Package ‘redistmetrics’

April 11, 2022

Title Redistricting Metrics
Version 1.0.2
Date 2022-04-11
Description Reliable and flexible tools for scoring redistricting plans using common measures and metrics. These functions provide key direct access to tools useful for non-simulation analyses of redistricting plans, such as for measuring compactness or partisan fairness. Tools are designed to work with the ‘redist’ package seamlessly.

Depends R (>= 2.10)
Imports sf, Rcpp, cli, foreach, doParallel, magrittr, dplyr, rlang, geos, wk
Suggests rmarkdown, knitr, testthat (>= 3.0.0), ggplot2
LinkingTo Rcpp, RcppArmadillo
License MIT + file LICENSE
Encoding UTF-8
LazyData true
SystemRequirements C++11
RoxygenNote 7.1.2
BugReports https://github.com/alarm-redist/redistmetrics/issues
VignetteBuilder knitr
Config/testthat.edition 3
NeedsCompilation yes
Author Christopher T. Kenny [aut, cre], Cory McCartan [aut], Ben Fifield [aut], Kosuke Imai [aut]
Maintainer Christopher T. Kenny <christopherkenny@fas.harvard.edu>
Repository CRAN
Date/Publication 2022-04-11 14:40:02 UTC
R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>by_plan</td>
<td>3</td>
</tr>
<tr>
<td>compet_talisman</td>
<td>3</td>
</tr>
<tr>
<td>comp_bc</td>
<td>4</td>
</tr>
<tr>
<td>comp_box_reock</td>
<td>5</td>
</tr>
<tr>
<td>comp_ch</td>
<td>6</td>
</tr>
<tr>
<td>comp_edges_rem</td>
<td>6</td>
</tr>
<tr>
<td>comp_fh</td>
<td>7</td>
</tr>
<tr>
<td>comp_frac_kept</td>
<td>8</td>
</tr>
<tr>
<td>comp_log_st</td>
<td>9</td>
</tr>
<tr>
<td>comp_lw</td>
<td>10</td>
</tr>
<tr>
<td>comp_polsby</td>
<td>11</td>
</tr>
<tr>
<td>comp_reock</td>
<td>12</td>
</tr>
<tr>
<td>comp_schwartz</td>
<td>13</td>
</tr>
<tr>
<td>comp_skew</td>
<td>14</td>
</tr>
<tr>
<td>comp_x_sym</td>
<td>15</td>
</tr>
<tr>
<td>comp_y_sym</td>
<td>16</td>
</tr>
<tr>
<td>dist_euc</td>
<td>17</td>
</tr>
<tr>
<td>dist_ham</td>
<td>17</td>
</tr>
<tr>
<td>dist_info</td>
<td>18</td>
</tr>
<tr>
<td>dist_man</td>
<td>19</td>
</tr>
<tr>
<td>inc_pairs</td>
<td>19</td>
</tr>
<tr>
<td>list_fn</td>
<td>20</td>
</tr>
<tr>
<td>nh</td>
<td>21</td>
</tr>
<tr>
<td>nh_m</td>
<td>23</td>
</tr>
<tr>
<td>nh_map</td>
<td>23</td>
</tr>
<tr>
<td>nh_plans</td>
<td>25</td>
</tr>
<tr>
<td>part_bias</td>
<td>26</td>
</tr>
<tr>
<td>part_decl</td>
<td>27</td>
</tr>
<tr>
<td>part_decl_simple</td>
<td>28</td>
</tr>
<tr>
<td>part_dseats</td>
<td>29</td>
</tr>
<tr>
<td>part_dvs</td>
<td>29</td>
</tr>
<tr>
<td>part_egap</td>
<td>30</td>
</tr>
<tr>
<td>part_egap_ep</td>
<td>31</td>
</tr>
<tr>
<td>part_lop_wins</td>
<td>32</td>
</tr>
<tr>
<td>part_mean_median</td>
<td>33</td>
</tr>
<tr>
<td>part_resp</td>
<td>34</td>
</tr>
<tr>
<td>part_rmd</td>
<td>35</td>
</tr>
<tr>
<td>part_sscd</td>
<td>36</td>
</tr>
<tr>
<td>part_tau_gap</td>
<td>37</td>
</tr>
<tr>
<td>prep_perims</td>
<td>38</td>
</tr>
<tr>
<td>seg_dissim</td>
<td>38</td>
</tr>
<tr>
<td>splits_admin</td>
<td>39</td>
</tr>
<tr>
<td>splits_count</td>
<td>40</td>
</tr>
<tr>
<td>splits_multi</td>
<td>41</td>
</tr>
<tr>
<td>splits_sub_admin</td>
<td>41</td>
</tr>
<tr>
<td>splits_total</td>
<td>42</td>
</tr>
</tbody>
</table>
by_plan

Description

If x is repeated for each district, it returns a plan level value. Otherwise it returns x.

Usage

by_plan(x, ndists)

Arguments

x summary statistic at the district level
ndists numeric. Number of districts. Estimated as the gcd of the unique run length encodings if missing.

Value

x or plan level subset of x

Examples

by_plan(letters)
by_plan(rep(letters, each = 2))

compet_talisman

Compute Talismanic Redistricting Competitiveness Metric

Description

Compute Talismanic Redistricting Competitiveness Metric

Usage

compet_talisman(plans, shp, rvote, dvote, alpha = 1, beta = 1)

Arguments

plans redist_plans object or plans_matrix where each row indicates a district assignment and each column is a plan
shp redist_map object, tibble, or data frame containing other columns
rvote unquoted name of column in shp with group population
dvote unquoted name of column in shp with total population
alpha Numeric scaling value
beta Numeric scaling value
**Value**

numeric vector

**References**


**Examples**

data(nh)
data(nh_m)

# For a single plan:
compet_talisman(plans = nh$r_2020, shp = nh, rvote = nrv, dvote = ndv)

# Or many plans:
compet_talisman(plans = nh_m[, 3:5], shp = nh, rvote = nrv, dvote = ndv)

---

**comp_bc**

*Calculate Boyce Clark Ratio*

**Description**

Calculate Boyce Clark Ratio

**Usage**

comp_bc(plans, shp, epsg = 3857, ncores = 1)

**Arguments**

- **plans**: redist_plans object or plans_matrix where each row indicates a district assignment and each column is a plan
- **shp**: redist_map object or tibble with sf geometry column
- **epsg**: numeric EPSG code to planarize to. Default is 3857.
- **ncores**: numeric. Number of cores to use. Default is 1.

**Value**

numeric vector

**References**

Examples

data(nh)
data(nh_m)
# For a single plan:
comp_box_reock(plans = nh$r_2020, shp = nh)

# Or many plans:

# slower, beware!
comp_box_reock(plans = nh_m[, 3:5], shp = nh)

comp_box_reock  Calculate Box Reock Compactness

Description

Box reock is the ratio of the area of the district by the area of the minimum bounding box (of any rotation). Scores are bounded between 0 and 1, where 1 is most compact.

Usage

comp_box_reock(plans, shp, epsg = 3857, ncores = 1)

Arguments

plans  redist_plans object or plans_matrix where each row indicates a district assignment and each column is a plan
shp   redist_map object or tibble with sf geometry column
epsg numeric EPSG code to planarize to. Default is 3857.
ncores numeric. Number of cores to use. Default is 1.

Value

numeric vector

Examples

# data(nh)
data(nh_m)
# For a single plan:
comp_box_reock(plans = nh$r_2020, shp = nh)

# Or many plans:

# slower, beware!
comp_box_reock(plans = nh_m[, 3:5], shp = nh)
Calculate Convex Hull Compactness

**Usage**

```r
comp_ch(plans, shp, epsg = 3857, ncores = 1)
```

**Arguments**

- `plans`: redist_plans object or plans_matrix where each row indicates a district assignment and each column is a plan.
- `shp`: redist_map object or tibble with sf geometry column.
- `epsg`: numeric EPSG code to planarize to. Default is 3857.
- `ncores`: numeric. Number of cores to use. Default is 1.

**Value**

numeric vector

**Examples**

```r
data(nh)
data(nh_m)
# For a single plan:
comp_ch(plans = nh$r_2020, shp = nh)

# Or many plans:
comp_ch(plans = nh_m[,3:5], shp = nh)
```

Calculate Edges Removed Compactness

**Usage**

```r
comp_edges_rem(plans, shp, adj)
```

**Description**

Calculate Edges Removed Compactness
**comp_fh**

**Calculate Fryer Holden Compactness**

**Description**

Calculate Fryer Holden Compactness

**Usage**

```
comp_fh(plans, shp, total_pop, epsg = 3857, ncores = 1)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>plans</td>
<td>redist_plans object or plans_matrix where each row indicates a district assignment and each column is a plan</td>
</tr>
<tr>
<td>shp</td>
<td>redist_map object or tibble with sf geometry column</td>
</tr>
<tr>
<td>total_pop</td>
<td>A numeric vector with the population for every observation.</td>
</tr>
<tr>
<td>epsg</td>
<td>numeric EPSG code to planarize to. Default is 3857.</td>
</tr>
<tr>
<td>ncores</td>
<td>TRUE</td>
</tr>
</tbody>
</table>

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>plans</td>
<td>redist_plans object or plans_matrix where each row indicates a district assignment and each column is a plan</td>
</tr>
<tr>
<td>shp</td>
<td>redist_map object or tibble with sf geometry column</td>
</tr>
<tr>
<td>total_pop</td>
<td>A numeric vector with the population for every observation.</td>
</tr>
<tr>
<td>epsg</td>
<td>numeric EPSG code to planarize to. Default is 3857.</td>
</tr>
<tr>
<td>ncores</td>
<td>TRUE</td>
</tr>
</tbody>
</table>

**Value**

numeric vector

**References**

Matthew P. Dube and Jesse Tyler Clark. 2016. Beyond the circle: Measuring district compactness using graph theory. In Annual Meeting of the Northeastern Political Science Association

**Examples**

```r
data(nh)
data(nh_m)
# For a single plan:
comp_edges_rem(plans = nh$r_2020, shp = nh, nh$adj)

# Or many plans:
comp_edges_rem(plans = nh_m[, 3:5], shp = nh, nh$adj)
```
Value

numeric vector

References


Examples

data(nh)
data(nh_m)
# For a single plan:
comp_fh(plans = nh$r_2020, shp = nh, total_pop = pop)

# Or many plans:
comp_fh(plans = nh_m[, 3:5], shp = nh, pop)

comp_frac_kept  Calculate Fraction Kept Compactness

Description

Calculate Fraction Kept Compactness

Usage

comp_frac_kept(plans, shp, adj)

Arguments

plans redist_plans object or plans_matrix where each row indicates a district assignment and each column is a plan
shp redist_map object or tibble with sf geometry column
adj zero-indexed adjacency list. Not required if redist_map supplied to shp.

Value

numeric vector

References

Matthew P. Dube and Jesse Tyler Clark. 2016. Beyond the circle: Measuring district compactness using graph theory. In Annual Meeting of the Northeastern Political Science Association
Examples

```r
data(nh)
data(nh_m)
# For a single plan:
comp_frac_kept(plans = nh$r_2020, shp = nh, nh$adj)

# Or many plans:
comp_frac_kept(plans = nh_m[, 3:5], shp = nh, nh$adj)
```

---

**comp_log_st**

*Calculate Log Spanning Tree Compactness*

**Description**

Calculate Log Spanning Tree Compactness

**Usage**

```r
comp_log_st(plans, shp, counties = NULL, adj)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>plans</td>
<td>redist_plans object or plans_matrix where each row indicates a district assignment and each column is a plan</td>
</tr>
<tr>
<td>shp</td>
<td>redist_map object or tibble with sf geometry column</td>
</tr>
<tr>
<td>counties</td>
<td>column name in shp containing counties</td>
</tr>
<tr>
<td>adj</td>
<td>zero-indexed adjacency list. Not required if redist_map supplied to shp.</td>
</tr>
</tbody>
</table>

**Value**

numeric vector

**References**


**Examples**

```r
data(nh)
data(nh_m)
# For a single plan:
comp_log_st(plans = nh$r_2020, shp = nh, counties = county, adj = nh$adj)

# Or many plans:
comp_log_st(plans = nh_m[, 3:5], shp = nh, counties = county, adj = nh$adj)
```
comp_lw

Calculate Length Width Compactness

Description

Calculate Length Width Compactness

Usage

comp_lw(plans, shp, epsg = 3857, ncores = 1)

Arguments

- **plans**: redist_plans object or plans_matrix where each row indicates a district assignment and each column is a plan
- **shp**: redist_map object or tibble with sf geometry column
- **epsg**: numeric EPSG code to planarize to. Default is 3857.
- **ncores**: numeric. Number of cores to use. Default is 1.

Value

numeric vector

References


Examples

data(nh)
data(nh_m)
# For a single plan:
comp_lw(plans = nh$r_2020, shp = nh)

# Or many plans:
# slower, beware!
comp_lw(plans = nh_m[, 3:5], shp = nh)
Calculate Polsby Popper Compactness

**Description**

Calculate Polsby Popper Compactness

**Usage**

```r
comp_polsby(
  plans,
  shp,
  use_Rcpp,
  perim_path,
  perim_df,
  epsg = 3857,
  ncores = 1
)
```

**Arguments**

- `plans`: redist_plans object or plans_matrix where each row indicates a district assignment and each column is a plan.
- `shp`: redist_map object or tibble with sf geometry column.
- `use_Rcpp`: If TRUE (the default for more than 8 plans), precompute boundaries shared by each pair of units and use them to quickly compute the compactness score.
- `perim_path`: path to perimeter tibble saved by `prep_perims()`.
- `perim_df`: tibble of perimeters from `prep_perims()`.
- `epsg`: numeric EPSG code to planarize to. Default is 3857.
- `ncores`: numeric. Number of cores to use. Default is 1.

**Value**

numeric vector

**References**


Examples

```r
data(nh)
data(nh_m)
# For a single plan:
comp_reock(plans = nh$r_2020, shp = nh)

# Or many plans:
comp_reock(plans = nh_m[, 3:5], shp = nh)
```

Description

Calculate Reock Compactness

Usage

```r
comp_reock(plans, shp, epsg = 3857, ncores = 1)
```

Arguments

- `plans`: redist_plans object or plans_matrix where each row indicates a district assignment and each column is a plan.
- `shp`: redist_map object or tibble with sf geometry column.
- `epsg`: numeric EPSG code to planarize to. Default is 3857.
- `ncores`: numeric. Number of cores to use. Default is 1.

Value

numeric vector

References


Examples

```r
data(nh)
data(nh_m)
# For a single plan:
comp_reock(plans = nh$r_2020, shp = nh)

# Or many plans:
comp_reock(plans = nh_m[, 3:5], shp = nh)
```
comp_schwartz

Calculate Schwartzberg Compactness

Description

Calculate Schwartzberg Compactness

Usage

```r
comp_schwartz(
  plans,
  shp,
  use_Rcpp,
  perim_path,
  perim_df,
  epsg = 3857,
  ncores = 1
)
```

Arguments

- `plans`: redist_plans object or plans_matrix where each row indicates a district assignment and each column is a plan
- `shp`: redist_map object or tibble with sf geometry column
- `use_Rcpp`: Logical. Use Rcpp?
- `perim_path`: path to perimeter tibble saved by prep_perims()
- `perim_df`: tibble of perimeters from prep_perims()
- `epsg`: numeric EPSG code to planarize to. Default is 3857.
- `ncores`: numeric. Number of cores to use. Default is 1.

Value

numeric vector

References


Examples

```r
data(nh)
data(nh_m)
# For a single plan:
comp_schwartz(plans = nh$r_2020, shp = nh)
```
# Or many plans:
comp_schwartz(plans = nh_m[, 3:5], shp = nh)

---

**comp_skew**

*Calculate Skew Compactness*

**Description**

Skew is defined as the ratio of the radii of the largest inscribed circle with the smallest bounding circle. Scores are bounded between 0 and 1, where 1 is most compact.

**Usage**

```
comp_skew(plans, shp, epsg = 3857, ncores = 1)
```

**Arguments**

- **plans**: redist_plans object or plans_matrix where each row indicates a district assignment and each column is a plan
- **shp**: redist_map object or tibble with sf geometry column
- **epsg**: numeric EPSG code to planarize to. Default is 3857.
- **ncores**: numeric. Number of cores to use. Default is 1.

**Value**

numeric vector

**References**


**Examples**

```
data(nh)
data(nh_m)
# For a single plan:
comp_skew(plans = nh$r_2020, shp = nh)

# Or many plans:

# slower, beware!
comp_skew(plans = nh_m[, 3:5], shp = nh)
```
**comp_x_sym**

**Calculate X Symmetry Compactness**

**Description**

X symmetry is the overlapping area of a shape and its projection over the x-axis.

**Usage**

```
comp_x_sym(plans, shp, epsg = 3857, ncores = 1)
```

**Arguments**

- **plans**: redist_plans object or plans_matrix where each row indicates a district assignment and each column is a plan.
- **shp**: redist_map object or tibble with sf geometry column.
- **epsg**: numeric EPSG code to planarize to. Default is 3857.
- **ncores**: numeric. Number of cores to use. Default is 1.

**Value**

numeric vector

**References**


**Examples**

```r
# data(nh)
data(nh_m)
# For a single plan:
comp_x_sym(plans = nh$r_2020, shp = nh)

# Or many plans:

# slower, beware!
comp_x_sym(plans = nh_m[, 3:5], shp = nh)
```
Calculate Y Symmetry Compactness

**Description**

Y symmetry is the overlapping area of a shape and its projection over the y-axis.

**Usage**

```r
comp_y_sym(plans, shp, epsg = 3857, ncores = 1)
```

**Arguments**

- `plans`: redist_plans object or plans_matrix where each row indicates a district assignment and each column is a plan
- `shp`: redist_map object or tibble with sf geometry column
- `epsg`: numeric EPSG code to planarize to. Default is 3857.
- `ncores`: numeric. Number of cores to use. Default is 1.

**Value**

numeric vector

**References**


**Examples**

```r
# data(nh)
data(nh_m)
# For a single plan:
comp_y_sym(plans = nh$r_2020, shp = nh)

# Or many plans:

# slower, beware!
comp_y_sym(plans = nh_m[, 3:5], shp = nh)
```
**dist_euc**

*Calculate Euclidean Distances*

**Description**

Calculate Euclidean Distances

**Usage**

\[ \text{dist}_\text{euc}(\text{plans}, \text{ncores} = 1) \]

**Arguments**

- **plans**: redist_plans object or plans_matrix where each row indicates a district assignment and each column is a plan.
- **ncores**: numeric. Number of cores to use. Default is 1.

**Value**

matrix of plan distances

**Examples**

```r
data(nh)
data(nh_m)
# For a single plan (distance is trivial, 0):
dist_euc(plans = nh$r_2020)

# Or many plans:
dist_euc(plans = nh_m[, 3:5])
```

---

**dist_ham**

*Calculate Hamming Distances*

**Description**

Calculate Hamming Distances

**Usage**

\[ \text{dist}_\text{ham}(\text{plans}, \text{ncores} = 1) \]

**Arguments**

- **plans**: redist_plans object or plans_matrix where each row indicates a district assignment and each column is a plan.
- **ncores**: numeric. Number of cores to use. Default is 1.
dist_info

Calculate Variation of Information Distances

dist_info(plans, shp, total_pop, ncores = 1)

Arguments

<table>
<thead>
<tr>
<th>plans</th>
<th>redist_plans object or plans_matrix where each row indicates a district assignment and each column is a plan</th>
</tr>
</thead>
<tbody>
<tr>
<td>shp</td>
<td>redist_map object, tibble, or data frame containing other columns</td>
</tr>
<tr>
<td>total_pop</td>
<td>unquoted name of column in shp with total population</td>
</tr>
<tr>
<td>ncores</td>
<td>numeric. Number of cores to use. Default is 1.</td>
</tr>
</tbody>
</table>

Value

matrix of plan distances

Examples

data(nh)
data(nh_m)

# For a single plan (distance is trivial, 0):
dist_info(plans = nh$r_2020)

# Or many plans:
dist_info(plans = nh_m[, 3:5])
**dist_man**  
*Calculate Manhattan Distances*

**Description**

Calculate Manhattan Distances

**Usage**

```r
dist_man(plans, ncores = 1)
```

**Arguments**

- `plans`: redist_plans object or plans_matrix where each row indicates a district assignment and each column is a plan.
- `ncores`: numeric. Number of cores to use. Default is 1.

**Value**

matrix of plan distances

**Examples**

```r
data(nh)
data(nh_m)
# For a single plan (distance is trivial, 0):
dist_man(plans = nh$r_2020)

# Or many plans:
dist_man(plans = nh_m[, 3:5])
```

---

**inc_pairs**  
*Count Incumbent Pairings*

**Description**

Count the number of incumbents paired with at least one other incumbent.

**Usage**

```r
inc_pairs(plans, shp, inc)
```
Arguments

plans redist_plans object or plans_matrix where each row indicates a district assignment and each column is a plan
shp redist_map object, tibble, or data frame containing other columns
inc unquoted name of logical column in shp indicating where incumbents live

Value

vector of number of incumbents paired

Examples

data(nh)
data(nh_m)
# Use incumbent data:
fake_inc <- rep(FALSE, nrow(nh))
fake_inc[3:4] <- TRUE

# For a single plan:
inc_pairs(plans = nh$r_2020, shp = nh, inc = fake_inc)

# Or many plans:
inc_pairs(plans = nh_m[, 3:5], shp = nh, inc = fake_inc)

---

list_fn

Return Functions Matching a Prefix

Description

This package uses prefixes for each function that correspond to the type of measure. This function returns the functions

Usage

list_fn(prefix)

Arguments

prefix character prefix of functions to return

Value

character vector of functions

Examples

list_fn('part_')
Description

This data set contains demographic, election, and geographic information for the 326 voting tabulation districts in New Hampshire in 2020.

Usage

data("nh")

Format

A tibble with 326 rows and 45 columns

- GEOID20: 2020 VTD GEOID
- state: state name
- county: county name
- vtd: VTD portion of GEOID
- pop: total population
- pop_hisp: Hispanic population
- pop_white: White, not Hispanic population
- pop_black: Black, not Hispanic population
- pop_aian: American Indian and Alaska Native, not Hispanic population
- pop_asian: Asian, not Hispanic population
- pop_nhpi: Native Hawaiian and Pacific Islander, not Hispanic population
- pop_other: other race, not Hispanic population
- pop_two: multi-race, not Hispanic population
- vap: total voting-age population
- vap_hisp: Hispanic voting-age population
- vap_white: White, not Hispanic voting-age population
- vap_black: Black, not Hispanic voting-age population
- vap_aian: American Indian and Alaska Native, not Hispanic voting-age population
- vap_asian: Asian, not Hispanic voting-age population
- vap_nhpi: Native Hawaiian and Pacific Islander, not Hispanic voting-age population
- vap_other: other race, not Hispanic voting-age population
- vap_two: multi-race, not Hispanic voting-age population
- pre_16_rep_tru: Votes for Republican president 2016
- pre_16_dem_cli: Votes for Democratic president 2016
• `uss_16_rep_ayo`: Votes for Republican senate 2016
• `uss_16_dem_has`: Votes for Democratic senate 2016
• `gov_16_rep_sun`: Votes for Republican governor 2016
• `gov_16_dem_van`: Votes for Democratic governor 2016
• `gov_18_rep_sun`: Votes for Republican governor 2018
• `gov_18_dem_kel`: Votes for Democratic governor 2018
• `pre_20_dem_bid`: Votes for Democratic president 2020
• `pre_20_rep_tru`: Votes for Republican president 2020
• `uss_20_dem_sha`: Votes for Democratic senate 2020
• `uss_20_rep_mes`: Votes for Republican senate 2020
• `gov_20_dem_fel`: Votes for Democratic governor 2020
• `gov_20_rep_sun`: Votes for Republican governor 2020
• `arv_16`: Average Republican vote 2016
• `adv_16`: Average Democratic vote 2016
• `arv_18`: Average Republican vote 2018
• `adv_18`: Average Democratic vote 2018
• `arv_20`: Average Republican vote 2020
• `adv_20`: Average Democratic vote 2020
• `nrv`: Normal Republican vote
• `ndv`: Normal Democratic vote
• `geometry`: sf geometry, simplified for size using rmapshaper
• `r_2020`: Republican proposed plan for 2020 Congressional districts
• `d_2020`: Democratic proposed plan for 2020 Congressional districts
• `adj`: zero-indexed adjacency graph

References


Examples

data(nh)
### nh_m

*Redistricting Plans for New Hampshire as matrix*

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>This data set contains two reference plans (d_2020 and r_2020) and 50 simulated plans for New Hampshire, based on 2020 demographics, simulated at a population tolerance of 0.05%.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>data(&quot;nh_m&quot;)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>A matrix with 52 columns and 326 rows where each column is a plan</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>data(nh_m)</td>
</tr>
</tbody>
</table>

### nh_map

*New Hampshire Election and Demographic Data as a redist_map*

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>This data set contains demographic, election, and geographic information for the 326 voting tabulation districts in New Hampshire in 2020.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>data(&quot;nh_map&quot;)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>A redist_map with 326 rows and 45 columns</td>
</tr>
</tbody>
</table>

- GEOID20: 2020 VTD GEOID
- state: state name
- county: county name
- vtd: VTD portion of GEOID
- pop: total population
- pop_hisp: Hispanic population
- pop_white: White, not Hispanic population
- pop_black: Black, not Hispanic population
- `pop_aian`: American Indian and Alaska Native, not Hispanic population
- `pop_asian`: Asian, not Hispanic population
- `pop_nhpi`: Native Hawaiian and Pacific Islander, not Hispanic population
- `pop_other`: other race, not Hispanic population
- `pop_two`: multi-race, not Hispanic population
- `vap`: total voting-age population
- `vap_hisp`: Hispanic voting-age population
- `vap_white`: White, not Hispanic voting-age population
- `vap_black`: Black, not Hispanic voting-age population
- `vap_aian`: American Indian and Alaska Native, not Hispanic voting-age population
- `vap_asian`: Asian, not Hispanic voting-age population
- `vap_nhpi`: Native Hawaiian and Pacific Islander, not Hispanic voting-age population
- `vap_other`: other race, not Hispanic voting-age population
- `vap_two`: multi-race, not Hispanic voting-age population
- `pre_16_rep_tru`: Votes for Republican president 2016
- `pre_16_dem_clt`: Votes for Democratic president 2016
- `uss_16_rep_ayo`: Votes for Republican senate 2016
- `uss_16_dem_has`: Votes for Democratic senate 2016
- `gov_16_rep_sun`: Votes for Republican governor 2016
- `gov_16_dem_van`: Votes for Democratic governor 2016
- `gov_18_rep_sun`: Votes for Republican governor 2018
- `gov_18_dem_kel`: Votes for Democratic governor 2018
- `pre_20_dem_bid`: Votes for Democratic president 2020
- `pre_20_rep_tru`: Votes for Republican president 2020
- `uss_20_dem_sha`: Votes for Democratic senate 2020
- `uss_20_rep_mes`: Votes for Republican senate 2020
- `gov_20_dem_fel`: Votes for Democratic governor 2020
- `gov_20_rep_sun`: Votes for Republican governor 2020
- `arv_16`: Average Republican vote 2016
- `adv_16`: Average Democratic vote 2016
- `arv_18`: Average Republican vote 2018
- `adv_18`: Average Democratic vote 2018
- `arv_20`: Average Republican vote 2020
- `adv_20`: Average Democratic vote 2020
- `nrv`: Normal Republican vote
- `ndv`: Normal Democratic vote
- `r_2020`: Republican proposed plan for 2020 Congressional districts
- `d_2020`: Democratic proposed plan for 2020 Congressional districts
- `adj`: zero-indexed adjacency graph
- `geometry`: sf geometry, simplified for size using rmapshaper
References


Examples

data(nh_map)

data(nh_plans)

Description

This data set contains two reference plans (d_2020 and r_2020) and 50 simulated plans for New Hampshire, based on 2020 demographics, simulated at a population tolerance of 0.05%.

Usage

data("nh_plans")

Format

A redist_plans with 104 rows and 3 columns

- draw: factor identifying the reference plans (d_2020 and r_2020) and 50 simulated plans
- district: district number (1 or 2)
- total_pop: total population in the district

Examples

data(nh_plans)
part_bias

Calculate Partisan Bias

Description

Calculate Partisan Bias

Usage

part_bias(plans, shp, dvote, rvote, v = 0.5)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>plans</td>
<td>redist_plans object or plans_matrix where each row indicates a district assignment and each column is a plan</td>
</tr>
<tr>
<td>shp</td>
<td>redist_map object, tibble, or data frame containing other columns</td>
</tr>
<tr>
<td>dvote</td>
<td>unquoted name of column in shp with total population</td>
</tr>
<tr>
<td>rvote</td>
<td>unquoted name of column in shp with group population</td>
</tr>
<tr>
<td>v</td>
<td>vote share to calculate bias at. Numeric. Default is 0.5.</td>
</tr>
</tbody>
</table>

Value

numeric vector

References


Examples

data(nh)
data(nh_m)
# For a single plan:
part_bias(plans = nh$r_2020, shp = nh, rvote = nrv, dvote = ndv)

# Or many plans:
part_bias(plans = nh_m[, 3:5], shp = nh, rvote = nrv, dvote = ndv)
part_decl

Calculate Declination

Description

Calculate Declination

Usage

part_decl(plans, shp, dvote, rvote, normalize = TRUE, adjust = TRUE)

Arguments

- plans: redist_plans object or plans_matrix where each row indicates a district assignment and each column is a plan
- shp: redist_map object, tibble, or data frame containing other columns
- dvote: unquoted name of column in shp with total population
- rvote: unquoted name of column in shp with group population
- normalize: Default is TRUE. Translate score to an angle?
- adjust: Default is TRUE. Applies a correction to increase cross-size comparison.

Value

numeric vector

References


Examples

data(nh)
data(nh_m)
# For a single plan:
part_decl(plans = nh$r_2020, shp = nh, rvote = nrv, dvote = ndv)

# Or many plans:
part_decl(plans = nh_m[, 3:5], shp = nh, rvote = nrv, dvote = ndv)
part_decl_simple  

*Calculate Simplified Declination*

**Description**

Calculate Simplified Declination

**Usage**

```r
part_decl_simple(plans, shp, dvote, rvote)
```

**Arguments**

- `plans`: redist_plans object or plans_matrix where each row indicates a district assignment and each column is a plan
- `shp`: redist_map object, tibble, or data frame containing other columns
- `dvote`: unquoted name of column in shp with total population
- `rvote`: unquoted name of column in shp with group population

**Value**

numeric vector

**References**


**Examples**

```r
data(nh)
data(nh_m)
# For a single plan:
part_decl_simple(plans = nh$r_2020, shp = nh, rvote = nrv, dvote = ndv)

# Or many plans:
part_decl_simple(plans = nh_m[, 3:5], shp = nh, rvote = nrv, dvote = ndv)
```
part_dseats  

Calculate Democratic Seats

Description

Calculate Democratic Seats

Usage

\texttt{part_dseats(plans, shp, dvote, rvote)}

Arguments

- \texttt{plans}: redist\_plans object or plans\_matrix where each row indicates a district assignment and each column is a plan
- \texttt{shp}: redist\_map object, tibble, or data frame containing other columns
- \texttt{dvote}: unquoted name of column in \texttt{shp} with total population
- \texttt{rvote}: unquoted name of column in \texttt{shp} with group population

Value

numeric vector

Examples

\begin{verbatim}
data(nh)
data(nh_m)
# For a single plan:
part_dseats(plans = nh$r_2020, shp = nh, rvote = nrv, dvote = ndv)

# Or many plans:
part_dseats(plans = nh_m[, 3:5], shp = nh, rvote = nrv, dvote = ndv)
\end{verbatim}

part_dvs  

Calculate Democratic Vote Share

Description

Calculate Democratic Vote Share

Usage

\texttt{part_dvs(plans, shp, dvote, rvote)}
part_egap

**Arguments**

- **plans**: redist_plans object or plans_matrix where each row indicates a district assignment and each column is a plan.
- **shp**: redist_map object, tibble, or data frame containing other columns.
- **dvote**: unquoted name of column in shp with total population.
- **rvote**: unquoted name of column in shp with group population.

**Value**

numeric vector

**Examples**

```r
data(nh)
data(nh_m)
# For a single plan:
part_dvs(plans = nh$r_2020, shp = nh, rvote = nrv, dvote = ndv)

# Or many plans:
part_dvs(plans = nh_m[, 3:5], shp = nh, rvote = nrv, dvote = ndv)
```

---

**part_egap**

*Calculate Efficiency Gap*

**Description**

Calculate Efficiency Gap.

**Usage**

`part_egap(plans, shp, dvote, rvote)`

**Arguments**

- **plans**: redist_plans object or plans_matrix where each row indicates a district assignment and each column is a plan.
- **shp**: redist_map object, tibble, or data frame containing other columns.
- **dvote**: unquoted name of column in shp with total population.
- **rvote**: unquoted name of column in shp with group population.

**Value**

numeric vector
part_egap_ep

References


Examples

data(nh)
data(nh_m)
# For a single plan:
part_egap(plans = nh$r_2020, shp = nh, rvote = nrv, dvote = ndv)

# Or many plans:
part_egap(plans = nh_m[, 3:5], shp = nh, rvote = nrv, dvote = ndv)

part_egap_ep

Calculate Efficiency Gap (Equal Population Assumption)

Description

Calculate Efficiency Gap (Equal Population Assumption)

Usage

part_egap_ep(plans, shp, dvote, rvote)

Arguments

plans redist_plans object or plans_matrix where each row indicates a district assignment and each column is a plan
shp redist_map object, tibble, or data frame containing other columns
dvote unquoted name of column in shp with total population
rvote unquoted name of column in shp with group population

Value

numeric vector

References

Examples

data(nh)
data(nh_m)
# For a single plan:
part_egap_ep(plans = nh$r_2020, shp = nh, rvote = nrv, dvote = ndv)

# Or many plans:
part_egap_ep(plans = nh_m[, 3:5], shp = nh, rvote = nrv, dvote = ndv)

part_lop_wins

Calculate Lopsided Wins

Description

Calculate Lopsided Wins

Usage

part_lop_wins(plans, shp, dvote, rvote)

Arguments

plans redist_plans object or plans_matrix where each row indicates a district assignment and each column is a plan
shp redist_map object, tibble, or data frame containing other columns
dvote unquoted name of column in shp with total population
rvote unquoted name of column in shp with group population

Value

numeric vector

References


Examples

data(nh)
data(nh_m)
# For a single plan:
part_lop_wins(plans = nh$r_2020, shp = nh, rvote = nrv, dvote = ndv)

# Or many plans:
part_lop_wins(plans = nh_m[, 3:5], shp = nh, rvote = nrv, dvote = ndv)
part_mean_median  

Calculate Mean Median Score

Description

Calculate Mean Median Score

Usage

part_mean_median(plans, shp, dvote, rvote)

Arguments

- plans: redist_plans object or plans_matrix where each row indicates a district assignment and each column is a plan
- shp: redist_map object, tibble, or data frame containing other columns
- dvote: unquoted name of column in shp with total population
- rvote: unquoted name of column in shp with group population

Value

numeric vector

References


Examples

data(nh)
data(nh_m)
# zero for the two district case:
# For a single plan:
part_mean_median(plans = nh$r_2020, shp = nh, rvote = nrv, dvote = ndv)

# Or many plans:
part_mean_median(plans = nh_m[, 3:5], shp = nh, rvote = nrv, dvote = ndv)
Calculate Responsiveness

Description

Calculate Responsiveness

Usage

part_resp(plans, shp, dvote, rvote, v = 0.5, bandwidth = 0.01)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>plans</td>
<td>redist_plans object or plans_matrix where each row indicates a district assignment and each column is a plan</td>
</tr>
<tr>
<td>shp</td>
<td>redist_map object, tibble, or data frame containing other columns</td>
</tr>
<tr>
<td>dvote</td>
<td>unquoted name of column in shp with total population</td>
</tr>
<tr>
<td>rvote</td>
<td>unquoted name of column in shp with group population</td>
</tr>
<tr>
<td>v</td>
<td>vote share to calculate bias at. Numeric. Default is 0.5.</td>
</tr>
<tr>
<td>bandwidth</td>
<td>Defaults to 0.01. A value between 0 and 1 for the step size to estimate the slope.</td>
</tr>
</tbody>
</table>

Value

numeric vector

References


Examples

data(nh)
data(nh_m)

# For a single plan:
part_resp(plans = nh$r_2020, shp = nh, rvote = nrv, dvote = ndv)

# Or many plans:
part_resp(plans = nh_m[, 3:5], shp = nh, rvote = nrv, dvote = ndv)
part_rmd

Calculate Ranked Marginal Deviation

Description

Calculate Ranked Marginal Deviation

Usage

part_rmd(plans, shp, dvote, rvote)

Arguments

- **plans**: redist_plans object or plans_matrix where each row indicates a district assignment and each column is a plan
- **shp**: redist_map object, tibble, or data frame containing other columns
- **dvote**: unquoted name of column in shp with total population
- **rvote**: unquoted name of column in shp with group population

Value

numeric vector

References


Examples

data(nh)
data(nh_m)
# For a single plan:
part_rmd(plans = nh$r_2020, shp = nh, rvote = nrv, dvote = ndv)

# Or many plans:
part_rmd(plans = nh_m[, 3:5], shp = nh, rvote = nrv, dvote = ndv)
part_sscd 

**Calculate Smoothed Seat Count Deviation**

**Description**

Calculate Smoothed Seat Count Deviation

**Usage**

```r
calculate_smoothed_seat_count_deviation(plans, shp, dvote, rvote)
```

**Arguments**

- `plans`: redist_plans object or plans_matrix where each row indicates a district assignment and each column is a plan
- `shp`: redist_map object, tibble, or data frame containing other columns
- `dvote`: unquoted name of column in shp with total population
- `rvote`: unquoted name of column in shp with group population

**Value**

numeric vector

**References**


**Examples**

```r
data(nh)
data(nh_m)
# For a single plan:
part_sscd(plans = nh$r_2020, shp = nh, rvote = nrv, dvote = ndv)

# Or many plans:
part_sscd(plans = nh_m[, 3:5], shp = nh, rvote = nrv, dvote = ndv)
```
part_tau_gap

Calculate Tau Gap

Description

Calculate Tau Gap

Usage

\[
\text{part\_tau\_gap}(\text{plans}, \text{shp}, \text{dvote}, \text{rvote}, \text{tau} = 1)
\]

Arguments

- \text{plans}: redist\_plans object or plans\_matrix where each row indicates a district assignment and each column is a plan
- \text{shp}: redist\_map object, tibble, or data frame containing other columns
- \text{dvote}: unqouted name of column in \text{shp} with total population
- \text{rvote}: unqouted name of column in \text{shp} with group population
- \text{tau}: A non-negative numeric for calculating Tau Gap. Defaults to 1.

Value

numeric vector

References


Examples

```r
data(nh)
data(nh_m)
# For a single plan:
part_tau_gap(plans = nh$r_2020, shp = nh, rvote = nrv, dvote = ndv)

# Or many plans:
part_tau_gap(plans = nh_m[, 3:5], shp = nh, rvote = nrv, dvote = ndv)
```
**prep_perims**  
*Prep Polsby Popper Perimeter Tibble*

**Description**
Replaces `redist.prep.polsbypopper`

**Usage**
```r
prep_perims(shp, epsg = 3857, perim_path, ncores = 1)
```

**Arguments**
- `shp`: `redist_map` object or tibble with sf geometry column
- `epsg`: numeric EPSG code to planarize to. Default is 3857.
- `perim_path`: A path to save an rds
- `ncores`: numeric. Number of cores to use. Default is 1.

**Value**
tibble of perimeters and lengths

**Examples**
```r
data(nh)
prep_perims(nh)
```

---

**seg_dissim**  
*Compute Dissimilarity Index*

**Description**
Compute Dissimilarity Index

**Usage**
```r
seg_dissim(plans, shp, group_pop, total_pop)
```

**Arguments**
- `plans`: `redist_plans` object or `plans_matrix` where each row indicates a district assignment and each column is a plan
- `shp`: `redist_map` object, tibble, or data frame containing other columns
- `group_pop`: unquoted name of column in `shp` with group population
- `total_pop`: unquoted name of column in `shp` with total population
Value

numeric vector

References


Examples

data(nh)
data(nh_m)
# For a single plan:
seg_dissim(plans = nh$r_2020, shp = nh, group_pop = vap_hisp, total_pop = vap)

# Or many plans:
seg_dissim(plans = nh_m[, 3:5], shp = nh, group_pop = vap_hisp, total_pop = vap)

splits_admin
Compute Number of Administrative Units Split

Description

Compute Number of Administrative Units Split

Usage

splits_admin(plans, shp, admin)

Arguments

plans redist_plans object or plans_matrix where each row indicates a district assignment and each column is a plan
shp redist_map object, tibble, or data frame containing other columns
admin unquoted name of column in shp with numeric identifiers for administrative units

Value

numeric vector
Examples

```r
data(nh)
data(nh_m)
# For a single plan:
splits_admin(plans = nh$r_2020, shp = nh, admin = county)

# Or many plans:
splits_admin(plans = nh_m[, 3:5], shp = nh, admin = county)
```

---

`splits_count`  
*Count the Number of Splits in Each Administrative Unit*

Description

Tallies the number of unique administrative unit-districts. An unsplit administrative unit will return an entry of 1, while each additional administrative unit-district adds 1.

Usage

`splits_count(plans, shp, admin)`

Arguments

- `plans` redist_plans object or plans_matrix where each row indicates a district assignment and each column is a plan
- `shp` redist_map object, tibble, or data frame containing other columns
- `admin` unquoted name of column in shp with numeric identifiers for administrative units

Value

numeric matrix

Examples

```r
data(nh)
data(nh_m)
# For a single plan:
splits_count(plans = nh$r_2020, shp = nh, admin = county)

# Or many plans:
splits_count(plans = nh_m[, 3:5], shp = nh, admin = county)
```
splits_multi

Compute Number of Administrative Units Split More than Once

Description

Compute Number of Administrative Units Split More than Once

Usage

splits_multi(plans, shp, admin)

Arguments

- **plans**: redist_plans object or plans_matrix where each row indicates a district assignment and each column is a plan
- **shp**: redist_map object, tibble, or data frame containing other columns
- **admin**: unquoted name of column in shp with numeric identifiers for administrative units

Value

numeric vector

Examples

```r
data(nh)
data(nh_m)
# For a single plan:
splits_multi(plans = nh$r_2020, shp = nh, admin = county)

# Or many plans:
splits_multi(plans = nh_m[, 3:5], shp = nh, admin = county)
```

splits_sub_admin

Compute Number of Sub-Administrative Units Split

Description

Compute Number of Sub-Administrative Units Split

Usage

splits_sub_admin(plans, shp, sub_admin)
splits_total

Arguments

- **plans**: redist_plans object or plans_matrix where each row indicates a district assignment and each column is a plan
- **shp**: redist_map object, tibble, or data frame containing other columns
- **sub_admin**: unquoted name of column in shp with numeric identifiers for subsidiary administrative units

Value

numeric vector

Examples

```r
data(nh)
data(nh_m)
# For a single plan:
splits_sub_admin(plans = nh$r_2020, shp = nh, sub_admin = county)

# Or many plans:
splits_sub_admin(plans = nh_m[, 3:5], shp = nh, sub_admin = county)
```

---

splits_total  
*Count the Total Splits in Each Plan*

Description

Counts the total number of administrative splits.

Usage

```r
splits_total(plans, shp, admin)
```

Arguments

- **plans**: redist_plans object or plans_matrix where each row indicates a district assignment and each column is a plan
- **shp**: redist_map object, tibble, or data frame containing other columns
- **admin**: unquoted name of column in shp with numeric identifiers for administrative units

Value

numeric matrix
**Examples**

```r
data(nh)
data(nh_m)
# For a single plan:
splits_total(plans = nh$r_2020, shp = nh, admin = county)

# Or many plans:
splits_total(plans = nh_m[, 3:5], shp = nh, admin = county)
```
Index

* compactness
  - comp_bc, 4
  - comp_box_reock, 5
  - comp_ch, 6
  - comp_edges_rem, 6
  - comp_fh, 7
  - comp_frac_kept, 8
  - comp_log_st, 9
  - comp_lw, 10
  - comp_polsby, 11
  - comp_reock, 12
  - comp_schwartz, 13
  - comp_skew, 14
  - comp_x_sym, 15
  - comp_y_sym, 16

* competitiveness
  - compet_talisman, 3

* data
  - nh, 21
  - nh_m, 23
  - nh_map, 23
  - nh_plans, 25

* distances
  - dist_euc, 17
  - dist_ham, 17
  - dist_info, 18
  - dist_man, 19

* incumbent
  - inc_pairs, 19

* partisan
  - part_bias, 26
  - part_decl, 27
  - part_decl_simple, 28
  - part_dseats, 29
  - part_dvs, 29
  - part_egap, 30
  - part_egap_ep, 31
  - part_lopWins, 32
  - part_mean_median, 33
  - part_resp, 34
  - part_rmd, 35
  - part_sscd, 36
  - part_tau_gap, 37

* segregation
  - seg_dissim, 38

* splits
  - splits_admin, 39
  - splits_count, 40
  - splits_multi, 41
  - splits_sub_admin, 41
  - splits_total, 42

by_plan, 3

  - comp_bc, 4
  - comp_box_reock, 5
  - comp_ch, 6
  - comp_edges_rem, 6
  - comp_fh, 7
  - comp_frac_kept, 8
  - comp_log_st, 9
  - comp_lw, 10
  - comp_polsby, 11
  - comp_reock, 12
  - comp_schwartz, 13
  - comp_skew, 14
  - comp_x_sym, 15
  - comp_y_sym, 16
  - compet_talisman, 3

  - dist_euc, 17
  - dist_ham, 17
  - dist_info, 18
  - dist_man, 19

inc_pairs, 19

list_fn, 20

nh, 21
nh_m, 23
nh_map, 23
nh_plans, 25

part_bias, 26
part_decl, 27
part_decl_simple, 28
part_dseats, 29
part_dvs, 29
part_egap, 30
part_egap_ep, 31
part_lap_wins, 32
part_mean_median, 33
part_resp, 34
part_rmd, 35
part_sscd, 36
part_tau_gap, 37
prep_perims, 38

seg_dissim, 38
splits_admin, 39
splits_count, 40
splits_multi, 41
splits_sub_admin, 41
splits_total, 42