Package ‘choroplethr’

April 12, 2025

Title Simplify the Creation of Choropleth Maps

Description Choropleths are thematic maps where geographic regions, such as states, are colored according to some metric, such as the number of people who live in that state. This package simplifies this process by 1. Providing ready-made functions for creating choropleths of common maps. 2. Providing data and API connections to interesting data sources for making choropleths. 3. Providing a framework for creating choropleths from arbitrary shapefiles. 4. Overlaying those maps over reference maps from 'Google Maps'.

Version 4.0.0

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

Copyright Trulia, Inc.

License BSD_3_clause + file LICENSE

Imports Hmisc, stringr, ggplot2 (>= 2.0.0), dplyr, R6, WDI, ggmap, RgoogleMaps, tigris (>= 1.0), gridExtra, xml2, rvest, tidyr, tidycensus

Suggests testthat, choroplethrMaps, choroplethrAdmin1 (>= 1.1.0),

Depends R (>= 3.5.0)

Collate 'acs.R' 'choropleth.R' 'admin1.R' 'admin1_region.R' 'choroplethr_wdi.R' 'country.R' 'usa.R' 'county.R' 'county_zoom.R' 'data.R' 'get_congress_116_party_data.R' 'get_congressional_demographics.R' 'get_usa_demographics.R' 'state.R' 'tract.R' 'utils.R'

RoxygenNote 7.3.2

Encoding UTF-8

NeedsCompilation no

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

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

Repository CRAN
Date/Publication 2025-04-11 22:10:08 UTC

Contents

`Admin1Choropleth` *An R6 object for creating Administration Level 1 choropleths.*

Description

An R6 object for creating Administration Level 1 choropleths.

An R6 object for creating Administration Level 1 choropleths.

Super class

`choroplethr::Choropleth` -> `Admin1Choropleth`

Methods

Public methods:

- `Admin1Choropleth$new()`
- `Admin1Choropleth$clone()`

Method `new()`**:**

Usage:

```
Admin1Choropleth$new(country.name, user.df)
```

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

Usage:

```
Admin1Choropleth$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`Admin1RegionChoropleth` *An R6 object for creating Administration Level 1 choropleths based on regions.*

Description

Compare with the Admin1Choropleth object, which creates Admin 1 choropleths based on Countries. This function is useful if you want a map that spans multiple countries - Especially if it only needs to include a portion of a country.

Super class

`choroplethr::Choropleth` -> `Admin1RegionChoropleth`

Methods

Public methods:

- `Admin1RegionChoropleth$new()`
- `Admin1RegionChoropleth$clone()`

Method `new()`**:**

Usage:

```
Admin1RegionChoropleth$new(user.df)
```

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

Usage:

```
Admin1RegionChoropleth$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`admin1_choropleth` *Create an admin1-level choropleth for a specified country*

Description

The map used comes from ?admin1.map in the choroplethrAdmin1 package. See ?get_admin_countries and ?get_admin_regions in the choroplethrAdmin1 package for help with the spelling of regions.

Usage

```
admin1_choropleth(
  country.name,
  df,
  title = "",
  legend = "",
  num_colors = 7,
  zoom = NULL,
  reference_map = FALSE
)
```

Arguments

`country.name`	The name of the country. Must exactly match how the country is named in the "country" column of ?admin1.regions in the choroplethrAdmin1 package.
`df`	A data.frame with a column named "region" and a column named "value". Elements in the "region" column must exactly match how regions are named in the "region" column in ?admin1.regions in the choroplethrAdmin1 package
`title`	An optional title for the map.
`legend`	An optional name for the legend.
`num_colors`	The number of colors on the map. A value of 1 will use a continuous scale. A value in [2, 9] will use that many colors.
`zoom`	An optional vector of regions to zoom in on. Elements of this vector must exactly match the names of regions as they appear in the "region" column of ?admin1.regions.
`reference_map`	If true, render the choropleth over a reference map from Google Maps.

Examples

```

library(choroplethrAdmin1)

data(df_japan_census)
head(df_japan_census)
# set the value we want to map to be the 2010 population estimates
df_japan_census$value=df_japan_census$pop_2010

# default map of all of japan
admin1_choropleth("japan",
                   df_japan_census,
                   "2010 Japan Population Estimates",
                   "Population")

# zoom in on the Kansai region and use a continuous scale
kansai = c("mie", "nara", "wakayama", "kyoto", "osaka", "hyogo", "shiga")
admin1_choropleth("japan",
                   df_japan_census,
                   "2010 Japan Population Estimates",
                   "Population",
                   1,
                   kansai)
```

admin1_region_choropleth

Create a map of Administrative Level 1 regions

Description

Unlike ?admin1_choropleth, the regions here can span multiple countries.

Usage

```
admin1_region_choropleth(
  df,
  title = "",
  legend = "",
  num_colors = 7,
  zoom = NULL,
  reference_map = FALSE
)
```

Arguments

`df`	A data.frame with a column named "region" and a column named "value". Elements in the "region" column must exactly match how regions are named in the "region" column in ?admin1.regions in the choroplethrAdmin1 package
`title`	An optional title for the map.
`legend`	An optional name for the legend.
`num_colors`	The number of colors on the map. A value of 1 will use a continuous scale. A value in [2, 9] will use that many colors.
`zoom`	An optional vector of regions to zoom in on. Elements of this vector must exactly match the names of regions as they appear in the "region" column of ?admin1.regions.
`reference_map`	If true, render the choropleth over a reference map from Google Maps.

Details

The map used comes from ?admin1.map in the choroplethrAdmin1 package. See ?get_admin_countries and ?get_admin_regions in the choroplethrAdmin1 package for help with the spelling of regions.

Examples

```
library(choroplethrAdmin1)

# map of continental us + southern canada

data("continental_us_states")
lower_canada = c("british columbia", "alberta", "saskatchewan", "manitoba", "ontario", "quebec")
regions = c(lower_canada, continental_us_states)
df = data.frame(region=regions, value=sample(1:length(regions)))

admin1_region_choropleth(df)
```

`calculate_percent_change`

Calculate the percentage change between two choroplethr dataframes.

Description

Merges df1 and df2 on column named "region", and computes percentage change from df1$value to df2$value. Result is in the new "value" column, and rounded to two digits.

Usage

```
calculate_percent_change(df1, df2)
```

Arguments

`df1`	A dataframe with columns named "region" and "value"
`df2`	A dataframe with columns named "region" and "value"

Examples

```
# load median age estimates from 2010 and 2015
data(df_state_age_2010)
data(df_state_age_2015)

df_age_diff = calculate_percent_change(df_state_age_2010, df_state_age_2015)
state_choropleth(df_age_diff,
    title      = "Percent Change in Median Age, 2010-2015",
    legend     = "Percent Change",
    num_colors = 0)
```

`Choropleth` *The base Choropleth object.*

Description

The base Choropleth object.

The base Choropleth object.

Methods

Public methods:

- `Choropleth$new()`
- `Choropleth$render()`
- `Choropleth$get_min_long()`
- `Choropleth$get_max_long()`
- `Choropleth$get_min_lat()`
- `Choropleth$get_max_lat()`
- `Choropleth$get_bounding_box()`
- `Choropleth$get_x_scale()`
- `Choropleth$get_y_scale()`
- `Choropleth$get_reference_map()`
- `Choropleth$get_choropleth_as_polygon()`
- `Choropleth$render_with_reference_map()`
- `Choropleth$clip()`
- `Choropleth$discretize()`
- `Choropleth$bind()`
- `Choropleth$prepare_map()`
- `Choropleth$get_scale()`
- `Choropleth$theme_clean()`
- `Choropleth$theme_inset()`
- `Choropleth$format_levels()`
- `Choropleth$set_zoom()`
- `Choropleth$get_zoom()`
- `Choropleth$set_num_colors()`
- `Choropleth$clone()`

Method `new()`:

Usage:

```
Choropleth$new(map.df, user.df)
```

Method `render()`:

Usage:

`Choropleth$render()`

Method `get_min_long():`

Usage:

`Choropleth$get_min_long()`

Method `get_max_long():`

Usage:

`Choropleth$get_max_long()`

Method `get_min_lat():`

Usage:

`Choropleth$get_min_lat()`

Method `get_max_lat():`

Usage:

`Choropleth$get_max_lat()`

Method `get_bounding_box():`

Usage:

`Choropleth$get_bounding_box(long_margin_percent, lat_margin_percent)`

Method `get_x_scale():`

Usage:

`Choropleth$get_x_scale()`

Method `get_y_scale():`

Usage:

`Choropleth$get_y_scale()`

Method `get_reference_map():`

Usage:

`Choropleth$get_reference_map()`

Method `get_choropleth_as_polygon():`

Usage:

`Choropleth$get_choropleth_as_polygon(alpha)`

Method `render_with_reference_map():`

Usage:

`Choropleth$render_with_reference_map(alpha = 0.5)`

Method `clip():`

Usage:

`Choropleth$clip()`

Method `discretize()`:

Usage:

```
Choropleth$discretize()
```

Method `bind()`:

Usage:

```
Choropleth$bind()
```

Method `prepare_map()`:

Usage:

```
Choropleth$prepare_map()
```

Method `get_scale()`:

Usage:

```
Choropleth$get_scale()
```

Method `theme_clean()`:

Usage:

```
Choropleth$theme_clean()
```

Method `theme_inset()`:

Usage:

```
Choropleth$theme_inset()
```

Method `format_levels()`:

Usage:

```
Choropleth$format_levels(x, nsep = " to ")
```

Method `set_zoom()`:

Usage:

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

Method `get_zoom()`:

Usage:

```
Choropleth$get_zoom()
```

Method `set_num_colors()`:

Usage:

```
Choropleth$set_num_colors(num_colors)
```

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.

choroplethr_wdi — *Create a country-level choropleth using data from the World Bank's World Development Indicators (WDI)*

Description

Create a country-level choropleth using data from the World Bank's World Development Indicators (WDI)

Usage

```
choroplethr_wdi(
  code = "SP.POP.TOTL",
  year = 2012,
  title = "",
  num_colors = 7,
  zoom = NULL
)
```

Arguments

`code`	The WDI code to use.
`year`	The year of data to use.
`title`	A title for the map. If not specified, automatically generated to include WDI code and year.
`num_colors`	The number of colors to use on the map. A value of 1 will use a continuous scale, and a value in [2, 9] will use that many colors.
`zoom`	An optional list of countries to zoom in on. Must come from the "name" column in ?country.regions.

Value

A choropleth.

References

Uses the WDI function from the WDI package by Vincent Arel-Bundock.

Examples

```
# See http://data.worldbank.org/indicator/SP.POP.TOTL
choroplethr_wdi(code="SP.POP.TOTL", year=2012, title="2012 Population Estimates", num_colors=1)

# See http://data.worldbank.org/indicator/SP.DYN.LE00.IN
choroplethr_wdi(code="SP.DYN.LE00.IN", year=2012, title="2012 Life Expectancy Estimates")

# See http://data.worldbank.org/indicator/NY.GDP.PCAP.CD
```

```
choroplethr_wdi(code="NY.GDP.PCAP.CD", year=2012, title="2012 Per Capita Income")
```

---

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

`CountryChoropleth` *An R6 object for creating country-level choropleths.*

Description

An R6 object for creating country-level choropleths.

An R6 object for creating country-level choropleths.

Super class

`choroplethr::Choropleth` -> `CountryChoropleth`

Methods

Public methods:

- `CountryChoropleth$new()`
- `CountryChoropleth$clone()`

Method `new()`**:**

Usage:

```
CountryChoropleth$new(user.df)
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`country_choropleth` *Create a country-level choropleth*

Description

The map used is country.map in the choroplethrMaps package. See country.regions for an object which can help you coerce your regions into the required format.

Usage

```
country_choropleth(df, title = "", legend = "", num_colors = 7, zoom = NULL)
```

Arguments

df	A data.frame with a column named "region" and a column named "value". Elements in the "region" column must exactly match how regions are named in the "region" column in ?country.map.
title	An optional title for the map.
legend	An optional name for the legend.
num_colors	The number of colors to use on the map. A value of 0 uses a divergent scale (useful for visualizing negative and positive numbers), A value of 1 uses a continuous scale (useful for visualizing outliers), and a value in [2, 9] will use that many quantiles.
zoom	An optional vector of countries to zoom in on. Elements of this vector must exactly match the names of countries as they appear in the "region" column of ?country.regions

Examples

```
# demonstrate default options
data(df_pop_country)
country_choropleth(df_pop_country, "2012 World Bank Populate Estimates")

# demonstrate continuous scale
country_choropleth(df_pop_country, "2012 World Bank Populate Estimates", num_colors=1)

# demonstrate zooming
country_choropleth(df_pop_country,
                   "2012 World Bank Population Estimates",
                   num_colors=1,
                   zoom=c("united states of america", "canada", "mexico"))
```

CountyChoropleth *Create a county-level choropleth*

Description

Create a county-level choropleth

Create a county-level choropleth

Super classes

`choroplethr::Choropleth` -> `choroplethr::USAChoropleth` -> `CountyChoropleth`

Methods

Public methods:

- `CountyChoropleth$new()`
- `CountyChoropleth$clip()`
- `CountyChoropleth$clone()`

Method `new()`**:**

Usage:

```
CountyChoropleth$new(user.df)
```

Method `clip()`**:**

Usage:

```
CountyChoropleth$clip()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`CountyZoomChoropleth` *Create a county-level choropleth that zooms on counties, not states.*

Description

Create a county-level choropleth that zooms on counties, not states.

Create a county-level choropleth that zooms on counties, not states.

Super class

`choroplethr::Choropleth` -> `CountyZoomChoropleth`

Methods

Public methods:

- `CountyZoomChoropleth$new()`
- `CountyZoomChoropleth$render()`
- `CountyZoomChoropleth$clone()`

Method `new()`**:**

Usage:

```
CountyZoomChoropleth$new(user.df)
```

Method `render()`**:**

Usage:

```
CountyZoomChoropleth$render()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`county_choropleth` *Create a choropleth of US Counties*

Description

The map used is county.map in the choroplethrMaps package. See country.regions in the choroplethrMaps package for an object which can help you coerce your regions into the required format.

Usage

```
county_choropleth(
  df,
  title = "",
  legend = "",
  num_colors = 7,
  state_zoom = NULL,
  county_zoom = NULL,
  reference_map = FALSE
)
```

Arguments

`df`	A data.frame with a column named "region" and a column named "value". Elements in the "region" column must exactly match how regions are named in the "region" column in county.map.
`title`	An optional title for the map.
`legend`	An optional name for the legend.
`num_colors`	The number of colors to use on the map. A value of 0 uses a divergent scale (useful for visualizing negative and positive numbers), A value of 1 uses a continuous scale (useful for visualizing outliers), and a value in [2, 9] will use that many quantiles.
`state_zoom`	An optional vector of states to zoom in on. Elements of this vector must exactly match the names of states as they appear in the "region" column of ?state.regions.
`county_zoom`	An optional vector of counties to zoom in on. Elements of this vector must exactly match the names of counties as they appear in the "region" column of ?county.regions.
`reference_map`	If true, render the choropleth over a reference map from Google Maps.

Examples

```
# default parameters
data(df_pop_county)
county_choropleth(df_pop_county,
                  title  = "US 2012 County Population Estimates",
                  legend = "Population")

# continuous scale
data(df_pop_county)
county_choropleth(df_pop_county,
                 title      = "US 2012 County Population Estimates",
                 legend     = "Population",
                 num_colors = 1,
                 state_zoom = c("california", "oregon", "washington"))

library(dplyr)
library(choroplethrMaps)
data(county.regions)

# show the population of the 5 counties (boroughs) that make up New York City
nyc_county_names = c("kings", "bronx", "new york", "queens", "richmond")
nyc_county_fips = county.regions %>%
  filter(state.name == "new york" & county.name %in% nyc_county_names) %>%
  select(region)
county_choropleth(df_pop_county,
                  title        = "Population of Counties in New York City",
                  legend       = "Population",
                  num_colors   = 1,
                  county_zoom = nyc_county_fips$region)
```

`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_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.

Examples

```
library(choroplethr)
data(df_county_demographics)

# examine the 2013, 5-year county percent hispanic estimates as a boxplot and choropleth

# the boxplot shows the distribution
boxplot(df_county_demographics$percent_hispanic)

# the choropleth map shows the location of the values
# first set the 'value' column to be the column we want to render
df_county_demographics$value = df_county_demographics$percent_hispanic
county_choropleth(df_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.

Examples

```
library(choroplethr)
data(df_state_demographics)

# examine the 2013, 5-year state percent hispanic estimates as a boxplot and choropleth

# the boxplot shows the distribution
boxplot(df_state_demographics$percent_hispanic)

# the choropleth map shows the location of the values
# first set the 'value' column to be the column we want to render
df_state_demographics$value = df_state_demographics$percent_hispanic
state_choropleth(df_state_demographics)
```

`double_map` *Place two maps side by side*

Description

With an optional title. Especially useful for contrasting choropleth maps both with and without a reference map underneath.

Usage

```
double_map(map1, map2, title = "")
```

Arguments

`map1`	The first map
`map2`	The second map
`title`	An optional title

`filter_to_voting_congressional_districts`
Remove non-voting Congressional Districts from a data.frame

Description

The data.frame must have a column named region with a 4-character Congressional District code.
Remove districts that have a district code of 98 (non-voting) or ZZ (undefined district). See https://www.census.gov/geographies/mapping-files/2019/dec/rdo/116-congressional-district-bef.html
At the time this function was created, tidycensus returned 5 non-voting districts. See https://github.com/walkerke/tidycensus/i

Usage

```
filter_to_voting_congressional_districts(df)
```

Arguments

`df`	A data.frame. Must have a column named region that contains character vectors of length 4. The first 2 characters should be a state FIPS code and the second 2 characters should be a Congressional District Number

---

`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_congressional_district_demographics

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

Description

The data comes from the American Community Survey (ACS). The variables are: total population, percent White not Hispanic, Percent Black or African American not Hispanic, percent Asian not Hispanic, percent Hispanic all races, per-capita income, median rent and median age.

Usage

```
get_congressional_district_demographics(year = 2018, survey = "acs5")
```

Arguments

`year`	The year the survey was published
`survey`	The survey. Either "acs5" or "acs1"

---

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

Examples

```
# get some demographic data on US counties from the 2010 5-year ACS
df = get_county_demographics(endyear=2010, span=5)
# A choropleth map shows the location of the values.
# Set the 'value' column to be the column we want to render.
df$value = df$median_hh_income
county_choropleth(df)
```

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

Examples

```
# get some demographic data on US states from the 2010 5-year ACS
df = get_state_demographics(endyear=2010, span=5)

# A choropleth map shows the location of the values.
# Set the 'value' column to be the column we want to render.
df$value = df$median_hh_income
state_choropleth(df)
```

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

Examples

```
# 36061 is the FIPS code for Manhatttan (technically "New York County"), NY.
df = get_tract_demographics("new york", 36061)
df$value = df$median_hh_income
tract_choropleth(df, "new york", county_zoom = 36061)
```

`get_tract_map` *Get a map of tracts in a state, as a data.frame*

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)
```

Arguments

`state_name`	The name of the state. See ?state.regions for proper spelling and capitalization.

`StateChoropleth` *Create a state-level choropleth*

Description

Create a state-level choropleth

Create a state-level choropleth

Super classes

`choroplethr::Choropleth` -> `choroplethr::USAChoropleth` -> `StateChoropleth`

Methods

Public methods:

- `StateChoropleth$new()`
- `StateChoropleth$render()`
- `StateChoropleth$clone()`

Method `new()`**:**

Usage:

```
StateChoropleth$new(user.df)
```

Method `render()`**:**

Usage:

```
StateChoropleth$render()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`state_choropleth` *Create a choropleth of US States*

Description

The map used is state.map in the package choroplethrMaps. See state.regions in the choroplethrMaps package for a data.frame that can help you coerce your regions into the required format.

Usage

```
state_choropleth(
  df,
  title = "",
  legend = "",
  num_colors = 7,
  zoom = NULL,
  reference_map = FALSE
)
```

Arguments

df	A data.frame with a column named "region" and a column named "value". Elements in the "region" column must exactly match how regions are named in the "region" column in state.map.
title	An optional title for the map.
legend	An optional name for the legend.
num_colors	The number of colors to use on the map. A value of 0 uses a divergent scale (useful for visualizing negative and positive numbers), A value of 1 uses a continuous scale (useful for visualizing outliers), and a value in [2, 9] will use that many quantiles.
zoom	An optional vector of states to zoom in on. Elements of this vector must exactly match the names of states as they appear in the "region" column of ?state.regions.
reference_map	If true, render the choropleth over a reference map from Google Maps.

Examples

```
# default parameters
data(df_pop_state)
state_choropleth(df_pop_state,
                 title  = "US 2012 State Population Estimates",
                 legend = "Population")

# continuous scale and zoom
data(df_pop_state)
state_choropleth(df_pop_state,
                 title      = "US 2012 State Population Estimates",
                 legend     = "Population",
                 num_colors = 1,
                 zoom       = c("california", "oregon", "washington"))

# demonstrate user creating their own discretization of the input
# demonstrate how choroplethr handles character and factor values
data(df_pop_state)
df_pop_state$str = ""
for (i in 1:nrow(df_pop_state))
{
  if (df_pop_state[i,"value"] < 1000000)
  {
    df_pop_state[i,"str"] = "< 1M"
  } else {
    df_pop_state[i,"str"] = "> 1M"
  }
}
df_pop_state$value = df_pop_state$str
state_choropleth(df_pop_state, title = "Which states have less than 1M people?")
```

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"))
```

`TractChoropleth` *An R6 object for creating choropleths of Census Tracts.*

Description

An R6 object for creating choropleths of Census Tracts.

An R6 object for creating choropleths of Census Tracts.

Super class

`choroplethr::Choropleth` -> `TractChoropleth`

Methods

Public methods:

- `TractChoropleth$new()`
- `TractChoropleth$set_zoom_tract()`
- `TractChoropleth$clone()`

Method `new()`**:**

Usage:

```
TractChoropleth$new(state_name, user.df)
```

Method `set_zoom_tract()`**:**

Usage:

```
TractChoropleth$set_zoom_tract(county_zoom, tract_zoom)
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`tract_choropleth` *Create a choropleth of Census Tracts in a particular state.*

Description

Create a choropleth of Census Tracts in a particular state.

Usage

```
tract_choropleth(
  df,
  state_name,
  title = "",
  legend = "",
  num_colors = 7,
  tract_zoom = NULL,
  county_zoom = NULL,
  reference_map = FALSE
)
```

Arguments

`df`	A data.frame with a column named "region" and a column named "value".
`state_name`	The name of the state. See ?state.regions for proper spelling and capitalization.
`title`	An optional title for the map.
`legend`	An optional name for the legend.
`num_colors`	The number of colors to use on the map. A value of 0 uses a divergent scale (useful for visualizing negative and positive numbers), A value of 1 uses a continuous scale (useful for visualizing outliers), and a value in [2, 9] will use that many quantiles.
`tract_zoom`	An optional vector of tracts to zoom in on. Elements of this vector must exactly match the names of tracts as they appear in the "region" column of the object returned from "get_tract_map".
`county_zoom`	An optional vector of county FIPS codes to zoom in on. Elements of this vector must exactly match the names of counties as they appear in the "county.fips.numeric" column of the object returned from "get_tract_map".
`reference_map`	If true, render the choropleth over a reference map from Google Maps.

See Also

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

`USAChoropleth` *Normal choropleth that draws Alaska and Hawaii as insets. In addition to a columns named "region" and "value", also requires a column named "state".*

Description

Normal choropleth that draws Alaska and Hawaii as insets. In addition to a columns named "region" and "value", also requires a column named "state".

Normal choropleth that draws Alaska and Hawaii as insets. In addition to a columns named "region" and "value", also requires a column named "state".

Super class

`choroplethr::Choropleth` -> `USAChoropleth`

Methods

Public methods:

- `USAChoropleth$new()`
- `USAChoropleth$render()`
- `USAChoropleth$render_helper()`
- `USAChoropleth$render_state_outline()`
- `USAChoropleth$set_zoom()`
- `USAChoropleth$clone()`

Method `new()`**:**

Usage:

```
USAChoropleth$new(map.df, user.df)
```

Method `render()`**:**

Usage:

```
USAChoropleth$render()
```

Method `render_helper()`**:**

Usage:

```
USAChoropleth$render_helper(choropleth.df, scale_name, theme)
```

Method `render_state_outline()`**:**

Usage:

```
USAChoropleth$render_state_outline(states)
```

Method `set_zoom()`**:**

Usage:

```
USAChoropleth$set_zoom(zoom)
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`visualize_df_by_race_ethnicity_party`
Create box plots to visualize race and ethnicity by party

Description

Requires a data.frame with specific column names. In practice, the data.frame is expected to come from a function like ?get_congressional_districts and then merged with a data.frame that has column "party".

Usage

```
visualize_df_by_race_ethnicity_party(df)
```

Arguments

`df`	A data.frame with columns "party", "percent_white", "percent_black", "percent_asian", "percent_hispanic"

Examples

```
data("df_congress116_demographics")
data("df_congress116_party")
df = merge(df_congress116_demographics, df_congress116_party)
# Race and Ethnicity of the 116th Congressional Districts using data from
# the 2018 5-year American Community Survey
visualize_df_by_race_ethnicity_party(df)
```

Index