Package ‘choroplethr’

August 11, 2020

Title  Simplify the Creation of Choropleth Maps in R
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 3.7.0
Maintainer Ari Lamstein <ari@lamsteinconsulting.com>
URL  www.choroplethr.com
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), acs
Collate  'acs.R' 'choropleth.R' 'admin1.R' 'admin1_region.R' 'choroplethr_animate.R' 'choropleth_wdi.R' 'country.R' 'usa.R'
  'county.R' 'county_zoom.R' 'data.R'
  'get_congress_116_party_data.R'
  'get_congressional_demographics.R' 'get_county_demographics.R'
  'get_state_demographics.R' 'get_tract_demographics.R'
  'startup_messages.R' 'state.R' 'tract.R' 'utils.R'
RoxygenNote 7.1.1
Encoding UTF-8
NeedsCompilation no
Author  Ari Lamstein [aut, cre],
  Brian Johnson [ctb],
  Trulia, Inc. [cph]
Repository CRAN
Date/Publication 2020-08-11 05:30:02 UTC

R topics documented:

  Admin1Choropleth ...........................................  3
  Admin1RegionChoropleth ....................................  4
  admin1_choropleth .........................................  4
  admin1_region_choropleth ..................................  6
  calculate_percent_change ...................................  7
  Choropleth ..................................................  8
  choroplethr_animate ........................................ 11
  choroplethr_wdi ............................................ 12
  congress116.regions ....................................... 13
  continental_us_states ..................................... 13
  CountryChoropleth ......................................... 14
  country_choropleth ........................................ 14
  CountyChoropleth .......................................... 15
  CountyZoomChoropleth ...................................... 16
  county_choropleth ......................................... 17
  county_choropleth_acs ..................................... 18
  df_congress116_demographics .............................. 20
  df_congress116_party ....................................... 20
  df_county_demographics .................................... 21
  df_japan_census ............................................. 21
  df_ny_tract_demographics .................................. 22
  df_pop_country .............................................. 22
  df_pop_county ............................................... 23
  df_pop_ny_tract ............................................. 23
  df_pop_state ................................................. 23
  df_president ............................................... 24
  df_president_ts ............................................. 24
  df_state_age_2010 .......................................... 25
  df_state_age_2015 .......................................... 25
  df_state_demographics ..................................... 26
  double_map ................................................... 26
  filter_to_voting_congressional_districts ............... 27
  get_acs_data ................................................ 27
  get_congressional_district_demographics .................. 28
  get_county_demographics .................................... 29
  get_state_demographics ..................................... 30
  get_tract_demographics ..................................... 31
  get_tract_map ............................................... 31
  StateChoropleth .............................................. 32
  state_choropleth ............................................ 33
  state_choropleth_acs ....................................... 34
  TractChoropleth .............................................. 35
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:
dee p 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

```r
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

```r
## Not run:
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,
  kansai)
```
"2010 Japan Population Estimates",
"Population",
1,
kansai)

admin1_choropleth("japan",
df_japan_census,
"2010 Japan Population Estimates",
"Population",
1,
kansai,
reference_map = TRUE)

## End(Not run)

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
```r
## Not run:
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)

## End(Not run)
```

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
```r
## Not run:
# 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)

## End(Not run)

---

**Choropleth**

The base Choropleth object.

**Description**

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$theme_inset()`
- `Choropleth$formatter_levels()`
- `Choropleth$set_scale()`
- `Choropleth$get_scale()`
- `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.
animate

Animate a list of choropleths

Description

Given a list of choropleths, represented as ggplot2 objects

1. Save the individual images to the working directory with the naming convention "choropleth_1.png", "choropleth_2.png", etc.
2. Write a file called "animated_choropleth.html" which contains a viewer which animates them.

Usage

choroplethr_animate(choropleths)

Arguments

choropleths A list of choropleths represented as ggplot2 objects.

Value

Nothing. However, a variable number of files are written to the current working directory.

Author(s)

Ari Lamstein (R code) and Brian Johnson (JavaScript, HTML and CSS code)

Examples

## Not run:
data(df_president_ts)
?df_president_ts # time series of all US presidential elections 1789-2012

# create a list of choropleths of presidential election results for each year
choropleths = list()
for (i in 2:(ncol(df_president_ts))) {
  df = df_president_ts[, c(1, i)]
  colnames(df) = c("region", "value")
  title = paste0("Presidential Election Results: ", colnames(df_president_ts)[i])
  choropleths[[i-1]] = state_choropleth(df, title=title)
}

# set working directory and animate
setwd("~/Desktop")
choroplethr_animate(choropleths)

## End(Not run)
Create a country-level choropleth using data from the World Bank’s World Development Indicators (WDI)

Usage

```r
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

```r
## Not run:
# 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")
```
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)


Description


Usage

data(continental_us_states)

Author(s)

Ari Lamstein
Description

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:
depth Whether to make a deep clone.

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

```r
# 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"))
```
Methods

Public methods:
• CountyZoomChoropleth$new()
• CountyZoomChoropleth$clip()
• CountyZoomChoropleth$clone()

Method new():
Usage:
CountyZoomChoropleth$new(user.df)

Method clip():
Usage:
CountyZoomChoropleth$clip()

Method clone(): The objects of this class are cloneable with this method.
Usage:
CountyZoomChoropleth$clone(deep = FALSE)
Arguments:
deep Whether to make a deep clone.
Usage:
CountyZoomChoropleth$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

```r
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.
## Not run:
```r
# default parameters
data(df_pop_county)
county_choropleth(df_pop_county,
                   title = "US 2012 County Population Estimates",
                   legend = "Population")

# zoom in on california and add a reference map
county_choropleth(df_pop_county,
                   title = "California County Population Estimates",
                   legend = "Population",
                   state_zoom = "california",
                   reference_map = TRUE)

# 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"))
```

```r
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)
```

## End(Not run)

---

**county_choropleth_acs**  
*Create a US County choropleth from ACS data*

**Description**

Creates a US County choropleth using the US Census’ American Community Survey (ACS) data. Requires the `acs` package to be installed, and a Census API Key to be set with the `acs`’s `api.key.install` function. Census API keys can be obtained at http://www.census.gov/developers/tos/key_request.html.
Usage

```r
county_choropleth_acs(
    tableId,
    endyear = 2011,
    span = 5,
    num_colors = 7,
    state_zoom = NULL,
    county_zoom = NULL
)
```

Arguments

- **tableId**: The id of an ACS table
- **endyear**: The end year of the survey to use. See `acs.fetch` and http://1.usa.gov/1geFSSj for details.
- **span**: The span of time to use. See `acs.fetch` and http://1.usa.gov/1geFSSj for details.
- **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.
- **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`.

Value

A choropleth.

References

Uses the acs package created by Ezra Haber Glenn.

See Also

- `api.key.install` in the acs package which sets an Census API key for the acs library

Examples

```r
## Not run:
# median income, all counties in US
county_choropleth_acs("B19301")

# continuous scale, zooming in on all counties in New York, New Jersey and Connecticut
county_choropleth_acs("B19301", num_colors=1, state_zoom=c("new york", "new jersey", "connecticut"))

# zooming in on the 5 counties (boroughs) that make up New York City
library(dplyr)
library(choroplethrMaps)
data(county.regions)

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_acs("B19301", num_colors=1, county_zoom=nyc_county_fips$region)

## End(Not run)

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_demographics

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**

```
## Not run:
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)

## End(Not run)
```

**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


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_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_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
df_state_age_2010

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

Usage

data(df_president_ts)

References


---

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**

```r
## Not run:
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)

## End(Not run)
```

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 = "")
filter_to_voting_congressional_districts

Remove non-voting Congressional Districts from a data.frame

Arguments

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

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

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/issues/277

Usage

filter_to_voting_congressional_districts(df)

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

Returns a list representing American Community Survey (ACS) estimates

Description

Given a map, ACS tableId, endyear and span. Prompts user for the column id if there are multiple tables. The first element of the list is a data.frame with estimates. The second element is the ACS title of the column. Requires the acs package to be installed, and a Census API Key to be set with the acs’s api.key.install function. Census API keys can be obtained at http://api.census.gov/data/key_signup.html.

Usage

get_acs_data(
  tableId,  # ACS tableId
  map,      # Map name
  endyear = 2012,  # End year
  span = 5,      # Span
  column_idx = -1,  # Column index
  include_moe = FALSE
)
get_congressional_district_demographics

Arguments

- `tableId`: The id of an ACS table
- `map`: The map you want to use. Must be one of "state", "county" or "zip".
- `endyear`: The end year of the survey to use. See acs.fetch (?acs.fetch) and http://1.usa.gov/1geFSSj for details.
- `span`: The span of time to use. See acs.fetch and http://1.usa.gov/1geFSSj for details. on the same longitude and latitude map to scale. This variable is only checked when the "states" variable is equal to all 50 states.
- `column_idx`: The optional column id of the table to use. If not specified and the table has multiple columns, you will be prompted for a column id.
- `include_moe`: Whether to include the 90 percent margin of error.

See Also


Examples

```r
## Not run:
library(Hmisc) # for cut2
# States with greater than 1M residents
df = get_acs_data("B01003", "state")[[1]] # population
df$value = cut2(df$value, c(0,1000000,Inf))
state_choropleth(df, title="States with a population over 1M", legend="Population")

# Counties with greater than or greater than 1M residents
df = get_acs_data("B01003", "county")[[1]] # population
df$value = cut2(df$value, c(0,1000000,Inf))
county_choropleth(df, title="Counties with a population over 1M", legend="Population")

## End(Not run)
```

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

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

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, 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_county_demographics(endyear = 2013, span = 5)

Arguments

- **endyear**: The end year for the survey
- **span**: The span of the survey

References

The choroplethr guide to Census data: http://www.arilamstein.com/open-source/choroplethr/mapping-us-census-data/


Examples

```r
## Not run:
# get some demographic data on US counties from the 2010 5-year ACS
df = get_county_demographics(endyear=2010, span=5)
colnames(df)

# analyze the percent of people who are white not hispanic
# a boxplot shows the distribution
boxplot(df$percent_white)

# a choropleth map shows the location of the values
# set the 'value' column to be the column we want to render
df$value = df$percent_white
county_choropleth(df)

## End(Not run)
```
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, 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_state_demographics(endyear = 2013, span = 5)

Arguments

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

References

The choroplethr guide to Census data: http://www.arilamstein.com/open-source/choroplethr/mapping-us-census-data/


Examples

## Not run:
# get some demographic data on US states from the 2010 5-year ACS
df = get_state_demographics(endyear=2010, span=5)
colnames(df)

# analyze the percent of people who are white not hispanic
# a boxplot shows the distribution
boxplot(df$percent_white)

# a choropleth map shows the location of the values
# set the 'value' column to be the column we want to render
df$value = df$percent_white
state_choropleth(df)

## End(Not run)
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, 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_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

References

The choroplethr guide to Census data: http://www.arilamstein.com/open-source/choroplethr/mapping-us-census-data/

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.

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

```r
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

```r
## Not run:
# default parameters
data(df_pop_state)
state_choropleth(df_pop_state,
  legend = "Population")

# choropleth over reference map of continental usa
data(continental_us_states)
state_choropleth(df_pop_state,
```
legend = "Population",
zoom = continental_us_states,
reference_map = TRUE)

# continuous scale and zoom
data(df_pop_state)
state_choropleth(df_pop_state,
    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?")

## End(Not run)

---

**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. Requires the acs package to be installed, and a Census API Key to be set with the acs’s api.key.install function. Census API keys can be obtained at http://www.census.gov/developers/tos/key_request.html.

**Usage**

```r
state_choropleth_acs(
    tableId,
    endyear = 2011,
    span = 5,
    num_colors = 7,
    zoom = NULL
)
```
Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tableId</td>
<td>The id of an ACS table</td>
</tr>
<tr>
<td>endyear</td>
<td>The end year of the survey to use. See <code>acs.fetch</code> and <a href="http://1.usa.gov/1geFSSj">http://1.usa.gov/1geFSSj</a> for details.</td>
</tr>
<tr>
<td>span</td>
<td>The span of time to use. See <code>acs.fetch</code> and <a href="http://1.usa.gov/1geFSSj">http://1.usa.gov/1geFSSj</a> for details.</td>
</tr>
<tr>
<td>num_colors</td>
<td>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.</td>
</tr>
<tr>
<td>zoom</td>
<td>An optional list of states to zoom in on. Must come from the &quot;name&quot; column in ?state.regions.</td>
</tr>
</tbody>
</table>

Value

A choropleth.

References

Uses the `acs` package created by Ezra Haber Glenn.

See Also

`api.key.install` in the `acs` package which sets an Census API key for the `acs` library


Examples

```r
## Not run:
# median income, default parameters
state_choropleth_acs("B19301")

# continuous scale, zooming in on New York, New Jersey and Connecticut
state_choropleth_acs("B19301", num_colors=1, zoom=c("new york", "new jersey", "connecticut"))

## End(Not run)
```

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:
deepee 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


---

**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()
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)
visualize_df_by_race_ethnicity_party

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

* **animation**
  choroplethr_animate, 11
* **choropleth**
  choroplethr_animate, 11
* **data**
  congress116.regions, 13
  continental_us_states, 13
  df_congress116_demographics, 20
  df_congress116_party, 20
  df_county_demographics, 21
  df_japan_census, 21
  df_ny_tract_demographics, 22
  df_pop_country, 22
  df_pop_county, 23
  df_pop_ny_tract, 23
  df_pop_state, 23
  df_president, 24
  df_president_ts, 24
  df_state_age_2010, 25
  df_state_age_2015, 25
  df_state_demographics, 26

  admin1_choropleth, 4
  admin1_region_choropleth, 6
  Admin1Choropleth, 3
  Admin1RegionChoropleth, 4
  calculate_percent_change, 7
  Choropleth, 8
  choroplethr::Choropleth, 3, 4, 14–16, 32, 35, 37
  choroplethr::USAChoropleth, 15, 32
  choroplethr_animate, 11
  choroplethr_wdi, 12
  congress116.regions, 13
  continental_us_states, 13
  country_choropleth, 14
  CountryChoropleth, 14
  county_choropleth, 17
  county_choropleth_acs, 18
  CountyChoropleth, 15
  CountyZoomChoropleth, 16
  df_congress116_demographics, 20
  df_congress116_party, 20
  df_county_demographics, 21
  df_japan_census, 21
  df_ny_tract_demographics, 22
  df_pop_country, 22
  df_pop_county, 23
  df_pop_ny_tract, 23
  df_pop_state, 23
  df_president, 24
  df_president_ts, 24
  df_state_age_2010, 25
  df_state_age_2015, 25
  df_state_demographics, 26
  double_map, 26
  filter_to_voting_congressional_districts, 27
  get_acs_data, 27
  get_congressional_district_demographics, 28
  get_county_demographics, 29
  get_tract_demographics, 30
  get_tract_map, 31
  state_choropleth, 33
  state_choropleth_acs, 34
  StateChoropleth, 32

  tract_choropleth, 36
  TractChoropleth, 35
  USAChoropleth, 37
  visualize_df_by_race_ethnicity_party, 38