

# Package ‘redistmetrics’

May 9, 2026

**Title** Redistricting Metrics

**Version** 1.0.11

**Date** 2025-12-15

**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 (>= 4.1.0)

**Imports** sf, Rcpp, vctrs, cli, foreach, doParallel, magrittr, dplyr, rlang, geos, wk, libgeos

**Suggests** rmarkdown, knitr, testthat (>= 3.0.0), ggplot2

**LinkingTo** Rcpp, RcppArmadillo, RcppThread, libgeos

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** true

**SystemRequirements** C++17

**RoxygenNote** 7.3.3

**URL** <https://alarm-redist.org/redistmetrics/>,  
<https://github.com/alarm-redist/redistmetrics>

**BugReports** <https://github.com/alarm-redist/redistmetrics/issues>

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**NeedsCompilation** yes

**Author** Christopher T. Kenny [aut, cre] (ORCID:  
<<https://orcid.org/0000-0002-9386-6860>>),  
Cory McCartan [aut],  
Ben Fifield [aut],  
Kosuke Imai [aut]

**Maintainer** Christopher T. Kenny <ctkenny@proton.me>

**Repository** CRAN

**Date/Publication** 2025-12-15 16:30:02 UTC

## Contents

by_plan . . . . .	3
compet_talisman . . . . .	4
comp_bbox_reock . . . . .	5
comp_bc . . . . .	5
comp_box_reock . . . . .	6
comp_ch . . . . .	7
comp_edges_rem . . . . .	8
comp_fh . . . . .	9
comp_frac_kept . . . . .	10
comp_log_st . . . . .	11
comp_lw . . . . .	12
comp_polsby . . . . .	13
comp_reock . . . . .	14
comp_schwartz . . . . .	15
comp_skew . . . . .	16
comp_x_sym . . . . .	17
comp_y_sym . . . . .	18
dist_euc . . . . .	19
dist_ham . . . . .	19
dist_info . . . . .	20
dist_man . . . . .	21
inc_pairs . . . . .	21
list_fn . . . . .	22
nh . . . . .	23
nh_m . . . . .	25
nh_map . . . . .	25
nh_plans . . . . .	27
part_bias . . . . .	28
part_decl . . . . .	29
part_decl_simple . . . . .	30
part_dil_asym . . . . .	31
part_dislocation . . . . .	32
part_dseats . . . . .	33
part_dvs . . . . .	34
part_egap . . . . .	34
part_egap_ep . . . . .	35
part_lop_wins . . . . .	36
part_mean_median . . . . .	37
part_resp . . . . .	38
part_rmd . . . . .	39
part_sscd . . . . .	40

*by\_plan* 3

part_tau_gap . . . . .	41
prep_perims . . . . .	42
seg_dissim . . . . .	42
splits_admin . . . . .	43
splits_count . . . . .	44
splits_district_fuzzy . . . . .	45
splits_multi . . . . .	46
splits_sub_admin . . . . .	46
splits_sub_count . . . . .	47
splits_sub_total . . . . .	48
splits_total . . . . .	49
tally . . . . .	49

**Index** 51

---

*by\_plan*                      *Shorten District by Plan vector*

---

## Description

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

## Usage

```
by_plan(x, ndists)
```

## Arguments

<i>x</i>	summary statistic at the district level
<i>ndists</i>	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**

<code>plans</code>	A <code>redist_plans</code> object or <code>plans_matrix</code> where each row indicates a district assignment and each column is a plan.
<code>shp</code>	A <code>redist_map</code> object, tibble, or data frame containing other columns.
<code>rvote</code>	Unquoted name of column in <code>shp</code> with group population.
<code>dvote</code>	Unquoted name of column in <code>shp</code> with total population.
<code>alpha</code>	Numeric scaling value
<code>beta</code>	Numeric scaling value

**Value**

A numeric vector. Can be shaped into a district-by-plan matrix.

**References**

Wendy K. Tam Cho and Yan Y. Liu Toward a Talismanic Redistricting Tool. *Election Law Journal*. 15, 4. Pp. 351-366.

**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_bbox_reock	<i>Calculate Bounding Box Reock Compactness</i>
-----------------	---

---

**Description**

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

**Usage**

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

**Arguments**

plans	A <code>redist_plans</code> object or <code>plans_matrix</code> where each row indicates a district assignment and each column is a plan.
shp	A <code>redist_map</code> object, tibble, or data frame with an <code>sf</code> geometry column.
epsg	Numeric EPSG code to use to project the shapefile, if needed. Default is 3857.
ncores	Integer number of cores to use. Default is 1.

**Value**

A numeric vector. Can be shaped into a district-by-plan matrix.

**Examples**

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

# Or many plans:
comp_bbox_reock(plans = nh_m[, 1:5], shp = nh)
```

---

comp_bc	<i>Calculate Boyce Clark Ratio</i>
---------	------------------------------------

---

**Description**

Calculate Boyce Clark Ratio

**Usage**

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

**Arguments**

plans	A <code>redist_plans</code> object or <code>plans_matrix</code> where each row indicates a district assignment and each column is a plan.
shp	A <code>redist_map</code> object, tibble, or data frame with an <code>sf</code> geometry column.
epsg	Numeric EPSG code to use to project the shapefile, if needed. Default is 3857.
ncores	Integer number of cores to use. Default is 1.

**Value**

A numeric vector. Can be shaped into a district-by-plan matrix.

**References**

Boyce, R., & Clark, W. 1964. The Concept of Shape in Geography. *Geographical Review*, 54(4), 561-572.

**Examples**

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

# Or many plans:

# slower, beware!
comp_bc(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	A <code>redist_plans</code> object or <code>plans_matrix</code> where each row indicates a district assignment and each column is a plan.
shp	A <code>redist_map</code> object, tibble, or data frame with an <code>sf</code> geometry column.
epsg	Numeric EPSG code to use to project the shapefile, if needed. Default is 3857.
ncores	Integer number of cores to use. Default is 1.

**Value**

A numeric vector. Can be shaped into a district-by-plan matrix.

**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\_ch

*Calculate Convex Hull Compactness*


---

**Description**

Calculate Convex Hull Compactness

**Usage**

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

**Arguments**

plans	A <code>redist_plans</code> object or <code>plans_matrix</code> where each row indicates a district assignment and each column is a plan.
shp	A <code>redist_map</code> object, tibble, or data frame with an sf geometry column.
epsg	Numeric EPSG code to use to project the shapefile, if needed. Default is 3857.
ncores	Integer number of cores to use. Default is 1.

**Value**

A numeric vector. Can be shaped into a district-by-plan matrix.

**Examples**

```
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)
```

---

comp_edges_rem	<i>Calculate Edges Removed Compactness</i>
----------------	--

---

### Description

Calculate Edges Removed Compactness

### Usage

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

### Arguments

plans	A <code>redist_plans</code> object or <code>plans_matrix</code> where each row indicates a district assignment and each column is a plan.
shp	A <code>redist_map</code> object, tibble, or data frame with an <code>sf</code> geometry column.
adj	Zero-indexed adjacency list. Not required if a <code>redist_map</code> is supplied for <code>shp</code> .

### Value

A numeric vector. Can be shaped into a district-by-plan matrix.

### 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

```
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)
```

---

comp_fh	<i>Calculate Fryer Holden Compactness</i>
---------	---

---

### Description

Calculate Fryer Holden Compactness

### Usage

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

### Arguments

plans	A <code>redist_plans</code> object or <code>plans_matrix</code> where each row indicates a district assignment and each column is a plan.
shp	A <code>redist_map</code> object, tibble, or data frame with an <code>sf</code> geometry column.
total_pop	A numeric vector with the population for every observation.
epsg	Numeric EPSG code to use to project the shapefile, if needed. Default is 3857.
ncores	TRUE

### Value

A numeric vector. Can be shaped into a district-by-plan matrix.

### References

Fryer R, Holden R. 2011. Measuring the Compactness of Political Districting Plans. *Journal of Law and Economics*.

### 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	<i>Calculate Fraction Kept Compactness</i>
----------------	--

---

**Description**

Calculate Fraction Kept Compactness

**Usage**

```
comp_frac_kept(plans, shp, adj)
```

**Arguments**

plans	A <code>redist_plans</code> object or <code>plans_matrix</code> where each row indicates a district assignment and each column is a plan.
shp	A <code>redist_map</code> object, tibble, or data frame with an sf geometry column.
adj	Zero-indexed adjacency list. Not required if a <code>redist_map</code> is supplied for <code>shp</code> .

**Value**

A numeric vector. Can be shaped into a district-by-plan matrix.

**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**

```
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	<i>Calculate Log Spanning Tree Compactness</i>
-------------	--

---

**Description**

Calculate Log Spanning Tree Compactness

**Usage**

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

**Arguments**

plans	A <code>redist_plans</code> object or <code>plans_matrix</code> where each row indicates a district assignment and each column is a plan.
shp	A <code>redist_map</code> object, tibble, or data frame with an <code>sf</code> geometry column.
counties	column name in <code>shp</code> containing counties
adj	Zero-indexed adjacency list. Not required if a <code>redist_map</code> is supplied for <code>shp</code> .

**Value**

A numeric vector. Can be shaped into a district-by-plan matrix.

**References**

Cory McCartan and Kosuke Imai. 2020. Sequential Monte Carlo for Sampling Balanced and Compact Redistricting Plans.

**Examples**

```
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**

<code>plans</code>	A <code>redist_plans</code> object or <code>plans_matrix</code> where each row indicates a district assignment and each column is a plan.
<code>shp</code>	A <code>redist_map</code> object, tibble, or data frame with an sf geometry column.
<code>epsg</code>	Numeric EPSG code to use to project the shapefile, if needed. Default is 3857.
<code>ncores</code>	Integer number of cores to use. Default is 1.

**Value**

A numeric vector. Can be shaped into a district-by-plan matrix.

**References**

Harris, Curtis C. 1964. "A scientific method of districting". *Behavioral Science* 3(9), 219–225.

**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)
```

---

`comp_polsby`*Calculate Polsby Popper Compactness*

---

**Description**

Calculate Polsby Popper Compactness

**Usage**

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

**Arguments**

<code>plans</code>	A <code>redist_plans</code> object or <code>plans_matrix</code> where each row indicates a district assignment and each column is a plan.
<code>shp</code>	A <code>redist_map</code> object, tibble, or data frame with an <code>sf</code> geometry column.
<code>use_Rcpp</code>	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.
<code>perim_path</code>	Path to perimeter tibble saved by <code>prep_perims()</code>
<code>perim_df</code>	Tibble of perimeters from <code>prep_perims()</code>
<code>epsg</code>	Numeric EPSG code to use to project the shapefile, if needed. Default is 3857.
<code>ncores</code>	Integer number of cores to use. Default is 1.

**Value**

A numeric vector. Can be shaped into a district-by-plan matrix.

**References**

Cox, E. 1927. A Method of Assigning Numerical and Percentage Values to the Degree of Roundness of Sand Grains. *Journal of Paleontology*, 1(3), 179-183.

Polsby, Daniel D., and Robert D. Popper. 1991. "The Third Criterion: Compactness as a procedural safeguard against partisan gerrymandering." *Yale Law & Policy Review* 9 (2): 301–353.

**Examples**

```

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

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

```

---

comp\_reock

*Calculate Reock Compactness*


---

**Description**

Calculate Reock Compactness

**Usage**

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

**Arguments**

plans	A <code>redist_plans</code> object or <code>plans_matrix</code> where each row indicates a district assignment and each column is a plan.
shp	A <code>redist_map</code> object, tibble, or data frame with an sf geometry column.
epsg	Numeric EPSG code to use to project the shapefile, if needed. Default is 3857.
ncores	Integer number of cores to use. Default is 1.

**Value**

A numeric vector. Can be shaped into a district-by-plan matrix.

**References**

Reock, E. 1961. A Note: Measuring Compactness as a Requirement of Legislative Apportionment. *Midwest Journal of Political Science*, 5(1), 70-74.

**Examples**

```

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	<i>Calculate Schwartzberg Compactness</i>
---------------	---

---

## Description

Calculate Schwartzberg Compactness

## Usage

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

## Arguments

plans	A <code>redist_plans</code> object or <code>plans_matrix</code> where each row indicates a district assignment and each column is a plan.
shp	A <code>redist_map</code> object, tibble, or data frame with an <code>sf</code> geometry column.
use_Rcpp	Logical. Use Rcpp?
perim_path	path to perimeter tibble saved by <code>prep_perims()</code>
perim_df	tibble of perimeters from <code>prep_perims()</code>
epsg	Numeric EPSG code to use to project the shapefile, if needed. Default is 3857.
ncores	Integer number of cores to use. Default is 1.

## Value

A numeric vector. Can be shaped into a district-by-plan matrix.

## References

Schwartzberg, Joseph E. 1966. Reapportionment, Gerrymanders, and the Notion of Compactness. *Minnesota Law Review*. 1701.

## Examples

```
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	A <code>redist_plans</code> object or <code>plans_matrix</code> where each row indicates a district assignment and each column is a plan.
shp	A <code>redist_map</code> object, tibble, or data frame with an <code>sf</code> geometry column.
epsg	Numeric EPSG code to use to project the shapefile, if needed. Default is 3857.
ncores	Integer number of cores to use. Default is 1.

## Value

A numeric vector. Can be shaped into a district-by-plan matrix.

## References

S.N. Schumm. 1963. Sinuosity of alluvial rivers on the Great Plains. *Bulletin of the Geological Society of America*, 74. 1089-1100.

## 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**

<code>plans</code>	A <code>redist_plans</code> object or <code>plans_matrix</code> where each row indicates a district assignment and each column is a plan.
<code>shp</code>	A <code>redist_map</code> object, tibble, or data frame with an sf geometry column.
<code>epsg</code>	Numeric EPSG code to use to project the shapefile, if needed. Default is 3857.
<code>ncores</code>	Integer number of cores to use. Default is 1.

**Value**

A numeric vector. Can be shaped into a district-by-plan matrix.

**References**

Aaron Kaufman, Gary King, and Mayya Komisarchik. 2021. How to Measure Legislative District Compactness If You Only Know it When You See It. *American Journal of Political Science*. 65, 3. Pp. 533-550.

**Examples**

```
#' 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)
```

---

`comp_y_sym`*Calculate Y Symmetry Compactness*

---

## Description

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

## Usage

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

## Arguments

<code>plans</code>	A <code>redist_plans</code> object or <code>plans_matrix</code> where each row indicates a district assignment and each column is a plan.
<code>shp</code>	A <code>redist_map</code> object, tibble, or data frame with an <code>sf</code> geometry column.
<code>epsg</code>	Numeric EPSG code to use to project the shapefile, if needed. Default is 3857.
<code>ncores</code>	Integer number of cores to use. Default is 1.

## Value

A numeric vector. Can be shaped into a district-by-plan matrix.

## References

Aaron Kaufman, Gary King, and Mayya Komisarchik. 2021. How to Measure Legislative District Compactness If You Only Know it When You See It. *American Journal of Political Science*. 65, 3. Pp. 533-550.

## Examples

```
#' 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	<i>Calculate Euclidean Distances</i>
----------	--------------------------------------

---

**Description**

Calculate Euclidean Distances

**Usage**

```
dist_euc(plans, ncores = 1)
```

**Arguments**

plans	A <code>redist_plans</code> object or <code>plans_matrix</code> where each row indicates a district assignment and each column is a plan.
ncores	Integer number of cores to use. Default is 1.

**Value**

matrix of plan distances

**Examples**

```
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	<i>Calculate Hamming Distances</i>
----------	------------------------------------

---

**Description**

Calculate Hamming Distances

**Usage**

```
dist_ham(plans, ncores = 1)
```

**Arguments**

plans	A <code>redist_plans</code> object or <code>plans_matrix</code> where each row indicates a district assignment and each column is a plan.
ncores	Integer number of cores to use. Default is 1.

**Value**

matrix of plan distances

**Examples**

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

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

---

dist\_info

*Calculate Variation of Information Distances*

---

**Description**

Calculate Variation of Information Distances

**Usage**

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

**Arguments**

plans	A <code>redist_plans</code> object or <code>plans_matrix</code> where each row indicates a district assignment and each column is a plan.
shp	A <code>redist_map</code> object, tibble, or data frame containing other columns.
total_pop	Unquoted name of column in <code>shp</code> with total population.
ncores	Integer number of cores to use. Default is 1.

**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, shp = nh, total_pop = pop)

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

---

dist_man	<i>Calculate Manhattan Distances</i>
----------	--------------------------------------

---

**Description**

Calculate Manhattan Distances

**Usage**

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

**Arguments**

plans	A <code>redist_plans</code> object or <code>plans_matrix</code> where each row indicates a district assignment and each column is a plan.
ncores	Integer number of cores to use. Default is 1.

**Value**

matrix of plan distances

**Examples**

```
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	<i>Count Incumbent Pairings</i>
-----------	---------------------------------

---

**Description**

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

**Usage**

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

**Arguments**

plans	A redist_plans object or plans_matrix where each row indicates a district assignment and each column is a plan.
shp	A 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_')
```

---

nh

*New Hampshire Election and Demographic Data*

---

### 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

- Voting and Election Science Team, 2020, "2020 Precinct-Level Election Results", <https://doi.org/10.7910/DVN/K7760H>, Harvard Dataverse, V23
- Voting and Election Science Team, 2018, "2016 Precinct-Level Election Results", <https://doi.org/10.7910/DVN/NH5S2I>, Harvard Dataverse, V71
- Voting and Election Science Team, 2019, "2018 Precinct-Level Election Results", <https://doi.org/10.7910/DVN/UBKYRU>, Harvard Dataverse, V48
- Kenny & McCartan (2021, Aug. 10). ALARM Project: 2020 Redistricting Data Files. Retrieved from <https://github.com/alarm-redist/census-2020/>

## Examples

```
data(nh)
```

---

nh_m	<i>Redistricting Plans for New Hampshire as matrix</i>
------	--

---

**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_m")
```

**Format**

A matrix with 52 columns and 326 rows where each column is a plan

**Examples**

```
data(nh_m)
```

---

nh_map	<i>New Hampshire Election and Demographic Data as a redist_map</i>
--------	--

---

**Description**

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

**Usage**

```
data("nh_map")
```

**Format**

A redist\_map 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
- 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

- Voting and Election Science Team, 2020, "2020 Precinct-Level Election Results", <https://doi.org/10.7910/DVN/K7760H>, Harvard Dataverse, V23
- Voting and Election Science Team, 2018, "2016 Precinct-Level Election Results", <https://doi.org/10.7910/DVN/NH5S2I>, Harvard Dataverse, V71
- Voting and Election Science Team, 2019, "2018 Precinct-Level Election Results", <https://doi.org/10.7910/DVN/UBKYRU>, Harvard Dataverse, V48
- Kenny & McCartan (2021, Aug. 10). ALARM Project: 2020 Redistricting Data Files. Retrieved from <https://github.com/alarm-redist/census-2020/>

## Examples

```
data(nh_map)
```

---

nh\_plans

*Redistricting Plans for New Hampshire as* redist\_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**

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

**Value**

A numeric vector. Can be shaped into a district-by-plan matrix.

**References**

Jonathan N. Katz, Gary King, and Elizabeth Rosenblatt. 2020. Theoretical Foundations and Empirical Evaluations of Partisan Fairness in District-Based Democracies. *American Political Science Review*, 114, 1, Pp. 164-178.

**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	<i>Calculate Declination</i>
-----------	------------------------------

---

**Description**

Calculate Declination

**Usage**

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

**Arguments**

plans	A redist_plans object or plans_matrix where each row indicates a district assignment and each column is a plan.
shp	A 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**

A numeric vector. Can be shaped into a district-by-plan matrix.

**References**

Gregory S. Warrington. 2018. "Quantifying Gerrymandering Using the Vote Distribution." Election Law Journal: Rules, Politics, and Policy. Pp. 39-57.<http://doi.org/10.1089/elj.2017.0447>

**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

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

### Arguments

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

### Value

A numeric vector. Can be shaped into a district-by-plan matrix.

### References

Jonathan N. Katz, Gary King, and Elizabeth Rosenblatt. 2020. Theoretical Foundations and Empirical Evaluations of Partisan Fairness in District-Based Democracies. *American Political Science Review*, 114, 1, Pp. 164-178.

### Examples

```
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_dil_asym	<i>Calculate Dilution Asymmetry</i>
---------------	-------------------------------------

---

**Description**

Calculate Dilution Asymmetry

**Usage**

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

**Arguments**

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

**Value**

A numeric vector. Can be shaped into a district-by-plan matrix.

**References**

Sanford C. Gordon and Sidak Yntiso. 2024. Base Rate Neglect and the Diagnosis of Partisan Gerrymanders. *Election Law Journal: Rules, Politics, and Policy*. doi:10.1089/elj.2023.0005.

**Examples**

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

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

---

part_dislocation	<i>Calculate Partisan Dislocation</i>
------------------	---------------------------------------

---

### Description

Calculate Partisan Dislocation

### Usage

```
part_dislocation(
  plans,
  shp,
  dvote,
  rvote,
  total_pop = dvote + rvote,
  epsg = 3857,
  by_precinct = FALSE,
  signed = TRUE
)
```

### Arguments

plans	A <code>redist_plans</code> object or <code>plans_matrix</code> where each row indicates a district assignment and each column is a plan.
shp	A <code>redist_map</code> object, tibble, or data frame with an <code>sf</code> geometry column.
dvote	Unquoted name of column in <code>shp</code> with total population.
rvote	Unquoted name of column in <code>shp</code> with group population.
total_pop	Unquoted name of column in <code>shp</code> with total population.
epsg	Numeric EPSG code to use to project the shapefile, if needed. Default is 3857.
by_precinct	Defaults to <code>FALSE</code> , returning district-level values.
signed	Defaults to <code>TRUE</code> . Should the output for district-level values be signed? Setting to <code>TRUE</code> returns precinct-level values.

### Value

A numeric vector. Can be shaped into a district-by-plan matrix.

### References

DeFord, D. R., Eubank, N., & Rodden, J. (2022). Partisan dislocation: A precinct-level measure of representation and gerrymandering. *Political Analysis*, 30(3), 403-425.

**Examples**

```

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

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

```

---

part_dseats	<i>Calculate Democratic Seats</i>
-------------	-----------------------------------

---

**Description**

Calculate Democratic Seats

**Usage**

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

**Arguments**

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

**Value**

A numeric vector. Can be shaped into a district-by-plan matrix.

**Examples**

```

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)

```

---

part_dvs	<i>Calculate Democratic Vote Share</i>
----------	--

---

**Description**

Calculate Democratic Vote Share

**Usage**

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

**Arguments**

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

**Value**

A numeric vector. Can be shaped into a district-by-plan matrix.

**Examples**

```
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	<i>Calculate Efficiency Gap</i>
-----------	---------------------------------

---

**Description**

Calculate Efficiency Gap

**Usage**

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

**Arguments**

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

**Value**

A numeric vector. Can be shaped into a district-by-plan matrix.

**References**

Nicholas O. Stephanopoulos. 2015. Partisan Gerrymandering and the Efficiency Gap. *The University of Chicago Law Review*, 82, Pp. 831-900.

**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	A <code>redist_plans</code> object or <code>plans_matrix</code> where each row indicates a district assignment and each column is a plan.
shp	A <code>redist_map</code> object, tibble, or data frame containing other columns.
dvote	Unquoted name of column in <code>shp</code> with total population.
rvote	Unquoted name of column in <code>shp</code> with group population.

**Value**

A numeric vector. Can be shaped into a district-by-plan matrix.

**References**

Nicholas O. Stephanopoulos. 2015. Partisan Gerrymandering and the Efficiency Gap. The University of Chicago Law Review, 82, Pp. 831-900.

**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	<i>Calculate Lopsided Wins</i>
---------------	--------------------------------

---

**Description**

Calculate Lopsided Wins

**Usage**

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

**Arguments**

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

**Value**

A numeric vector. Can be shaped into a district-by-plan matrix.

**References**

Samuel S.-H. Wang. 2016. "Three Tests for Practical Evaluation of Partisan Gerrymandering." Stanford Law Review, 68, Pp. 1263 - 1321.

**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	<i>Calculate Mean Median Score</i>
------------------	------------------------------------

---

**Description**

Calculate Mean Median Score

**Usage**

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

**Arguments**

plans	A redist_plans object or plans_matrix where each row indicates a district assignment and each column is a plan.
shp	A 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**

A numeric vector. Can be shaped into a district-by-plan matrix.

**References**

Michael D. McDonald and Robin E. Best. 2015. Unfair Partisan Gerrymanders in Politics and Law: A Diagnostic Applied to Six Cases. *Election Law Journal: Rules, Politics, and Policy*. 14. 4. Pp. 312-330.

**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)
```

---

part\_resp

*Calculate Responsiveness*

---

## Description

Calculate Responsiveness

## Usage

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

## Arguments

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

## Value

A numeric vector. Can be shaped into a district-by-plan matrix.

## References

Jonathan N. Katz, Gary King, and Elizabeth Rosenblatt. 2020. Theoretical Foundations and Empirical Evaluations of Partisan Fairness in District-Based Democracies. *American Political Science Review*, 114, 1, Pp. 164-178.

## 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	<i>Calculate Ranked Marginal Deviation</i>
----------	--

---

**Description**

Calculate Ranked Marginal Deviation

**Usage**

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

**Arguments**

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

**Value**

A numeric vector. Can be shaped into a district-by-plan matrix.

**References**

Gregory Herschlag, Han Sung Kang, Justin Luo, Christy Vaughn Graves, Sachet Bangia, Robert Ravier & Jonathan C. Mattingly (2020) Quantifying Gerrymandering in North Carolina, *Statistics and Public Policy*, 7:1, 30-38, DOI: 10.1080/2330443X.2020.1796400

**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**

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

**Arguments**

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

**Value**

A numeric vector. Can be shaped into a district-by-plan matrix.

**References**

Gregory Herschlag, Han Sung Kang, Justin Luo, Christy Vaughn Graves, Sachet Bangia, Robert Ravier & Jonathan C. Mattingly (2020) Quantifying Gerrymandering in North Carolina, *Statistics and Public Policy*, 7:1, 30-38, DOI: 10.1080/2330443X.2020.1796400

**Examples**

```
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	<i>Calculate Tau Gap</i>
--------------	--------------------------

---

**Description**

Calculate Tau Gap

**Usage**

```
part_tau_gap(plans, shp, dvote, rvote, tau = 1)
```

**Arguments**

plans	A <code>redist_plans</code> object or <code>plans_matrix</code> where each row indicates a district assignment and each column is a plan.
shp	A <code>redist_map</code> object, tibble, or data frame containing other columns.
dvote	Unquoted name of column in <code>shp</code> with total population.
rvote	Unquoted name of column in <code>shp</code> with group population.
tau	A non-negative numeric for calculating Tau Gap. Defaults to 1.

**Value**

A numeric vector. Can be shaped into a district-by-plan matrix.

**References**

Gregory S. Warrington. 2018. "Quantifying Gerrymandering Using the Vote Distribution." *Election Law Journal: Rules, Politics, and Policy*. Pp. 39-57.<http://doi.org/10.1089/elj.2017.0447>

**Examples**

```
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**

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

**Arguments**

<code>shp</code>	A <code>redist_map</code> object, tibble, or data frame with an sf geometry column.
<code>epsg</code>	Numeric EPSG code to use to project the shapefile, if needed. Default is 3857.
<code>perim_path</code>	A path to save an rds
<code>ncores</code>	Integer number of cores to use. Default is 1.

**Value**

tibble of perimeters and lengths

**Examples**

```
data(nh)
prep_perims(nh)
```

---

```
seg_dissim          Compute Dissimilarity Index
```

---

**Description**

Compute Dissimilarity Index

**Usage**

```
seg_dissim(plans, shp, group_pop, total_pop)
```

**Arguments**

<code>plans</code>	A <code>redist_plans</code> object or <code>plans_matrix</code> where each row indicates a district assignment and each column is a plan.
<code>shp</code>	A <code>redist_map</code> object, tibble, or data frame containing other columns.
<code>group_pop</code>	Unquoted name of column in <code>shp</code> with group population.
<code>total_pop</code>	Unquoted name of column in <code>shp</code> with total population.

**Value**

A numeric vector. Can be shaped into a district-by-plan matrix.

**References**

Douglas Massey and Nancy Denton. 1987. The Dimensions of Social Segregation. Social Forces.

**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	A <code>redist_plans</code> object or <code>plans_matrix</code> where each row indicates a district assignment and each column is a plan.
shp	A <code>redist_map</code> object, tibble, or data frame containing other columns.
admin	Unquoted name of column in <code>shp</code> with numeric identifiers for administrative units.

**Value**

A numeric vector. Can be shaped into a district-by-plan matrix.

**Examples**

```

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	A <code>redist_plans</code> object or <code>plans_matrix</code> where each row indicates a district assignment and each column is a plan.
shp	A <code>redist_map</code> object, tibble, or data frame containing other columns.
admin	Unquoted name of column in <code>shp</code> with numeric identifiers for administrative units.

**Value**

numeric matrix

**Examples**

```

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\_district\_fuzzy *Fuzzy Splits by District (Experimental)*

---

## Description

Not all relevant geographies nest neatly into Census blocks, including communities of interest or neighborhood. For these cases, this provides a tabulation by district of the number of splits. As some geographies can be split multiple times, the sum of these splits may not reflect the total number of splits.

## Usage

```
splits_district_fuzzy(plans, shp, nbr, thresh = 0.01, epsg)
```

## Arguments

plans	A <code>redist_plans</code> object or <code>plans_matrix</code> where each row indicates a district assignment and each column is a plan.
shp	A <code>redist_map</code> object, tibble, or data frame with an <code>sf</code> geometry column.
nbr	Geographic neighborhood, community, or other unit to check splits for.
thresh	Percent as decimal of an area to trim away. Default is .01, which is 1%.
epsg	Numeric EPSG code to use to project the shapefile, if needed. Default is 3857.

## Details

Beware, this requires a `nbr` shape input and will be slower than checking splits in cases where administrative unit nests cleanly into the geographies represented by `shp`.

## Value

numeric matrix

## Examples

```
data(nh)
data(nh_m)

# toy example,
# suppose we care about the splits of the counties and they don't nest
nh_cty <- nh %>% dplyr::group_by(county) %>% dplyr::summarize()

# For a single plan:
splits_district_fuzzy(plans = nh$r_2020, shp = nh, nbr = nh_cty)

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

---

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

**Value**

A numeric vector. Can be shaped into a district-by-plan matrix.

**Examples**

```
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)
```

**Arguments**

plans	A <code>redist_plans</code> object or <code>plans_matrix</code> where each row indicates a district assignment and each column is a plan.
shp	A <code>redist_map</code> object, tibble, or data frame containing other columns.
sub_admin	Unquoted name of column in <code>shp</code> with numeric identifiers for subsidiary administrative units.

**Value**

A numeric vector. Can be shaped into a district-by-plan matrix.

**Examples**

```
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_sub_count	<i>Count the Number of Splits in Each Sub-Administrative Unit</i>
------------------	---

---

**Description**

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

**Usage**

```
splits_sub_count(plans, shp, sub_admin)
```

**Arguments**

plans	A <code>redist_plans</code> object or <code>plans_matrix</code> where each row indicates a district assignment and each column is a plan.
shp	A <code>redist_map</code> object, tibble, or data frame containing other columns.
sub_admin	Unquoted name of column in <code>shp</code> with numeric identifiers for subsidiary administrative units.

**Value**

numeric matrix

**Examples**

```

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

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

```

---

splits_sub_total	<i>Count the Total Sub-Administrative Unit in Each Plan</i>
------------------	---

---

**Description**

Counts the total number of sub-administrative splits.

**Usage**

```
splits_sub_total(plans, shp, sub_admin)
```

**Arguments**

plans	A <code>redist_plans</code> object or <code>plans_matrix</code> where each row indicates a district assignment and each column is a plan.
shp	A <code>redist_map</code> object, tibble, or data frame containing other columns.
sub_admin	Unquoted name of column in <code>shp</code> with numeric identifiers for subsidiary administrative units.

**Value**

numeric matrix

**Examples**

```

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

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

```

---

splits_total	<i>Count the Total Splits in Each Plan</i>
--------------	--

---

**Description**

Counts the total number of administrative splits.

**Usage**

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

**Arguments**

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

**Value**

numeric matrix

**Examples**

```
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)
```

---

tally	<i>Tally a Column by District</i>
-------	-----------------------------------

---

**Description**

Helper function to aggregate a vector by district. Can be used to calculate total population, group percentages, and more.

**Usage**

```
tally(plans, shp, x)
```

**Arguments**

plans	A <code>redist_plans</code> object or <code>plans_matrix</code> where each row indicates a district assignment and each column is a plan.
shp	A <code>redist_map</code> object, tibble, or data frame containing other columns.
x	The numeric vector to tally.

**Value**

A numeric vector with the tallies. Can be shaped into a district-by-plan matrix.

**Examples**

```
data(nh)
data(nh_m)
```

```
tally(nh_m, nh, pop) # total population
tally(nh_m, nh, vap_hisp) / tally(nh_m, nh, vap) # HVAP
```

# Index

- \* **compactness**
  - comp\_bbox\_reock, 5
  - comp\_bc, 5
  - comp\_box\_reock, 6
  - comp\_ch, 7
  - comp\_edges\_rem, 8
  - comp\_fh, 9
  - comp\_frac\_kept, 10
  - comp\_log\_st, 11
  - comp\_lw, 12
  - comp\_polsby, 13
  - comp\_reock, 14
  - comp\_schwartz, 15
  - comp\_skew, 16
  - comp\_x\_sym, 17
  - comp\_y\_sym, 18
  - prep\_perims, 42
- \* **competitiveness**
  - compet\_talisman, 4
- \* **data**
  - nh, 23
  - nh\_m, 25
  - nh\_map, 25
  - nh\_plans, 27
- \* **distances**
  - dist\_euc, 19
  - dist\_ham, 19
  - dist\_info, 20
  - dist\_man, 21
- \* **incumbent**
  - inc\_pairs, 21
- \* **partisan**
  - part\_bias, 28
  - part\_decl, 29
  - part\_decl\_simple, 30
  - part\_dil\_asym, 31
  - part\_dislocation, 32
  - part\_dseats, 33
  - part\_dvs, 34
  - part\_egap, 34
  - part\_egap\_ep, 35
  - part\_lop\_wins, 36
  - part\_mean\_median, 37
  - part\_resp, 38
  - part\_rmd, 39
  - part\_sscd, 40
  - part\_tau\_gap, 41
- \* **segregation**
  - seg\_dissim, 42
- \* **splits**
  - splits\_admin, 43
  - splits\_count, 44
  - splits\_district\_fuzzy, 45
  - splits\_multi, 46
  - splits\_sub\_admin, 46
  - splits\_sub\_count, 47
  - splits\_sub\_total, 48
  - splits\_total, 49
- by\_plan, 3
- comp\_bbox\_reock, 5
- comp\_bc, 5
- comp\_box\_reock, 6
- comp\_ch, 7
- comp\_edges\_rem, 8
- comp\_fh, 9
- comp\_frac\_kept, 10
- comp\_log\_st, 11
- comp\_lw, 12
- comp\_polsby, 13
- comp\_reock, 14
- comp\_schwartz, 15
- comp\_skew, 16
- comp\_x\_sym, 17
- comp\_y\_sym, 18
- compet\_talisman, 4
- dist\_euc, 19

dist\_ham, 19  
dist\_info, 20  
dist\_man, 21

inc\_pairs, 21

list\_fn, 22

nh, 23  
nh\_m, 25  
nh\_map, 25  
nh\_plans, 27

part\_bias, 28  
part\_decl, 29  
part\_decl\_simple, 30  
part\_dil\_asym, 31  
part\_dislocation, 32  
part\_dseats, 33  
part\_dvs, 34  
part\_egap, 34  
part\_egap\_ep, 35  
part\_lop\_wins, 36  
part\_mean\_median, 37  
part\_resp, 38  
part\_rmd, 39  
part\_sscd, 40  
part\_tau\_gap, 41  
prep\_perims, 42

seg\_dissim, 42  
splits\_admin, 43  
splits\_count, 44  
splits\_district\_fuzzy, 45  
splits\_multi, 46  
splits\_sub\_admin, 46  
splits\_sub\_count, 47  
splits\_sub\_total, 48  
splits\_total, 49

tally, 49