Package 'geomander'

June 17, 2021

Type Package

Title Geographic Tools for Studying Gerrymandering

Version 1.0.8

Date 2021-06-14

Description A compilation of tools to complete common tasks for studying gerrymandering. This focuses on the geographic tool side of common problems, such as linking different levels of spatial units or estimating how to break up units. Functions exist for creating redistricting-focused data for the US.

License MIT + file LICENCE

URL https://christopherkenny.github.io/geomander/

BugReports https://github.com/christopherkenny/geomander/issues

RoxygenNote 7.1.1

LinkingTo Rcpp

Imports dplyr, magrittr, nngeo, sf, stringr, tibble, tidycensus, tidyr, tigris, ggplot2, readr, spdep

Depends R (>= 2.10)

Suggests redist, knitr, rmarkdown, testthat (>= 3.0.0)

LazyData true

Encoding UTF-8

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation yes

Author Christopher T. Kenny [aut, cre] (<https://orcid.org/0000-0002-9386-6860>)

Maintainer Christopher T. Kenny <christopherkenny@fas.harvard.edu>

Repository CRAN

Date/Publication 2021-06-16 22:50:02 UTC

R topics documented:

geomander-package *Geographic Tools for Studying Gerrymandering*

Description

A compilation of tools to complete common tasks for studying gerrymandering. This focuses on the geographic tool side of common problems, such as linking different levels of spatial units or estimating how to break up units. Functions exist for creating redistricting-focused data for the US.

Package Content

Index of help topics:

```
split_precinct          Split a Precinct
st_centerish            Get the kind of center of each shape
suggest_component_connection
                        Suggest Connections for Disconnected Groups
suggest_neighbors       Suggest Neighbors for Lonely Precincts
towns                   towns
va18sub                 va18sub
va_blocks               va_blocks
va_vtd                  va_vtd
```

Maintainer

NA

Author(s)

NA

---

add_edge *Add Edges to an Adjacency List*

Description

Add Edges to an Adjacency List

Usage

```
add_edge(adjacency, v1, v2, zero = TRUE)
```

Arguments

adjacency	list of adjacent precincts
v1	integer or integer array for first vertex to connect. If array, connects each to corresponding entry in v2.
v2	integer or integer array for second vertex to connect. If array, connects each to corresponding entry in v1.
zero	boolean, TRUE if the list is zero indexed. False if one indexed.

Value

adjacency list.

Examples

```
data(towns)
adj <- adjacency(towns)
add_edge(adj, 2, 3)
```

adjacency *Build Adjacency List*

Description

This is similar to the old version of redist's adjacency function which has been replaced by sf internals. This is faster but less reliable. This wraps spdep::poly2nb() and replaces the indicator for no adjacent precincts with a integer(0)

Usage

```
adjacency(shp, zero = TRUE, rook = TRUE)
```

Arguments

shp	sf dataframe
zero	Default is TRUE for zero indexed. FALSE gives a one indexed adjacency list.
rook	Default is TRUE for rook adjacency. FALSE gives queen adjacency

Value

list with nrow(shp) entries

Examples

```
data(precincts)
adj <- adjacency(precincts)
```

block2prec *Aggregate Block Table by Matches*

Description

Aggregates block table values up to a higher level, normally precincts, hence the name block2prec.

Usage

```
block2prec(block_table, matches, geometry = FALSE)
```

Arguments

block_table	Required. Block table output from create_block_table
matches	Required. Grouping variable to aggregate up by, typically made with geo_match
geometry	Boolean. Whether to keep geometry or not.

Value

dataframe with length(unique(matches)) rows

Examples

```
set.seed(1)
data(rockland)
rockland$id <- sample(1:2, nrow(rockland), TRUE)
block2prec(rockland, rockland$id)
```

`block2prec_by_county` *Aggregate Block Table by Matches and County*

Description

Performs the same type of operation as block2prec, but subsets a precinct geometry based on a County fips column. This helps get around the problem that county geometries often have borders that follow rivers and lead to funny shaped blocks. This guarantees that every block is matched to a precinct which is in the same county.

Usage

```
block2prec_by_county(block_table, precinct, precinct_county_fips)
```

Arguments

`block_table`	Required. Block table output from create_block_table
`precinct`	sf dataframe of shapefiles to match to.
`precinct_county_fips`	Column within precincts

Value

dataframe with nrow(precinct) rows

Examples

```
## Not run:
# Need Census API
data(towns)
towns$fips <- '087'
block <- create_block_table('NY', 'Rockland')
block2prec_by_county(block, towns, 'fips')

## End(Not run)
```

checkerboard *Checkerboard*

Description

This data set contains 64 squares in an 8x8 grid, like a checkerboard.

Usage

```
data("checkerboard")
```

Format

An sf dataframe with 64 observations

Examples

```
data("checkerboard")
```

checkerboard_adj *Checkerboard Adjacency*

Description

This data contains a zero indexed adjacency list for the checkerboard dataset.

Usage

```
data("checkerboard_adj")
```

Format

A list with 64 entries

Examples

```
data("checkerboard_adj")
```

check_contiguity *Check Contiguity by Group*

Description

Check Contiguity by Group

Usage

```
check_contiguity(adjacency, group)
```

Arguments

adjacency	adjacency list
group	array of group identifiers. Typically district numbers or county names.

Value

tibble with a column for each of inputted group, created group number, and the identified connected component number

Examples

```
data(checkerboard)
adj <- adjacency(checkerboard)
check_contiguity(adj)
```

compare_adjacencies *Compare Adjacency Lists*

Description

Compare Adjacency Lists

Usage

```
compare_adjacencies(adj1, adj2, shp, zero = TRUE)
```

Arguments

adj1	Required. A first adjacency list.
adj2	Required. A second adjacency list.
shp	shapefile to compare intersection types.
zero	Boolean. Defaults to TRUE. Are adj1 and adj2 zero indexed?

Value

tibble with row indices to compare, and optionally columns which describe the DE-9IM relationship between differences.

Examples

```
data(towns)
rook <- adjacency(towns)
sf_rook <- lapply(sf::st_relate(towns, pattern = 'F***1****'), function(x){x-1L})
compare_adjacencies(rook, sf_rook, zero = FALSE)
```

---

count_connections *Count Times Precincts are Connected*

Description

Count Times Precincts are Connected

Usage

```
count_connections(dm, normalize = FALSE)
```

Arguments

dm	district membership matrix
normalize	Whether to normalize all values by the number of columns.

Value

matrix with the number of connections between precincts

Examples

```
set.seed(1)
dm <- matrix(sample(1:2, size = 100, TRUE), 10)
count_connections(dm)
```

create_block_table *Create Block Level Data*

Description

Creates a block level dataset, using the decennial census information, with the standard redistricting variables.

Usage

```
create_block_table(state, county, geography = TRUE, year = 2010)
```

Arguments

state	Required. Two letter state postal code.
county	Optional. Name of county. If not provided, returns blocks for the entire state.
geography	Defaults to TRUE. Whether to return the geography or not.
year	year, must be 2010 at the moment. 2020 to be added once available. 2000 if rereleased.

Value

dataframe with data for each block in the selected region. Data includes 2 sets of columns for each race or ethnicity category: population (pop) and voting age population (vap)

Examples

```
## Not run:
# uses the Census API
create_block_table(state = 'NY', county = 'Rockland', geography = FALSE)

## End(Not run)
```

create_tract_table *Create Tract Level Data*

Description

Create Tract Level Data

Usage

```
create_tract_table(state, county, geography = TRUE, year = 2019)
```

Arguments

state	Required. Two letter state postal code.
county	Optional. Name of county. If not provided, returns tracts for the entire state.
geography	Defaults to TRUE. Whether to return the geography or not.
year	year, must be >= 2009 and <= 2019.

Value

dataframe with data for each tract in the selected region. Data includes 3 sets of columns for each race or ethnicity category: population (pop), voting age population (vap), and citizen voting age population (cvap)

Examples

```
## Not run:
# Relies on Census Bureau API
tract <- create_tract_table('NY', 'Rockland', year = 2018)

## End(Not run)
```

---

dra2r *DRA to R*

Description

Creates a block or precinct level dataset from DRA csv output.

Usage

```
dra2r(dra, state, precincts)
```

Arguments

dra	The path to an exported csv or a dataframe with columns Id and District, loaded from a DRA export.
state	the state postal code of the state
precincts	an sf dataframe of precinct shapes to link the output to

Value

sf dataframe either at the block level or precinct level

Examples

```
## Not run:
# Needs Census Bureau API
# dra_utah_test is available at https://bit.ly/3c6UDKk
blocklevel <- dra2r('dra_utah_test.csv', state = 'UT')

## End(Not run)
```

`estimate_down` *Estimate Down Levels*

Description

Non-geographic partner function to geo_estimate_down. Allows users to estimate down without the costly matching operation if they've already matched.

Usage

```
estimate_down(wts, value, group)
```

Arguments

`wts`	numeric vector. Defaults to 1. Typically population or VAP, as a weight to give each precinct.
`value`	numeric vector. Defaults to 1. Typically electoral outcomes, as a value to estimate down into blocks.
`group`	matches of length(wts) that correspond to row indices of value. Often, this input is the output of geo_match.

Value

numeric vector with each value split by weight

Examples

```
library(dplyr)
set.seed(1)
data(checkerboard)
counties <- checkerboard %>% group_by(id <= 32) %>%
summarize(geometry = sf::st_union(geometry)) %>% mutate(pop = c(100,200))
matches <- geo_match(checkerboard, counties)
estimate_down(wts = rep(1, nrow(checkerboard)), value = counties$pop, group = matches)
```

estimate_up *Estimate Up Levels*

Description

Non-geographic partner function to geo_estimate_up. Allows users to aggregate up without the costly matching operation if they've already matched.

Usage

```
estimate_up(value, group)
```

Arguments

value	numeric vector. Defaults to 1. Typically population values.
group	matches of length(value) that correspond to row indices of value. Often, this input is the output of geo_match.

Value

numeric vector with each value aggregated by group

Examples

```
library(dplyr)
set.seed(1)
data(checkerboard)
counties <- checkerboard %>% group_by(id <= 32) %>%
summarize(geometry = sf::st_union(geometry)) %>% mutate(pop = c(100,200))
matches <- geo_match(checkerboard, counties)
estimate_up(value = checkerboard$i, group = matches)
```

geo_estimate_down *Estimate Down Geography Levels*

Description

Simple method for estimating data down to a lower level. This is most often useful for getting election data down from a precinct level to a block level in the case that a state or other jurisdiction split precincts when creating districts. Geographic partner to estimate_down.

Usage

```
geo_estimate_down(from, to, wts, value, method = "center")
```

Arguments

from	Larger geography level
to	smaller geography level
wts	numeric vector of length nrow(to). Defaults to 1. Typically population or VAP, as a weight to give each precinct.
value	numeric vector of length nrow(from). Defaults to 1. Typically electoral outcomes, as a value to estimate down into blocks.
method	string from center, centroid, point, or area for matching levels

Value

numeric vector with each value split by weight

Examples

```
library(dplyr)
set.seed(1)
data(checkerboard)
counties <- checkerboard %>% group_by(id <= 32) %>%
summarize(geometry = sf::st_union(geometry)) %>% mutate(pop = c(100,200))
geo_estimate_down(from = counties, to = checkerboard, value = counties$pop)
```

---

geo_estimate_up *Estimate Up Geography Levels*

Description

Simple method for aggregating data up to a higher level This is most often useful for getting population data from a block level up to a precinct level. Geographic partner to estimate_up.

Usage

```
geo_estimate_up(from, to, value, method = "center")
```

Arguments

from	smaller geography level
to	larger geography level
value	numeric vector of length nrow(from). Defaults to 1.
method	string from center, centroid, point, or area for matching levels

Value

numeric vector with each value aggregated by group

Examples

```
library(dplyr)
 set.seed(1)
data(checkerboard)
counties <- checkerboard %>% group_by(id <= 32) %>%
summarize(geometry = sf::st_union(geometry)) %>% mutate(pop = c(100,200))
geo_estimate_up(from = checkerboard, to = counties, value = checkerboard$i)
```

`geo_filter` *Filter to Intersecting Pieces*

Description

Filter to Intersecting Pieces

Usage

```
geo_filter(from, to, bool = FALSE)
```

Arguments

`from`	Required. sf dataframe. the geography to subset
`to`	Required. sf dataframe. the geography that from must intersect
`bool`	Optional, defaults to FALSE. Should this just return a logical vector?

Value

sf data frame or logical vector if bool=TRUE

Examples

```
## Not run:
data(towns)
block <- create_block_table('NY', 'Rockland')
geo_filter(block, towns)

## End(Not run)
```

geo_match *Match Across Geographic Layers*

Description

Match Across Geographic Layers

Usage

```
geo_match(from, to, method = "center", tiebreaker = TRUE)
```

Arguments

from	smaller geographic level to match up from
to	larger geographic level to be matched to
method	string from center, centroid, point, or area for matching method
tiebreaker	Should ties be broken? boolean. If FALSE, precincts with no matches get value -1 and precincts with multiple matches get value -2.

Value

Integer Vector of matches length(to) with values in 1:nrow(from)

Examples

```
library(dplyr)
data(checkerboard)
counties <- sf::st_as_sf(as.data.frame(rbind(sf::st_union(checkerboard %>% filter(i < 4)),
sf::st_union(checkerboard %>% filter(i >= 4)) )))

geo_match(from = checkerboard, to = counties)
geo_match(from = checkerboard, to = counties, method = 'area')
```

geo_plot *Plots a Shape with Row Numbers as Text*

Description

One liner to plot a shape with row numbers

Usage

```
geo_plot(shp)
```

Arguments

shp	An sf shapefile

Value

ggplot

Examples

```
data(checkerboard)
geo_plot(checkerboard)
```

---

geo_plot_group *Create Plots of Shapes by Group with Connected Components Colored*

Description

Create Plots of Shapes by Group with Connected Components Colored

Usage

```
geo_plot_group(shp, adjacency, group, save = F, path = "")
```

Arguments

shp	An sf shapefile
adjacency	adjacency list
group	array of group identifiers. Typically district numbers or county names.
save	Boolean, whether to save or not.
path	Path to save, only used if save is TRUE. Defaults to working directory.

Value

list of ggplots

Examples

```
library(dplyr)
data("checkerboard")
data("checkerboard_adj")

checkerboard <- checkerboard %>% mutate(discont = as.integer(j == 5| j == 6))

p <- geo_plot_group(checkerboard, checkerboard_adj, checkerboard$discont)

p[[1]]
p[[2]]
```

geo_sort *Sort Precincts*

Description

Reorders precincts by distance from the NW corner of the bounding box.

Usage

```
geo_sort(shp)
```

Arguments

shp	sf dataframe, required.

Value

sf dataframe

Examples

```
data(checkerboard)
geo_sort(checkerboard)
```

geo_trim *Trim Away Small Pieces*

Description

Trim Away Small Pieces

Usage

```
geo_trim(from, to, thresh = 0.01, bool = FALSE)
```

Arguments

from	Required. sf dataframe. the geography to subset
to	Required. sf dataframe. the geography that from must intersect
thresh	Percent as decimal of an area to trim away. Default is .01, which is 1%.
bool	Optional, defaults to FALSE. Should this just return a logical vector?

Value

sf data frame or logical vector if bool=TRUE

Examples

```
## Not run:
# Needs Census Bureau API
data(towns)
block <- create_block_table('NY', 'Rockland')
geo_trim(block, towns, thresh = 0.05)

## End(Not run)


data(towns)
data(rockland)
sub <- geo_filter(rockland, towns)
rem <- geo_trim(sub, towns, thresh = 0.05)
```

`global_gearys` *Compute Global Geary's C*

Description

Computes the Global Geary's Contiguity statistic. Can produce spatial weights from an adjacency or sf data frame, in which case the spatial_mat is a contiguity matrix. Users can also provide a spatial_mat argument directly.

Usage

```
global_gearys(shp, adj, wts, spatial_mat)
```

Arguments

`shp`	sf data frame. Optional if adj or spatial_mat provided.
`adj`	zero indexed adjacency list. Optional if shp or spatial_mat provided.
`wts`	Required. Numeric vector with weights to use for Moran's I.
`spatial_mat`	matrix of spatial weights. Optional if shp or adj provided.

Value

double

Examples

```
library(dplyr)
data("checkerboard")
checkerboard <- checkerboard %>% mutate(m = as.numeric((id+i) %% 2 == 0))
global_gearys(shp = checkerboard, wts = checkerboard$m)
```

global_morans *Compute Global Moran's I*

Description

Computes the Global Moran's I statistic and expectation. Can produce spatial weights from an adjacency or sf data frame, in which case the spatial_mat is a contiguity matrix. Users can also provide a spatial_mat argument directly.

Usage

```
global_morans(shp, adj, wts, spatial_mat)
```

Arguments

shp	sf data frame. Optional if adj or spatial_mat provided.
adj	zero indexed adjacency list. Optional if shp or spatial_mat provided.
wts	Required. Numeric vector with weights to use for Moran's I.
spatial_mat	matrix of spatial weights. Optional if shp or adj provided.

Value

list

Examples

```
library(dplyr)
data("checkerboard")
checkerboard <- checkerboard %>% mutate(m = as.numeric((id+i) %% 2 == 0))
global_morans(shp = checkerboard, wts = checkerboard$m)
```

gstar_i *Compute Standardized Getis Ord G*i*

Description

Returns the Getis Ord G*i in standardized form.

Usage

```
gstar_i(shp, adj, wts, spatial_mat)
```

Arguments

shp	sf data frame. Optional if adj or spatial_mat provided.
adj	zero indexed adjacency list. Optional if shp or spatial_mat provided.
wts	Required. Numeric vector with weights to use for Moran's I.
spatial_mat	matrix of spatial weights. Optional if shp or adj provided.

Value

vector of G*i scores

Examples

```
library(dplyr)
data("checkerboard")
checkerboard <- checkerboard %>% mutate(m = as.numeric((id+i) %% 2 == 0))
gstar_i(shp = checkerboard, wts = checkerboard$m)
```

---

local_gearys *Compute Local Geary's C*

Description

Compute Local Geary's C

Usage

```
local_gearys(shp, adj, wts, spatial_mat)
```

Arguments

shp	sf data frame. Optional if adj or spatial_mat provided.
adj	zero indexed adjacency list. Optional if shp or spatial_mat provided.
wts	Required. Numeric vector with weights to use for Moran's I.
spatial_mat	matrix of spatial weights. Not required if shp or adj provided.

Value

numeric vector

Examples

```
library(dplyr)
data("checkerboard")
checkerboard <- checkerboard %>% mutate(m = as.numeric((id+i) %% 2 == 0))
local_gearys(shp = checkerboard, wts = checkerboard$m)
```

local_morans *Compute Local Moran's I*

Description

Compute Local Moran's I

Usage

```
local_morans(shp, adj, wts, spatial_mat)
```

Arguments

shp	sf data frame. Optional if adj or spatial_mat provided.
adj	zero indexed adjacency list. Optional if shp or spatial_mat provided.
wts	Required. Numeric vector with weights to use for Moran's I.
spatial_mat	matrix of spatial weights. Optional if shp or adj provided.

Value

tibble

Examples

```
library(dplyr)
data("checkerboard")
checkerboard <- checkerboard %>% mutate(m = as.numeric((id+i) %% 2 == 0))
local_morans(shp = checkerboard, wts = checkerboard$m)
```

nrcsd *nrcsd*

Description

The data contains the North Rockland Central School District.

Usage

```
data('nrcsd')
```

Format

An sf dataframe with 1 observation

Examples

```
data('nrcsd')
```

orange *orange*

Description

This data contains the blocks for Orange County NY, with geographies simplified to allow for better examples.

Usage

```
data("orange")
```

Format

An sf dataframe with 10034 observations

Details

It can be recreated with: orange <- create_block_table('NY', 'Orange') orange <- rmapshaper::ms_simplify(orange, keep_shapes = TRUE)

Examples

```
data("orange")
```

precincts *precincts*

Description

This data contains the election districts (or precincts) for Rockland County NY, with geographies simplified to allow for better examples.

Usage

```
data("precincts")
```

Format

An sf dataframe with 278 observations

References

https://www.rocklandgis.com/portal/apps/sites/#/data/datasets/2d91f9db816c48318848ad66eb1a18e9

Examples

```
data("precincts")
```

r2dra *R to DRA*

Description

Project a plan at the precinct level down to blocks into a format that can be used with DRA. Projecting down to blocks can take a lot of time for larger states.

Usage

```
r2dra(precincts, plan, state, path)
```

Arguments

precincts	Required. an sf dataframe of precinct shapes
plan	Required. Either a vector of district assignments or the name of a column in precincts with district assignments.
state	Required. the state postal code of the state
path	Optional. A path to try to save to. Warns if saving failed.

Value

tibble with columns Id, as used by DRA, identical to GEOID in census terms and District.

Examples

```
## Not run:
# Needs Census Bureau API
cd <- tigris::congressional_districts() %>% filter(STATEFP == '49')
cnty <- tigris::counties(state = 49)
matchedcty <- geo_match(from = cnty, to = cd)
# use counties as precincts and let the plan be their center match:
r2dra(cnty, matchedcty, 'UT', 'r2dra_ex.csv')

## End(Not run)
```

rockland *rockland*

Description

This data contains the blocks for Rockland County NY, with geographies simplified to allow for better examples.

Usage

```
data("rockland")
```

Format

An sf dataframe with 4764 observations

Details

It can be recreated with: rockland <- create_block_table('NY', 'Rockland') rockland <- rmapshaper::ms_simplify(rockland, keep_shapes = TRUE)

Examples

```
data("rockland")
```

---

split_precinct *Split a Precinct*

Description

States often split a precinct when they create districts but rarely provide the geography for the split precinct. This allows you to split a precinct using a lower geography, typically blocks.

Usage

```
split_precinct(lower, precinct, split_by, lower_wt, split_by_id)
```

Arguments

lower	The lower geography that makes up the precinct, this is often a block level geography.
precinct	The single precinct that you would like to split.
split_by	The upper geography that you want to split precinct by
lower_wt	Optional. Numeric weights to give to each precinct, typically VAP or population.
split_by_id	Optional. A string that names a column in split_by that identifies each observation in split_by

Value

sf data frame with precinct split

Examples

```
library(dplyr)
library(sf)
data(checkerboard)
low <- checkerboard %>% slice(1:3, 9:11)
prec <- checkerboard %>% slice(1:3) %>% summarize(geometry = st_union(geometry))
dists <-  checkerboard %>% slice(1:3, 9:11) %>% mutate(dist = c(1,2,2,1,3,3)) %>%
group_by(dist) %>% summarize(geometry = st_union(geometry))

split_precinct(low, prec, dists, split_by_id = 'dist')
```

`st_centerish` *Get the kind of center of each shape*

Description

Returns points within the shape, near the center. Uses the centroid if that's in the shape, or point on surface if not.

Usage

```
st_centerish(shp)
```

Arguments

shp	An sf dataframe

Value

An sf dataframe where geometry is the center(ish) of each shape in shp

Examples

```
data(towns)
st_centerish(towns)
```

suggest_component_connection *Suggest Connections for Disconnected Groups*

Description

Suggests nearest neighbors for connecting a disconnected group.

Usage

```
suggest_component_connection(shp, adjacency, group)
```

Arguments

shp	An sf data frame
adjacency	adjacency list
group	array of group identifiers. Typically district numbers or county names. Defaults to rep(1, length(adjacency)) if missing.

Value

tibble with two columns of suggested rows of shp to connect in adj

Examples

```
library(dplyr)
data(checkerboard)
checkerboard <- checkerboard %>% filter(i != 1, j != 1)
adj <- adjacency(checkerboard)
suggest_component_connection(checkerboard, adj)
```

suggest_neighbors *Suggest Neighbors for Lonely Precincts*

Description

For precincts which have no adjacent precincts, this suggests the nearest precinct as a friend to add. This is useful for when a small number of precincts are disconnected from the remainder of the geography, such as an island.

Usage

```
suggest_neighbors(shp, adjacency, idx, neighbors = 1)
```

Arguments

shp	an sf shapefile
adjacency	an adjacency list
idx	Optional. Which indices to suggest neighbors for. If blank, suggests for those with no neighbors.
neighbors	number of neighbors to suggest

Value

tibble with two columns of suggested rows of shp to connect in adj

Examples

```
library(dplyr)
data(va18sub)
va18sub <- va18sub %>% filter(!VTDST %in% c('000516', '000510', '000505', '000518'))
adj <- adjacency(va18sub)
suggests <- suggest_neighbors(va18sub, adj)
adj <- adj %>% add_edge(v1 = suggests$x, v2 = suggests$y)
```

---

towns *towns*

Description

This data contains 7 town boundaries for the towns which overlap North Rockland School District in NY.

Usage

```
data("towns")
```

Format

An sf dataframe with 7 observations

References

https://www.rocklandgis.com/portal/apps/sites/#/data/items/746ec7870a0b4f46b168e07369e79a27

Examples

```
data("towns")
```

va18sub *va18sub*

Description

This data contains a 90 precinct subset of Virginia from the 2018 Senate race. Contains results for Henrico County

Usage

```
data("va18sub")
```

Format

An sf dataframe with 90 observations

References

Voting and Election Science Team, 2019, "va_2018.zip", 2 018 Precinct-Level Election Results, https://doi.org/10.7910/DVN/UBKYRU/FQDLOO, Harvard Dataverse, V4

Examples

```
data("va18sub")
```

va_blocks *va_blocks*

Description

This data contains the blocks Henrico County, VA with geographies simplified to allow for better examples.

Usage

```
data("va_blocks")
```

Format

An sf dataframe with 6354 observations

Details

blocks87 <- create_block_table(state = 'VA', county = '087') va_blocks <- rmapshaper::ms_simplify(va_blocks, keep_shapes = TRUE)

Examples

```
data("va_blocks")
```

---

va_vtd *va_vtd*

Description

This data contains the blocks for Henrico County, VA with geographies simplified to allow for better examples.

Usage

```
data("va_blocks")
```

Format

An sf dataframe with 93 observations

Details

va_vtd <- tigris::voting_districts(state = 'VA') va_vtd <- rmapshaper::ms_simplify(va_vtd, keep_shapes = TRUE)

Examples

```
data("va_blocks")
```

Index