in 2024*

Description

Supported regions for US counties in 2024

Usage

```
data(county.regions.2024)
```

county_choropleth *Create a choropleth map using U.S. county level data:*

Description

Counties must be identified by FIPS code; see choroplethr::county.regions.2015 or choroplethr::county.regions.2024 for an object that can help you coerce your county names into this format.

Usage

```
county_choropleth(
  df,
  map_year = 2024,
  geoid.name = "region",
  geoid.type = "auto",
  value.name = "value",
  num_colors = 7,
  color.max = NULL,
  color.min = NULL,
  na.color = "grey",
  custom.colors = NULL,
  nbreaks = 5,
```

```
  county_zoom = NULL,
  state_zoom = NULL,
  projection = "albers",
  border_color = "grey15",
  border_thickness = 0.2,
  background_color = "white",
  gridlines = FALSE,
  latlon_ticks = FALSE,
  whitespace = TRUE,
  label = NULL,
  label_text_size = 2.25,
  label_text_color = "black",
  label_box_color = "white",
  ggrepel_options = NULL,
  legend = NULL,
  legend_position = "right",
  title = NULL,
  return = "plot",
  add_state_outline = TRUE
)
```

Arguments

df	A dataframe containing U.S. county level data
map_year	Either 2015 or 2024; uses county definitions from that particular year.
geoid.name	The name of the variable that identifies each county
geoid.type	Either "fips.numeric" or "fips.character"; if "auto", the function will try to automatically determine geoid.type. See choroplethr::county.regions.2015 or choroplethr::county.regions.2024 a lookup table.
value.name	The name of the variable you wish to plot.
num_colors	The number of colors you want in your graph when plotting continuous data. If num_colors > 1, the variable in question will be divided into quantiles and converted into a factor with that many levels. If num_colors = 1, a continuous color gradient will be used; if num_colors = 0, a diverging color gradient will be used (useful for visualizing negative and positive numbers). Use color.max and color.min to control the range of colors displayed. num_colors is ignored when plotting categorical data.
color.max	The color of the highest value in your data. Ignored if the plotted variable is categorical.
color.min	The color of the lowest value in your data. Ignored if the plotted variable is categorical.
na.color	The color you want to assign for regions with missing data
custom.colors	A vector of valid R color terms of the to use for the map when plotting factor variables. The length of this vector must match the number of levels in your factor variable, or num_colors for a continuous variable that will be discretized by the function, and the order should match the order of the levels of in your factor variable.

nbreaks The number of breaks you wish to show in the legend when using a continuous color scale. Ignored if num_colors > 1.

county_zoom An optional vector of counties to zoom in on, written in the same manner as geoid.name.

state_zoom An optional vector of states to zoom in on. Elements of this vector must match one of the columns in choroplethr::state.regions.

projection One of the following: "cartesian", "mercator", "robinson", or "albers", for equirectangular, Mercator, Robinson, and Albers Equal Area projections, respectively. When using the Mercator projection for world maps, setting limits_lon is recommended to prevent exaggeration of the size of Antarctica.

border_color The color of the borders on your map

border_thickness
The thickness of the borders on your map

background_color
The background color of your map

gridlines Should gridlines appear on your map?

latlon_ticks Should lat/lon tick marks appear on the edge of your map?

whitespace Add some blank space to the sides of your map? For some projections, this must be set to FALSE in order for lat/lon ticks and display correctly.

label The name of variable you wish to use to label your map; must be one of the variables that appears in the spatial dataframe just prior plotting (use return = 'sf' to see this dataframe), and in general, can be any of the allowed geoid.type. This function uses ggplot2::geom_label_repel to create the labels and ensure that they do not overlap.

label_text_size
The size of the text that will appear in each label

label_text_color
The color of the text that will appear in each label

label_box_color
The color of the box around each label

ggrepel_options
A list containing additional arguments to be passed to geom_label_repel (see ?ggplot2::geom_label_repel)

legend A title for your legend; if NULL, value.name will be used.

legend_position
The position of your legend relative to the rest of the map; can be "top", "bottom", "left", or "right".

title A title for your plot; if NULL, no title will be added.

return If "plot", the function will return the requested map as a ggplot object. If "sf", the function will return the spatial dataframe used to draw the map (useful if you wish to customize the map yourself).

add_state_outline
Should state borders be outlined in your map?

Examples

```
# Create a map based on US county data:
data("df_county_demographics")
county_choropleth(df_county_demographics, geoid.name = 'region', geoid.type = 'fips.numeric',
                  value.name = 'median_hh_income',
                  title = "Median Household Income of U.S. Counties",
                  legend = 'Median HH Income')

county_choropleth(df_county_demographics, geoid.name = 'region', geoid.type = 'fips.numeric',
                  value.name = 'median_hh_income',
                  state_zoom = c('CA', 'OR', 'WA'),
                  title = "Median Household Income of West Coast Counties",
                  legend = 'Median HH Income')
```

---

county_choropleth_acs *Create a US County choropleth from ACS data*

Description

Creates a choropleth of US counties using the US Census' American Community Survey (ACS) data.

Usage

```
county_choropleth_acs(
  variable = NULL,
  tableId = NULL,
  column_idx = NULL,
  endyear,
  span = 5,
  title = NULL,
  census_api_key = NULL,
  ...
)
```

Arguments

variable	The variable you wish to plot. A list of available census variables can be obtained using tidycensus::load_variables()
tableId	Alternatively, you may specify the ACS table you wish to plot. If the table has more than one variable inside it, you must also specify the index of the column you wish to plot.
column_idx	The index of the desired column within the table.
endyear	The end year of the survey to use.
span	Either 1, 3, or 5, the ACS vintage you wish to use.

title	A title for the plot; if not specified, a title will be assigned based on the variable.
census_api_key	Optional. Census API keys can be obtained at: https://api.census.gov/data/key_signup.html
...	Other arguments passed to county_choropleth; see ?county_choropleth()

Value

A choropleth.

Examples

```
# Median household income, zooming in on all counties in New York, New Jersey and Connecticut
county_choropleth_acs(variable = "B19013_001", num_colors=1, endyear = 2011,
state_zoom=c("new york", "new jersey", "connecticut"))
```

---

df_congress116_demographics

A data.frame containing demographic statistics about the 116th Congressional Districts

---

Description

A data.frame containing demographic statistics about the 116th Congressional Districts

Usage

```
data(df_congress116_demographics)
```

References

Data comes from the 2018 5-year American Community Survey (ACS). Data generated by ?get_congressional_district_demog

---

df_congress116_party

A data.frame containing party affiliation data about the Congressional Districts of 116th US Congress

---

Description

Contains the party affiliation of each member elected to the House of Representatives of the 116th Congress, along with metadata. Note that party affiliation is of who the citizens voted for, and not who is currently (July 30, 2020) serving. Currently three members have resigned since being elected, one switched party and one died. For details of how this data was compiled, please see function get_congressional_116_party_data in file get_congress_116_party_data. That file ships with this package, but is not exported, since it relies on scraping data from Wikipedia, and that web page is subject to change.

Usage

```
data(df_congress116_party)
```

---

df_country_demographics

A data.frame containing population estimates for Countries in 2012.

Description

A data.frame containing population estimates for Countries in 2012.

Usage

```
data(df_country_demographics)
```

References

Data obtained using the ne_countries function from rnaturalearth; https://github.com/ropensci/rnaturalearth, https://www.naturalearthdata.com/

---

df_county_demographics

A data.frame containing demographic statistics for each county in the United States.

Description

A data.frame containing demographic statistics for each county in the United States.

Usage

```
data(df_county_demographics)
```

References

Data comes from the 2013 5-year American Community Survey (ACS). Data generated by ?get_county_demographics.

`df_japan_census` *A data.frame containing basic demographic information about Japan.*

Description

A data.frame containing basic demographic information about Japan.

Usage

```
data(df_japan_census)
```

References

Taken from the "Total Population" table from the Statistics Bureau of Japan website (https://www.stat.go.jp/english/data/nenkan/1431-02.html) on 12/1/2014.

`df_ny_tract_demographics` *A data.frame containing demographic statistics for each Census Tract in New York State.*

Description

A data.frame containing demographic statistics for each Census Tract in New York State.

Usage

```
data(df_ny_tract_demographics)
```

References

Data comes from the 2013 5-year American Community Survey (ACS). Data generated by ?get_tract_demographics.

`df_pop_country` *A data.frame containing population estimates for Countries in 2012.*

Description

A data.frame containing population estimates for Countries in 2012.

Usage

```
data(df_pop_country)
```

References

Taken from the WDI package with code SP.POP.TOTL for year 2012.

df_pop_county *A data.frame containing population estimates for US Counties in 2012.*

Description

A data.frame containing population estimates for US Counties in 2012.

Usage

```
data(df_pop_county)
```

References

Taken from the US American Community Survey (ACS) 5 year estimates.

df_pop_ny_tract *A data.frame containing population estimates for all Census Tracts in New York State in 2012.*

Description

A data.frame containing population estimates for all Census Tracts in New York State in 2012.

Usage

```
data(df_pop_ny_tract)
```

References

Taken from the US American Community Survey (ACS) 5 year estimates.

df_pop_state *A data.frame containing population estimates for US States in 2012.*

Description

A data.frame containing population estimates for US States in 2012.

Usage

```
data(df_pop_state)
```

References

Taken from the US American Community Survey (ACS) 5 year estimates.

`df_president` *A data.frame containing election results from the 2012 US Presidential election.*

Description

A data.frame containing election results from the 2012 US Presidential election.

Usage

```
data(df_president)
```

Author(s)

Ari Lamstein and Richard Careaga

References

Taken from the FEC website on 11/21/2014.

`df_president_ts` *A data.frame containing all US presidential election results from 1789 to 2012*

Description

Legend:

- R = Republican
- D = Democratic
- DR = Democratic-Republican
- W = Whig
- F = Federalist
- GW = George Washington
- NR = National Republican
- SD = Southern Democrat
- PR = Progressive
- AI = American Independent
- SR = States' Rights
- PO = Populist
- CU = Constitutional Union
- I = Independent

- ND = Northern Democrat
- KN = Know Nothing
- AM = Anti-Masonic
- N = Nullifier
- SP = Split evenly

Usage

```
data(df_president_ts)
```

References

Taken from https://en.wikipedia.org/wiki/List_of_United_States_presidential_election_results_by_state 3/20/2014.

---

df_state_age_2010 *A data.frame containing median age estimates for US states in 2010*

Description

A data.frame containing median age estimates for US states in 2010

Usage

```
data(df_state_age_2010)
```

References

Taken from the US American Community Survey (ACS) 5 year estimates.

---

df_state_age_2015 *A data.frame containing median age estimates for US states in 2015*

Description

A data.frame containing median age estimates for US states in 2015

Usage

```
data(df_state_age_2015)
```

References

Taken from the US American Community Survey (ACS) 5 year estimates.

df_state_demographics *A data.frame containing demographic statistics for each state plus the District of Columbia.*

Description

A data.frame containing demographic statistics for each state plus the District of Columbia.

Usage

```
data(df_state_demographics)
```

References

Data comes from the 2013 5-year American Community Survey (ACS). Data generated by ?get_state_demographics.

get_acs_data *Use tidycensus to obtain the data needed to create a choropleth map.*

Description

Use tidycensus to obtain the data needed to create a choropleth map.

Usage

```
get_acs_data(
  variable = NULL,
  tableId = NULL,
  column_idx = NULL,
  map,
  endyear,
  span,
  census_api_key,
  include_moe = FALSE
)
```

Arguments

variable	The variable you wish to plot. A list of available census variables can be obtained using tidycensus::load_variables()
tableId	Alternatively, you may specify the ACS table you wish to plot. If the table has more than one variable inside it, you must also specify the index of the column you wish to plot.
column_idx	The index of the desired column within the table.
map	The type map you wish to create; either 'state', 'county', 'zip', or 'tract'

endyear	The end year of the survey to use.
span	Either 1, 3, or 5, the ACS vintage you wish to use.
census_api_key	Optional. Census API keys can be obtained at: https://api.census.gov/data/key_signup.html
include_moe	Whether to include the 90 percent margin of error.

`get_admin1_map` *Download a map of first level administrative regions from naturalearthdata.com*

Description

Uses the rnaturalearth package.

Usage

```
get_admin1_map(cache = TRUE, drop_geometry = TRUE)
```

Arguments

cache	Cache the map and use cached map if available.
drop_geometry	Drop geometry data?

Value

An sf dataframe uniquely identified by the variables "adm1_code", "diss_me", and "ne_id".

`get_county_demographics` *Get a handful of demographic variables on US Counties from the US Census Bureau as a data.frame.*

Description

The data comes from the American Community Survey (ACS). The variables are total population and median household income.

Usage

```
get_county_demographics(endyear = 2013, span = 5)
```

Arguments

endyear	The end year for the survey
span	The span of the survey

`get_state_demographics`

Get a handful of demographic variables on US States from the US Census Bureau as a data.frame.

Description

The data comes from the American Community Survey (ACS). The variables are total population and median household income.

Usage

```
get_state_demographics(endyear = 2013, span = 5)
```

Arguments

`endyear`	The end year for the survey
`span`	The span of the survey

`get_tract_demographics`

Get a handful of demographic variables on Census Tracts in a State from the US Census Bureau as a data.frame.

Description

The data comes from the American Community Survey (ACS). The variables are total population and median household income.

Usage

```
get_tract_demographics(
  state_name,
  county_fips = NULL,
  endyear = 2013,
  span = 5
)
```

Arguments

`state_name`	The name of the state. See ?state.regions for proper spelling and capitalization.
`county_fips`	An optional vector of county fips codes within the state. Useful to set because getting data on all tracts can be slow.
`endyear`	The end year for the survey
`span`	The span of the survey

get_tract_map *Download a map of all census tracts in a given state*

Description

The map returned is exactly the same map which tract_choropleth uses. It is downloaded using the "tracts" function in the tigris package, and then it is modified for use with choroplethr.

Usage

```
get_tract_map(state_name, drop_geometry = TRUE)
```

Arguments

state_name	The name of the state, given by proper name, abbreviation, for FIPS code.
drop_geometry	Drop geometry data?

state.map.bigdc *An sf containing geometry data for US states with DC enlarged*

Description

An sf containing geometry data for US states with DC enlarged

Usage

```
data(state.map.bigdc)
```

state.map.hex *An sf containing a hexagonal tile map for US states*

Description

An sf containing a hexagonal tile map for US states

Usage

```
data(state.map.hex)
```

References

obtained from: https://raw.githubusercontent.com/Z3tt/30DayMapChallenge/master/data/us_states_hexgrid.geojson.json

state.map.hires *An sf containing higher resolution geometry data for US states*

Description

Note: Resolution is still much lower than raw data from tigris

Usage

```
data(state.map.hires)
```

References

obtained using tigris::states()

state.map.lores *An sf containing lower resolution geometry data for US states*

Description

An sf containing lower resolution geometry data for US states

Usage

```
data(state.map.lores)
```

References

obtained using tigris::states()

state.regions *Supported regions for US states*

Description

Supported regions for US states

Usage

```
data(state.regions)
```

`state_choropleth` *Create a choropleth map using U.S. state level data*

Description

To see the list of allowed state names, see choroplethr::state.regions.

Usage

```
state_choropleth(
  df,
  geoid.name = "region",
  geoid.type = "auto",
  value.name = "value",
  style = "geographic_bigdc",
  num_colors = 7,
  color.max = NULL,
  color.min = NULL,
  na.color = "grey",
  custom.colors = NULL,
  nbreaks = 5,
  zoom = NULL,
  projection = "albers",
  border_color = "grey15",
  border_thickness = 0.2,
  background_color = "white",
  gridlines = FALSE,
  latlon_ticks = FALSE,
  whitespace = TRUE,
  label = NULL,
  label_text_size = 2.25,
  label_text_color = "black",
  label_box_color = "white",
  ggrepel_options = list(force = 0.01, box.padding = 0.15, label.padding = 0.15,
    max.overlaps = Inf),
  legend = NULL,
  legend_position = "right",
  title = NULL,
  return = "plot"
)
```

Arguments

`df`	A dataframe containing U.S. state level data
`geoid.name`	The variable that identifies each state

`geoid.type` How the variable given by geoid.name specifies each state (full name, abbreviation, etc). The allowed geoid.type are given in choroplethr::state.regions. If "auto", the function will try to automatically determine geoid.type.

`value.name` The name of the variable you wish to plot.

`style` Either "geographic" for a literal map of US states, "geographic_bigdc" to make Washington DC more visible, or "hexgrid" for a stylized hexagonal tile map. Note: projection = 'mercator' is suggested when using the hexgrid map.

`num_colors` The number of colors you want in your graph when plotting continuous data. If num_colors > 1, the variable in question will be divided into quantiles and converted into a factor with that many levels. If num_colors = 1, a continuous color gradient will be used; if num_colors = 0, a diverging color gradient will be used (useful for visualizing negative and positive numbers). Use color.max and color.min to control the range of colors displayed. num_colors is ignored when plotting categorical data.

`color.max` The color of the highest value in your data. Ignored if the plotted variable is categorical.

`color.min` The color of the lowest value in your data. Ignored if the plotted variable is categorical.

`na.color` The color you want to assign for regions with missing data

`custom.colors` A vector of valid R color terms of the to use for the map when plotting factor variables. The length of this vector must match the number of levels in your factor variable, or num_colors for a continuous variable that will be discretized by the function, and the order should match the order of the levels of in your factor variable.

`nbreaks` The number of breaks you wish to show in the legend when using a continuous color scale. Ignored if num_colors > 1.

`zoom` An optional vector of states to zoom in on, written in the same manner as geoid.name.

`projection` One of the following: "cartesian", "mercator", "robinson", or "albers", for equirectangular, Mercator, Robinson, and Albers Equal Area projections, respectively. When using the Mercator projection for world maps, setting limits_lon is recommended to prevent exaggeration of the size of Antarctica.

`border_color` The color of the borders on your map

`border_thickness` The thickness of the borders on your map

`background_color` The background color of your map

`gridlines` Should gridlines appear on your map?

`latlon_ticks` Should lat/lon tick marks appear on the edge of your map?

`whitespace` Add some blank space to the sides of your map? For some projections, this must be set to FALSE in order for lat/lon ticks and display correctly.

`label` The name of variable you wish to use to label your map; must be one of the variables that appears in the spatial dataframe just prior plotting (use return =

'sf' to see this dataframe), and in general, can be any of the allowed geoid.type. This function uses ggplot2::geom_label_repel to create the labels and ensure that they do not overlap.

`label_text_size`
: The size of the text that will appear in each label

`label_text_color`
: The color of the text that will appear in each label

`label_box_color`
: The color of the box around each label

`ggrepel_options`
: A list containing additional arguments to be passed to geom_label_repel (see ?ggplot2::geom_label_repel)

`legend`
: A title for your legend; if NULL, value.name will be used.

`legend_position`
: The position of your legend relative to the rest of the map; can be "top", "bottom", "left", or "right".

`title`
: A title for your plot; if NULL, no title will be added.

`return`
: If "plot", the function will return the requested map as a ggplot object. If "sf", the function will return the spatial dataframe used to draw the map (useful if you wish to customize the map yourself).

Examples

```
# Plot continuous state level data:
data(df_state_demographics)
state_choropleth(df = df_state_demographics,
                 geoid.name = 'region',
                 geoid.type = 'name.lower',
                 value.name = 'population',
                 title  = "U.S. State Population",
                 legend = "Population")

# Plot categorical data with custom colors:
data("df_president")
state_choropleth(df = df_president,
                 geoid.name = 'region',
                 geoid.type = 'name.lower',
                 value.name = 'value',
                 title  = "2012 US Presidential Election Results",
                 legend = "Candidate",
                 custom.colors = c('blue4', 'red3'),
                 border_color = 'lightgrey')

# Label states and pass additional arguments to ggrepel
state_choropleth(df = df_president,
                 geoid.name = 'region',
                 geoid.type = 'name.lower',
                 value.name = 'value',
                 title  = "2012 US Presidential Election Results",
```

```
                  legend = "Candidate",
                  custom.colors = c('blue4', 'red3'),
                  border_color = 'lightgrey',
                  label = 'state.abb',
                  label_text_size = 4,
                  ggrepel_options = list(label.r = 0, force = 0.02))

# Use a styled hexagonal tile map instead of actual state shapes:
state_choropleth(df = df_president,
                  style = 'hexgrid',
                  projection = 'mercator',
                  geoid.name = 'region',
                  geoid.type = 'name.lower',
                  value.name = 'value',
                  title  = "2012 US Presidential Election Results",
                  legend = "Candidate",
                  custom.colors = c('blue4', 'red3'),
                  border_color = 'lightgrey',
                  label = 'state.abb',
                  label_text_size = 3)
```

`state_choropleth_acs` *Create a US State choropleth from ACS data*

Description

Creates a choropleth of US states using the US Census' American Community Survey (ACS) data.

Usage

```
state_choropleth_acs(
  variable = NULL,
  tableId = NULL,
  column_idx = NULL,
  endyear,
  span = 5,
  title = NULL,
  census_api_key = NULL,
  ...
)
```

Arguments

`variable`	The variable you wish to plot. A list of available census variables can be obtained using tidycensus::load_variables()
`tableId`	Alternatively, you may specify the ACS table you wish to plot. If the table has more than one variable inside it, you must also specify the index of the column you wish to plot.

column_idx	The index of the desired column within the table.
endyear	The end year of the survey to use.
span	Either 1, 3, or 5, the ACS vintage you wish to use.
title	A title for the plot; if not specified, a title will be assigned based on the variable.
census_api_key	Optional. Census API keys can be obtained at: https://api.census.gov/data/key_signup.html
...	Other arguments passed to state_choropleth; see ?state_choropleth()

Value

A choropleth.

Examples

```
# Create a state choropleth for median household income zooming in
# on New York, New Jersey and Connecticut
state_choropleth_acs(variable = "B19013_001", endyear = 2011, num_colors=1,
zoom=c("new york", "new jersey", "connecticut"))
```

---

tract_choropleth *Create a choropleth map using census tract level data for a given state.*

Description

Create a choropleth map using census tract level data for a given state.

Usage

```
tract_choropleth(
  df,
  state_name,
  geoid.name = "region",
  geoid.type = "auto",
  value.name = "value",
  num_colors = 7,
  color.max = NULL,
  color.min = NULL,
  na.color = "grey",
  custom.colors = NULL,
  nbreaks = 5,
  tract_zoom = NULL,
  county_zoom = NULL,
  projection = "cartesian",
  border_color = "grey15",
  border_thickness = 0.2,
```

```
  background_color = "white",
  gridlines = FALSE,
  latlon_ticks = FALSE,
  whitespace = TRUE,
  label = NULL,
  label_text_size = 2.25,
  label_text_color = "black",
  label_box_color = "white",
  ggrepel_options = NULL,
  legend = NULL,
  legend_position = "right",
  title = NULL,
  return = "plot"
)
```

Arguments

df	A dataframe containing census tract level data for a given state.
state_name	The state in question, given by either proper name, abbreviation, or FIPS code.
geoid.name	The variable that identifies each tract.
geoid.type	How the variable given by geoid.name specifies each tract; the allowed geoid.type are given by the columns "AFFGEOID", "GEOID", or "tractid.numeric" variable obtained from get_tract_map(). If "auto", the function will try to automatically determine geoid.type.
value.name	The name of the variable you wish to plot.
num_colors	The number of colors you want in your graph when plotting continuous data. If num_colors > 1, the variable in question will be divided into quantiles and converted into a factor with that many levels. If num_colors = 1, a continuous color gradient will be used; if num_colors = 0, a diverging color gradient will be used (useful for visualizing negative and positive numbers). Use color.max and color.min to control the range of colors displayed. num_colors is ignored when plotting categorical data.
color.max	The color of the highest value in your data. Ignored if the plotted variable is categorical.
color.min	The color of the lowest value in your data. Ignored if the plotted variable is categorical.
na.color	The color you want to assign for regions with missing data
custom.colors	A vector of valid R color terms of the to use for the map when plotting factor variables. The length of this vector must match the number of levels in your factor variable, or num_colors for a continuous variable that will be discretized by the function, and the order should match the order of the levels of in your factor variable.
nbreaks	The number of breaks you wish to show in the legend when using a continuous color scale. Ignored if num_colors > 1.
tract_zoom	An optional vector of tracts to zoom in on, written in the same manner as geoid.name.

county_zoom An optional vector of countries to zoom in on, written as they appear in the "county.fips.numeric" column of the object returned from get_tract_map().

projection One of the following: "cartesian", "mercator", "robinson", or "albers", for equirectangular, Mercator, Robinson, and Albers Equal Area projections, respectively. When using the Mercator projection for world maps, setting limits_lon is recommended to prevent exaggeration of the size of Antarctica.

border_color The color of the borders on your map

border_thickness
The thickness of the borders on your map

background_color
The background color of your map

gridlines Should gridlines appear on your map?

latlon_ticks Should lat/lon tick marks appear on the edge of your map?

whitespace Add some blank space to the sides of your map? For some projections, this must be set to FALSE in order for lat/lon ticks and display correctly.

label The name of variable you wish to use to label your map; must be one of the variables that appears in the spatial dataframe just prior plotting (use return = 'sf' to see this dataframe), and in general, can be any of the allowed geoid.type. This function uses ggplot2::geom_label_repel to create the labels and ensure that they do not overlap.

label_text_size
The size of the text that will appear in each label

label_text_color
The color of the text that will appear in each label

label_box_color
The color of the box around each label

ggrepel_options
A list containing additional arguments to be passed to geom_label_repel (see ?ggplot2::geom_label_repel)

legend A title for your legend; if NULL, value.name will be used.

legend_position
The position of your legend relative to the rest of the map; can be "top", "bottom", "left", or "right".

title A title for your plot; if NULL, no title will be added.

return If "plot", the function will return the requested map as a ggplot object. If "sf", the function will return the spatial dataframe used to draw the map (useful if you wish to customize the map yourself).

See Also

https://www.census.gov/data/academy/data-gems/2018/tract.html for more information on Census Tracts

Examples

```
# Plot tract level data from New York state:
df_ny_tract_demographics = choroplethr::df_ny_tract_demographics
tract_choropleth(df = df_ny_tract_demographics, state_name = 'NY',
                 geoid.name = 'region', value.name = 'population')

# Zoom in on the five counties that comprise New York City:
tract_choropleth(df = df_ny_tract_demographics, state_name = 'NY',
                 geoid.name = 'region', value.name = 'population',
                 county_zoom = c(36005, 36047, 36061, 36081, 36085))
```

Index