Package 'redist'

October 14, 2020

Version 2.0.2

Date 2020-10-03

Title Simulation Methods for Legislative Redistricting

Maintainer Ben Fifield <benfifield@gmail.com>

Description Enables researchers to sample redistricting plans from a pre-specified target distribution using Sequential Monte Carlo and Markov Chain Monte Carlo algorithms. The package allows for the implementation of various constraints in the redistricting process such as geographic compactness and population parity requirements. Tools for analysis such as computation of various summary statistics and plotting functionality are also included. The package implements methods described in Fifield, Higgins, Imai and Tarr (2020) <doi: 10.1080/10618600.2020.1739532>, Fifield, Imai, Kawahara, and Kenny (2020) <doi: 10.1080/2330443X.2020.1791773>, and McCartan and Imai (2020) <arXiv: 2008.06131>.

Depends R (>= 3.5.0)

Imports Rcpp (>= 0.11.0), spdep, sp, sf, coda, parallel, doParallel, foreach, lwgeom, dplyr, ggplot2, magrittr, readr, servr, sys, tibble, stringr

Suggests testthat, Rmpi, knitr, rmarkdown, igraph

LinkingTo Rcpp, RcppArmadillo, RcppEigen, BH

License GPL (>= 2)

SystemRequirements gmp, libxml2, python

NeedsCompilation yes

BugReports https://github.com/kosukeimai/redist/issues

RoxygenNote 7.1.1

VignetteBuilder knitr

Encoding UTF-8

Author Ben Fifield [aut, cre],
Christopher T. Kenny [aut],
Cory McCartan [aut],
Alexander Tarr [aut],
Michael Higgins [ctb],

Jun Kawahara [aut],
Kosuke Imai [aut]

Repository CRAN

Date/Publication 2020-10-13 23:50:07 UTC

R topics documented:

redist-package *Markov Chain Monte Carlo Methods for Redistricting Simulation*

Description

Enables researchers to sample redistricting plans from a pre-specified target distribution using a Markov Chain Monte Carlo algorithm. The package allows for the implementation of various constraints in the redistricting process such as geographic compactness and population parity requirements. The algorithm also can be used in combination with efficient simulation methods such as simulated and parallel tempering algorithms. Tools for analysis such as inverse probability reweighting and plotting functionality are included. The package implements methods described in Fifield, Higgins, Imai and Tarr (2016) "A New Automated Redistricting Simulator Using Markov Chain Monte Carlo," working paper available at <http://imai.fas.harvard.edu/research/files/redist.pdf>.

Details

Package:	redist
Type:	Package
Version:	2.0.2
Date:	2020-10-03
License:	GPL (>= 2)

Author(s)

Benjamin Fifield, Department of Politics, Princeton University <benfifield@gmail.com>, https://www.benfifield.com

Michael Higgins, Department of Statistics, Princeton University <mikehiggins@k-state.edu>, http://www-personal.k-state.edu/~mikehiggins/

Alexander Tarr, Department of Electrical Engineering, Princeton University <atarr@princeton.edu>

Kosuke Imai, Department of Politics, Princeton University <kimai@princeton.edu>, http://imai.fas.harvard.edu

Maintainer: Ben Fifield <benfifield@gmail.com>

References

Barbu, Adrian and Song-Chun Zhu. (2005) "Generalizing Swendsen-Wang to Sampling Arbitrary Posterior Probabilities." IEEE Transactions on Pattern Analysis and Machine Intelligence.

Fifield, Benjamin, Michael Higgins, Kosuke Imai and Alexander Tarr. (2016) "A New Automated Redistricting Simulator Using Markov Chain Monte Carlo." *Working Paper.* Available at http://imai.princeton.edu/research/files/redist.pdf.

Swendsen, Robert and Jian-Sheng Wang. (1987) "Nonuniversal Critical Dynamics in Monte Carlo Simulations." Physical Review Letters.

`algdat.p10` *All Partitions of 25 Precincts into 3 Congressional Districts (10% Population Constraint)*

Description

This data set contains demographic and geographic information about 25 contiguous precincts in the state of Florida. The data lists all possible partitions of the 25 precincts into three contiguous congressional districts, conditional on the congressional districts falling within 10% of population parity.

Usage

```
data("algdat.p10")
```

Format

A list with five entries:

`adjlist` An adjacency list for the 25 precincts.

`cdmat` A matrix containing every partition of the 25 precincts into three contiguous congressional districts, with no population constraint.

`precinct.data` A matrix containing demographic information for each of the 25 precincts.

`segregation.index` A matrix containing the dissimilarity index of segregation (Massey and Denton 1987) for each congressional district map in `cdmat`.

`distancemat` A square matrix containing the squared distance between the centroids of any two precincts.

References

Fifield, Benjamin, Michael Higgins, Kosuke Imai and Alexander Tarr. (2016) "A New Automated Redistricting Simulator Using Markov Chain Monte Carlo." Working Paper. Available at http://imai.princeton.edu/research/files/redist.pdf.

Massey, Douglas and Nancy Denton. (1987) "The Dimensions of Social Segregation". Social Forces.

Examples

```
## Not run:
data(algdat.p10)

## End(Not run)
```

`algdat.p20` *All Partitions of 25 Precincts into 3 Congressional Districts (20% Population Constraint)*

Description

This data set contains demographic and geographic information about 25 contiguous precincts in the state of Florida. The data lists all possible partitions of the 25 precincts into three contiguous congressional districts, conditional on the congressional districts falling within 20% of population parity.

Usage

```
data("algdat.p20")
```

Format

A list with five entries:

`adjlist` An adjacency list for the 25 precincts.

`cdmat` A matrix containing every partition of the 25 precincts into three contiguous congressional districts, with no population constraint.

`precinct.data` A matrix containing demographic information for each of the 25 precincts.

`segregation.index` A matrix containing the dissimilarity index of segregation (Massey and Denton 1987) for each congressional district map in `cdmat`.

`distancemat` A square matrix containing the squared distance between the centroids of any two precincts.

References

Fifield, Benjamin, Michael Higgins, Kosuke Imai and Alexander Tarr. (2016) "A New Automated Redistricting Simulator Using Markov Chain Monte Carlo." Working Paper. Available at http://imai.princeton.edu/research/files/redist.pdf.

Massey, Douglas and Nancy Denton. (1987) "The Dimensions of Social Segregation". Social Forces.

Examples

```
## Not run:
data(algdat.p20)

## End(Not run)
```

`algdat.pfull` *All Partitions of 25 Precincts into 3 Congressional Districts (No Population Constraint)*

Description

This data set contains demographic and geographic information about 25 contiguous precincts in the state of Florida. The data lists all possible partitions of the 25 precincts into three contiguous congressional districts.

Usage

```
data("algdat.pfull")
```

Format

A list with five entries:

`adjlist` An adjacency list for the 25 precincts.

`cdmat` A matrix containing every partition of the 25 precincts into three contiguous congressional districts, with no population constraint.

`precinct.data` A matrix containing demographic information for each of the 25 precincts.

`segregation.index` A matrix containing the dissimilarity index of segregation (Massey and Denton 1987) for each congressional district map in `cdmat`.

`distancemat` A square matrix containing the squared distance between the centroids of any two precincts.

References

Fifield, Benjamin, Michael Higgins, Kosuke Imai and Alexander Tarr. (2016) "A New Automated Redistricting Simulator Using Markov Chain Monte Carlo." Working Paper. Available at http://imai.princeton.edu/research/files/redist.pdf.

Massey, Douglas and Nancy Denton. (1987) "The Dimensions of Social Segregation". Social Forces.

Examples

```
## Not run:
data(algdat.pfull)

## End(Not run)
```

as.matrix.redist *Extract the redistricting matrix from a* redist *object*

Description

Extract the redistricting matrix from a redist object

Usage

```
## S3 method for class 'redist'
as.matrix(x, ...)
```

Arguments

x	redist object
\dots	additional arguments

fl25 *Florida 25 Precinct File*

Description

This data set contains the 25 Precinct shapefile and related data for each precinct.

Usage

```
data("fl25")
```

Format

sf data.frame containing columns for useful data related to the redistricting process, subsetted from real data in Florida, and sf geometry column.

geoid Contains unique identifier for each precinct which can be matched to the full Florida dataset.

pop Contains the population of each precinct.

vap Contains the voting age population of each precinct.

obama Contains the 2012 presidential vote for Obama.

mccain Contains the 2012 presidential vote for McCain.

TotPop Contains the population of each precinct. Identical to pop.

BlackPop Contains the black population of each precinct.

HispPop Contains the Hispanic population of each precinct.

VAP Contains the voting age population of each precinct. Identical to vap.

BlackVAP Contains the voting age population of black constituents of each precinct.

HispVAP Contains the voting age population of hispanic constituents of each precinct.

geometry Contains sf geometry of each precinct.

References

Fifield, Benjamin, Michael Higgins, Kosuke Imai and Alexander Tarr. (2016) "A New Automated Redistricting Simulator Using Markov Chain Monte Carlo." Working Paper. Available at http://imai.princeton.edu/research/files/redist.pdf.

Examples

```
## Not run:
data(fl25)

## End(Not run)
```

`fl250` *Florida 250 Precinct File*

Description

This data set contains the 250 Precinct shapefile and related data for each precinct.

Usage

```
data("fl250")
```

Format

sf data.frame containing columns for useful data related to the redistricting process, subsetted from real data in Florida, and sf geometry column.

`geoid` Contains unique identifier for each precinct which can be matched to the full Florida dataset.

`pop` Contains the population of each precinct.

`vap` Contains the voting age population of each precinct.

`obama` Contains the 2012 presidential vote for Obama.

`mccain` Contains the 2012 presidential vote for McCain.

`TotPop` Contains the population of each precinct. Identical to pop.

`BlackPop` Contains the black population of each precinct.

`HispPop` Contains the Hispanic population of each precinct.

`VAP` Contains the voting age population of each precinct. Identical to vap.

`BlackVAP` Contains the voting age population of black constituents of each precinct.

`HispVAP` Contains the voting age population of hispanic constituents of each precinct.

`geometry` Contains sf geometry of each precinct.

References

Fifield, Benjamin, Michael Higgins, Kosuke Imai and Alexander Tarr. (2016) "A New Automated Redistricting Simulator Using Markov Chain Monte Carlo." Working Paper. Available at http://imai.princeton.edu/research/files/redist.pdf.

Examples

```
## Not run:
data(fl250)

## End(Not run)
```

`fl70` *Florida 70 Precinct File*

Description

This data set contains the 70 Precinct shapefile and related data for each precinct.

Usage

```
data("fl70")
```

Format

sf data.frame containing columns for useful data related to the redistricting process, subsetted from real data in Florida, and sf geometry column.

`geoid` Contains unique identifier for each precinct which can be matched to the full Florida dataset.

`pop` Contains the population of each precinct.

`vap` Contains the voting age population of each precinct.

`obama` Contains the 2012 presidential vote for Obama.

`mccain` Contains the 2012 presidential vote for McCain.

`TotPop` Contains the population of each precinct. Identical to pop.

`BlackPop` Contains the black population of each precinct.

`HispPop` Contains the Hispanic population of each precinct.

`VAP` Contains the voting age population of each precinct. Identical to vap.

`BlackVAP` Contains the voting age population of black constituents of each precinct.

`HispVAP` Contains the voting age population of hispanic constituents of each precinct.

`geometry` Contains sf geometry of each precinct.

References

Fifield, Benjamin, Michael Higgins, Kosuke Imai and Alexander Tarr. (2016) "A New Automated Redistricting Simulator Using Markov Chain Monte Carlo." Working Paper. Available at http://imai.princeton.edu/research/files/redist.pdf.

Examples

```
## Not run:
data(fl70)

## End(Not run)
```

is_last *check if last edge*

Description

check if last edge

Usage

```
is_last(i, v, edges)
```

Arguments

i	integer, current frontier
v	integer, vertex to search for
edges	edgelist matrix

Value

bool

redist.adjacency *Adjacency List functionality for redist*

Description

Creates an adjacency list that is zero indexed with no skips

Usage

```
redist.adjacency(shp)
```

Arguments

shp	A SpatialPolygonsDataFrame or sf object. Required.

Value

Adjacency list

redist.calc.frontier.size *Calculate Frontier Size*

Description

Calculate Frontier Size

Usage

```
redist.calc.frontier.size(ordered_path)
```

Arguments

ordered_path	path to ordered path created by redist.prep.enumpart

Value

List, four objects

- maxnumeric, maximum frontier size
- averagenumeric, average frontier size
- average_sqnumeric, average((frontier size)^2)
- sequencenumeric vector, lists out all sizes for every frontier

Examples

```
## Not run:
data(fl25)
adj <- redist.adjacency(fl25)
redist.prep.enumpart(adj, 'unordered', 'ordered')
redist.calc.frontier.size('ordered')

## End(Not run)
```

redist.combine *Combine successive runs of* redist.mcmc

Description

redist.combine is used to combine successive runs of redist.mcmc into a single data object

Usage

```
redist.combine(savename, nloop, nthin, temper)
```

Arguments

savename	The name (without the loop or .RData suffix) of the saved simulations.
nloop	The number of loops being combined.
nthin	How much to thin the simulations being combined.
temper	Wheterh simulated tempering was used (1) or not (0) in the simulations. Default is 0.

Details

This function allows users to combine multiple successive runs of redist.mcmc into a single redist object for analysis.

Value

redist.combine returns an object of class "redist". The object redist is a list that contains the folowing components (the inclusion of some components is dependent on whether tempering techniques are used):

partitions	Matrix of congressional district assignments generated by the algorithm. Each row corresponds to a geographic unit, and each column corresponds to a simulation.
distance_parity	Vector containing the maximum distance from parity for a particular simulated redistricting plan.
mhdecisions	A vector specifying whether a proposed redistricting plan was accepted (1) or rejected (0) in a given iteration.
mhprob	A vector containing the Metropolis-Hastings acceptance probability for each iteration of the algorithm.
pparam	A vector containing the draw of the p parameter for each simulation, which dictates the number of swaps attempted.
constraint_pop	A vector containing the value of the population constraint for each accepted redistricting plan.
constraint_compact	A vector containing the value of the compactness constraint for each accepted redistricting plan.
constraint_segregation	A vector containing the value of the segregation constraint for each accepted redistricting plan.
constraint_similar	A vector containing the value of the similarity constraint for each accepted redistricting plan.
beta_sequence	A vector containing the value of beta for each iteration of the algorithm. Returned when tempering is being used.
mhdecisions_beta	A vector specifying whether a proposed beta value was accepted (1) or rejected (0) in a given iteration of the algorithm. Returned when tempering is being used.

`mhprob_beta` A vector containing the Metropolis-Hastings acceptance probability for each iteration of the algorithm. Returned when tempering is being used.

References

Fifield, Benjamin, Michael Higgins, Kosuke Imai and Alexander Tarr. (2016) "A New Automated Redistricting Simulator Using Markov Chain Monte Carlo." Working Paper. Available at http://imai.princeton.edu/research/files/redist.pdf.

Examples

```
## Not run:
data(algdat.pfull)

## Code to run the simulations in Figure 4 in Fifield, Higgins,Imai and
Tarr (2015)

## Get an initial partition
set.seed(1)
initcds <- algdat.pfull$cdmat[,sample(1:ncol(algdat.pfull$cdmat), 1)]

## Run the algorithm
alg_253 <- redist.mcmc(adjobj = algdat.pfull$adjlist,
popvec = algdat.pfull$precinct.data$pop,
initcds = initcds,
nsims = 10000, nloops = 2, savename = "test")
out <- redist.combine(savename = "test", nloop = 2,
nthin = 10)

## End(Not run)
```

`redist.combine.anneal` *redist.combine.anneal*

Description

Combine files generated by redist.mcmc.anneal()

Usage

```
redist.combine.anneal(file_name)
```

Arguments

`file_name` The file name to search for in current working directory.

redist.combine.mpi *Combine successive runs of* redist.mcmc.mpi

Description

redist.combine.mpi is used to combine successive runs of redist.mcmc.mpi into a single data object

Usage

```
redist.combine.mpi(savename, nloop, nthin, tempadj)
```

Arguments

savename	The name (without the loop or .RData suffix) of the saved simulations.
nloop	The number of loops being combined.
nthin	How much to thin the simulations being combined.
tempadj	The temperature adjacency object saved by redist.mcmc.mpi.

Details

This function allows users to combine multiple successive runs of redist.mcmc.mpi into a single redist object for analysis.

Value

redist.combine.mpi returns an object of class "redist". The object redist is a list that contains the folowing components (the inclusion of some components is dependent on whether tempering techniques are used):

partitions	Matrix of congressional district assignments generated by the algorithm. Each row corresponds to a geographic unit, and each column corresponds to a simulation.
distance_parity	Vector containing the maximum distance from parity for a particular simulated redistricting plan.
mhdecisions	A vector specifying whether a proposed redistricting plan was accepted (1) or rejected (0) in a given iteration.
mhprob	A vector containing the Metropolis-Hastings acceptance probability for each iteration of the algorithm.
pparam	A vector containing the draw of the p parameter for each simulation, which dictates the number of swaps attempted.
constraint_pop	A vector containing the value of the population constraint for each accepted redistricting plan.

`constraint_compact`
: A vector containing the value of the compactness constraint for each accepted redistricting plan.

`constraint_segregation`
: A vector containing the value of the segregation constraint for each accepted redistricting plan.

`constraint_similar`
: A vector containing the value of the similarity constraint for each accepted redistricting plan.

`beta_sequence`
: A vector containing the value of beta for each iteration of the algorithm. Returned when tempering is being used.

`mhdecisions_beta`
: A vector specifying whether a proposed beta value was accepted (1) or rejected (0) in a given iteration of the algorithm. Returned when tempering is being used.

`mhprob_beta`
: A vector containing the Metropolis-Hastings acceptance probability for each iteration of the algorithm. Returned when tempering is being used.

References

Fifield, Benjamin, Michael Higgins, Kosuke Imai and Alexander Tarr. (2016) "A New Automated Redistricting Simulator Using Markov Chain Monte Carlo." Working Paper. Available at http://imai.princeton.edu/research/files/redist.pdf.

Examples

```
## Not run:
data(algdat.pfull)

## Code to run the simulations in Figure 4 in Fifield, Higgins, Imai and
## Tarr (2015)

## Get an initial partition
set.seed(1)
initcds <- algdat.pfull$cdmat[,sample(1:ncol(algdat.pfull$cdmat), 1)]

## Run the algorithm
redist.mcmc.mpi(adjobj = algdat.pfull$adjlist,
popvec = algdat.pfull$precinct.data$pop,
initcds = initcds,
nsims = 10000, nloops = 2, savename = "test")
out <- redist.combine.mpi(savename = "test", nloop = 2,
nthin = 10, tempadj = tempAdjMat)

## End(Not run)
```

`redist.compactness` *Calculate compactness measures for a set of districts*

Description

`redist.compactness` is used to compute different compactness statistics for a shapefile. It currently computes the Polsby-Popper, Schwartzberg score, Length-Width Ratio, Convex Hull score, Reock score, Boyce Clark Index, Fryer Holden score, Edges Removed number, and the log of the Spanning Trees.

Usage

```
redist.compactness(
  shp = NULL,
  district_membership,
  measure = c("PolsbyPopper"),
  population = NULL,
  adjacency = NULL,
  nloop = 1,
  ncores = 1
)
```

Arguments

`shp`	A SpatialPolygonsDataFrame or sf object. Required unless "EdgesRemoved" and "logSpanningTree" with adjacency provided.
`district_membership`	A numeric vector (if only one map) or matrix with one row for each precinct and one column for each map. Required.
`measure`	A vector with a string for each measure desired. "PolsbyPopper", "Schwartzberg", "LengthWidth", "ConvexHull", "Reock", "BoyceClark", "FryerHolden", "EdgesRemoved", and "logSpanningTree" are implemented. Defaults to "PolsbyPopper". Use "all" to return all implemented measures.
`population`	A numeric vector with the population for every observation. Is only necessary when "FryerHolden" is used for measure. Defaults to NULL.
`adjacency`	A zero-indexed adjacency list. Only used for "EdgesRemoved" and "logSpanningTree". Created with `redist.adjacency` if not supplied and needed. Default is NULL.
`nloop`	A numeric to specify loop number. Defaults to 1 if only one map provided and the column number if multiple maps given.
`ncores`	Number of cores to use for parallel computing. Default is 1.

Details

This function computes specified compactness scores for a map. If there is more than one shape specified for a single district, it combines them, if necessary, and computes one score for each district.

Polsby-Popper is computed as

$$\frac{4 * \pi * A(d)}{P(d)^2}$$

where A is the area function, the district is d, and P is the perimeter function.

Schwartzberg is computed as

$$\frac{P(d)}{2 * \pi * \sqrt{\frac{A(d)}{\pi}}}$$

where A is the area function, the district is d, and P is the perimeter function.

The Length Width ratio is computed as

$$\frac{length}{width}$$

where length is the shorter of the maximum x distance and the maximum y distance. Width is the longer of the two values.

The Reock score is computed as

$$\frac{A(d)}{A(CVH)}$$

where A is the area function, d is the district, and CVH is the convex hull of the district.

The Boyce Clark Index is computed as

$$1 - \sum_{1}^{16} \{ \frac{|\frac{r_i}{\sum_i r_i} * 100 - 6.25|\}}{200}$$

. The r_i are the distances of the 16 radii computed from the geometric centroid of the shape to the most outward point of the shape that intersects the radii, if the centroid is contained within the shape. If the centroid lies outside of the shape, a point on the surface is used, which will naturally incur a penalty to the score.

The Fryer Holden score for each district is computed with

$$Pop \odot D(precinct)^2$$

, where Pop is the population product matrix. Each element is the product of the ith and jth precinct's populations. D represents the distance, where the matrix is the distance between each precinct. To fully compute this index, for any map, the sum of these values should be used as the numerator. The denominator can be calculated from the full enumeration of districts as the smallest calculated numerator.

The log spanning tree measure is the log number of spanning trees.

The edges removed measure is number of egdes removed from the underlying adjacency graph.

Value

A tibble with a column that specifies the district, a column for each specified measure, and a column that specifies the map number.

References

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

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.

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

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

Maceachren, A. 1985. Compactness of Geographic Shape: Comparison and Evaluation of Measures. Geografiska Annaler. Series B, Human Geography, 67(1), 53-67.

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.

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

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

Examples

```
## Not run:
library(sf)
library(lwgeom)
library(redist)
library(tidyverse)
#Create (or load) a shapefile, in this case the unit square
box <-  rbind(c(0,0), c(1,0), c(1,1), c(0,1), c(0,0)) %>% list() %>%
st_polygon() %>% st_sfc() %>% st_as_sf()
# Index the congressional districts
box <- box %>% mutate(cds = 1, pop = 10)
# Run redist.compactness
redist.compactness(box, "cds", "pop")

## End(Not run)
```

`redist.crsg` *Redistricting via Compact Random Seed and Grow Algorithm*

Description

`redist.crsg` generates redistricting plans using a random seed a grow algorithm. This is the compact districting algorithm described in Chen and Rodden (2013).

Usage

```
redist.crsg(
  adj.list,
  population,
  area,
  x_center,
  y_center,
  ndists,
  thresh,
  verbose = TRUE,
  maxiter = 5000
)
```

Arguments

adj.list	List of length N, where N is the number of precincts. Each list element is an integer vector indicating which precincts that precinct is adjacent to. It is assumed that precinct numbers start at 0.
population	numeric vector of length N, where N is the number of precincts. Each element lists the population total of the corresponding precinct, and is used to enforce population constraints.
area	numeric vector of length N, where N is the number of precincts. Each element is the area of the corresponding precinct.
x_center	numeric vector of length N, where N is the number of precincts. Each element is the x coordinate of the geographic centroid of the corresponding precinct.
y_center	numeric vector of length N, where N is the number of precincts. Each element is the y coordinate of the geographic centroid of the corresponding precinct.
ndists	integer, the number of districts we want to partition the precincts into.
thresh	numeric, indicating how close district population targets have to be to the target population before algorithm converges. thresh=0.05 for example means that all districts must be between 0.95 and 1.05 times the size of target.pop in population size.
verbose	boolean, indicating whether the time to run the algorithm is printed.
maxiter	integer, indicating maximum number of iterations to attempt before convergence to population constraint fails. If it fails once, it will use a different set of start values and try again. If it fails again, redist.rsg() returns an object of all NAs, indicating that use of more iterations may be advised. Default is 5000.

Value

list, containing three objects containing the completed redistricting plan.

- `district_membership` A vector of length N, indicating the district membership of each precinct.
- `district_list` A list of length Ndistrict. Each list contains a vector of the precincts in the respective district.

- district_pop A vector of length Ndistrict, containing the population totals of the respective districts.

References

Jowei Chen and Jonathan Rodden (2013) "Unintentional Gerrymandering: Political Geography and Electoral Bias in Legislatures." Quarterly Journal of Political Science. 8(3): 239-269.

Examples

```
## Not run:
data("fl25")
adj <- redist.adjacency(fl25)
area <- sf::st_area(fl25)
centers <- sf::st_coordinates(sf::st_centroid(fl25))
redist.crsg(adj.list = adj, population = fl25$pop, area = area,
x_center = centers[,1], y_center = centers[,2], ndists = 2, thresh = .1)

## End(Not run)
```

---

redist.diagplot *Diagnostic plotting functionality for MCMC redistricting.*

Description

redist.diagplot generates several common MCMC diagnostic plots.

Usage

```
redist.diagplot(sumstat,
plot = c("trace", "autocorr", "densplot", "mean", "gelmanrubin"),
logit = FALSE, savename = NULL)
```

Arguments

sumstat	A vector, list, mcmc or mcmc.list object containing a summary statistic of choice.
plot	The type of diagnostic plot to generate: one of "trace", "autocorr", "densplot", "mean", "gelmanrubin". If plot = "gelmanrubin", the input sumstat must be of class mcmc.list or list.
logit	Flag for whether to apply the logistic transformation for the summary statistic. The default is FALSE.
savename	Filename to save the plot. Default is NULL.

Details

This function allows users to generate several standard diagnostic plots from the MCMC literature, as implemented by Plummer et. al (2006). Diagnostic plots implemented include trace plots, autocorrelation plots, density plots, running means, and Gelman-Rubin convergence diagnostics (Gelman & Rubin 1992).

Value

Returns a plot of file type `.pdf`.

References

Fifield, Benjamin, Michael Higgins, Kosuke Imai and Alexander Tarr. (2016) "A New Automated Redistricting Simulator Using Markov Chain Monte Carlo." Working Paper. Available at http://imai.princeton.edu/research/files/redist.pdf.

Gelman, Andrew and Donald Rubin. (1992) "Inference from iterative simulations using multiple sequences (with discussion)." Statistical Science.

Plummer, Martin, Nicky Best, Kate Cowles and Karen Vines. (2006) "CODA: Convergence Diagnosis and Output Analysis for MCMC." R News.

Examples

```
## Not run:
data(algdat.pfull)

## Get an initial partition
set.seed(1)
initcds <- algdat.pfull$cdmat[,sample(1:ncol(algdat.pfull$cdmat), 1)]

## 25 precinct, three districts - no pop constraint ##
alg_253 <- redist.mcmc(adjobj = algdat.pfull$adjlist,
popvec = algdat.pfull$precinct.data$pop,
initcds = initcds,nsims = 10000)

## Get Republican Dissimilarity Index from simulations
rep_dmi_253 <- redist.segcalc(alg_253,
algdat.pfull$precinct.data$repvote,
algdat.pfull$precinct.data$pop)

## Generate diagnostic plots
redist.diagplot(rep_dmi_253, plot = "trace")
redist.diagplot(rep_dmi_253, plot = "autocorr")
redist.diagplot(rep_dmi_253, plot = "densplot")
redist.diagplot(rep_dmi_253, plot = "mean")

## Gelman Rubin needs two chains, so we run a second
alg_253_2 <- redist.mcmc(adjobj = algdat.pfull$adjlist,
popvec = algdat.pfull$precinct.data$pop,
initcds = initcds,nsims = 10000)
```

```
rep_dmi_253_2 <- redist.segcalc(alg_253_2,
algdat.pfull$precinct.data$repvote,
algdat.pfull$precinct.data$pop)

## Make a list out of the objects:
rep_dmi_253_list <- list(rep_dmi_253, rep_dmi_253_2)


## Generate Gelman Rubin diagnostic plot
redist.diagplot(sumstat = rep_dmi_253_list, plot = 'gelmanrubin')

## End(Not run)
```

`redist.distances` *Compute Distance between Partitions*

Description

Compute Distance between Partitions

Usage

```
redist.distances(district_membership, measure = "Hamming", ncores = 1)
```

Arguments

`district_membership`	A matrix with one row for each precinct and one column for each map. Required.
`measure`	String vector indicating which distances to compute. Implemented currently are "Hamming", "Manhattan", and "Euclidean". Use all to return all implemented measures.
`ncores`	Number of cores to use for parallel computing. Default is 1.

Value

list of matrices of distances with one for each distance measure selected

Examples

```
## Not run:
data("algdat.p10")
distances <- redist.distances(district_membership = algdat.p10$cdmat)
distances$Hamming[1:5,1:5]

## End(Not run)
```

redist.enumerate *Exact Redistricting Plan Enumerator*

Description

redist.enumerate uses a spanning-tree method to fully enumerate all valid redistricting plans with n districts given a set of geographic units. redist.enumerate also allows suers to implement minimum and maximum numbers of geographic units per district, as well as population parity requirements.

Usage

```
redist.enumerate(adjobj,
ndists = 2, popvec = NULL, nconstraintlow = NULL,
nconstrainthigh = NULL, popcons = NULL, contiguitymap = "rooks")
```

Arguments

adjobj	An adjacency list, matrix, or object of class SpatialPolygonsDataFrame.
ndists	The desired number of congressional districts. The default is 2.
popvec	A vector of geographic unit populations. The default is NULL.
nconstraintlow	Lower bound for number of geographic units to include in a district. The default is NULL.
nconstrainthigh	Lower bound for number of geographic units to include in a district. The default is NULL.
popcons	The strength of the hard population constraint. popcons = 0.05 means that any proposed swap that brings a district more than 5% away from population parity will be rejected. The default is NULL.
contiguitymap	Use queens or rooks distance criteria for generating an adjacency list from a "SpatialPolygonsDataFrame" data type. Default is "rooks".

Details

This function allows users to input a set of geographic units to generate all valid partitions of n congressional districts. The function uses a set of spanning-tree methods to generate all valid, contiguous partitions, which makes it more efficient than brute-force methods. However, even with these methods, full redistricting problems quickly become intractable, necessitating the use of the MCMC-based methods implemented in redist.mcmc.

Value

redist.enumerate returns an object of class "list". Each entry in the list is a vector of congressional district assignments, where the first entry in the vector corresponds to the congressional district assignment of the first geographic unit.

References

Fifield, Benjamin, Michael Higgins, Kosuke Imai and Alexander Tarr. (2016) "A New Automated Redistricting Simulator Using Markov Chain Monte Carlo." Working Paper. Available at http://imai.princeton.edu/research/files/redist.pdf.

Examples

```
## Not run:
data(algdat.pfull)
test <- redist.enumerate(adjobj = algdat.pfull$adjlist)

## End(Not run)
```

---

redist.findparams *Run parameter testing for* redist.mcmc

Description

redist.findparams is used to find optimal parameter values of redist.mcmc for a given map.

Usage

```
redist.findparams(adjobj, popvec, nsims, ndists = NULL, initcds = NULL,
adapt_lambda = FALSE, adapt_eprob = FALSE,
params, ssdmat = NULL, grouppopvec = NULL, countymembership = NULL,
nstartval_store, maxdist_startval,
maxiterrsg = 5000, report_all = TRUE,
parallel = FALSE, nthreads = NULL, log = FALSE, verbose = TRUE)
```

Arguments

adjobj	An adjacency matrix, list, or object of class "SpatialPolygonsDataFrame."
popvec	A vector containing the populations of each geographic unit.
nsims	The number of simulations run before a save point.
ndists	The number of congressional districts. The default is NULL.
initcds	A vector containing the congressional district labels of each geographic unit. The default is NULL. If not provided, random and contiguous congressional district assignments will be generated using redist.rsg.
adapt_lambda	Whether to adaptively tune the lambda parameter so that the Metropolis-Hastings acceptance probability falls between 20% and 40%. Default is FALSE.
adapt_eprob	Whether to adaptively tune the edgecut probability parameter so that the Metropolis-Hastings acceptance probability falls between 20% and 40%. Default is FALSE.
params	A matrix of parameter values to test, such as the output of expand.grid. Parameters accepted for params include eprob, lambda, popcons, beta, and constraint.
ssdmat	A matrix of squared distances between geographic units. The default is NULL.

`grouppopvec` A vector of populations for some sub-group of interest. The default is `NULL`.

`countymembership` A vector of county membership assignments. The default is `NULL`.

`nstartval_store` The number of maps to sample from the preprocessing chain for use as starting values in future simulations. Default is 1.

`maxdist_startval` The maximum distance from the starting map that sampled maps should be. Default is 100 (no restriction).

`maxiterrsg` Maximum number of iterations for random seed-and-grow algorithm to generate starting values. Default is 5000.

`report_all` Whether to report all summary statistics for each set of parameter values. Default is `TRUE`.

`parallel` Whether to run separate parameter settings in parallel. Default is `FALSE`.

`nthreads` Number of parallel tasks to run, declared outside of the function. Default is `NULL`.

`log` Whether to open a log to track progress for each parameter combination being tested. Default is FALSE.

`verbose` Whether to print additional information about the tests. Default is `TRUE`.

Details

This function allows users to test multiple parameter settings of `redist.mcmc` in preparation for a longer run for analysis.

Value

`redist.findparams` returns a print-out of summary statistics about each parameter setting.

References

Fifield, Benjamin, Michael Higgins, Kosuke Imai and Alexander Tarr. (2016) "A New Automated Redistricting Simulator Using Markov Chain Monte Carlo." Working Paper. Available at http://imai.princeton.edu/research/files/redist.pdf.

Examples

```
## Not run:
data(algdat.pfull)

## Code to run the simulations in Figure 4 in Fifield, Higgins, Imai and
## Tarr (2015)

## Get an initial partition
set.seed(1)
initcds <- algdat.pfull$cdmat[,sample(1:ncol(algdat.pfull$cdmat), 1)]

params <- expand.grid(eprob = c(.01, .05, .1))
```

```
## Run the algorithm
redist.findparams(adjobj = algdat.pfull$adjlist,
popvec = algdat.pfull$precinct.data$pop,
initcds = initcds, nsims = 10000, params = params)

## End(Not run)
```

`redist.init.enumpart` *Initialize enumpart*

Description

This ensures that the enumerate partitions programs is prepared to run. This must be run once per install of the redist package.

Usage

```
redist.init.enumpart()
```

Value

0 on success

References

Benjamin Fifield, Kosuke Imai, Jun Kawahara, and Christopher T Kenny. "The Essential Role of Empirical Validation in Legislative Redistricting Simulation." Forthcoming, Statistics and Public Policy.

Examples

```
## Not run:
redist.init.enumpart()

## End(Not run)
```

`redist.ipw` *Inverse probability reweighting for MCMC Redistricting*

Description

`redist.ipw` properly weights and resamples simulated redistricting plans so that the set of simulated plans resemble a random sample from the underlying distribution. `redist.ipw` is used to correct the sample when population parity, geographic compactness, or other constraints are implemented.

Usage

```
redist.ipw(algout, targetpop = NULL)
```

Arguments

algout	An object of class "redist".
targetpop	The desired level of population parity. targetpop = 0.01 means that the desired distance from population parity is 1%. The default is NULL.

Details

This function allows users to resample redistricting plans using inverse probability weighting techniques described in Rubin (1987). This techniques reweights and resamples redistricting plans so that the resulting sample is representative of a random sample from the uniform distribution.

Value

redist.ipw returns an object of class "redist". The object redist is a list that contains the folowing components (the inclusion of some components is dependent on whether tempering techniques are used):

partitions	Matrix of congressional district assignments generated by the algorithm. Each row corresponds to a geographic unit, and each column corresponds to a simulation.
distance_parity	Vector containing the maximum distance from parity for a particular simulated redistricting plan.
mhdecisions	A vector specifying whether a proposed redistricting plan was accepted (1) or rejected (0) in a given iteration.
mhprob	A vector containing the Metropolis-Hastings acceptance probability for each iteration of the algorithm.
pparam	A vector containing the draw of the p parameter for each simulation, which dictates the number of swaps attempted.
constraint_pop	A vector containing the value of the population constraint for each accepted redistricting plan.
constraint_compact	A vector containing the value of the compactness constraint for each accepted redistricting plan.
constraint_segregation	A vector containing the value of the segregation constraint for each accepted redistricting plan.
constraint_similar	A vector containing the value of the similarity constraint for each accepted redistricting plan.
beta_sequence	A vector containing the value of beta for each iteration of the algorithm. Returned when tempering is being used.

`mhdecisions_beta`
: A vector specifying whether a proposed beta value was accepted (1) or rejected (0) in a given iteration of the algorithm. Returned when tempering is being used.

`mhprob_beta`
: A vector containing the Metropolis-Hastings acceptance probability for each iteration of the algorithm. Returned when tempering is being used.

References

Fifield, Benjamin, Michael Higgins, Kosuke Imai and Alexander Tarr. (2016) "A New Automated Redistricting Simulator Using Markov Chain Monte Carlo." Working Paper. Available at http://imai.princeton.edu/research/files/redist.pdf.

Rubin, Donald. (1987) "Comment: A Noniterative Sampling/Importance Resampling Alternative to the Data Augmentation Algorithm for Creating a Few Imputations when Fractions of Missing Information are Modest: the SIR Algorithm." Journal of the American Statistical Association.

Examples

```
## Not run:
data(algdat.p20)

## Code to run the simulations in Figure 4 of Fifield, Higgins,
## Imai and Tarr (2015)

## Get an initial partition
set.seed(1)
initcds <- algdat.p20$cdmat[,sample(1:ncol(algdat.p20$cdmat), 1)]

## Vector of beta weights
betaweights <- rep(NA, 10); for(i in 1:10){betaweights[i] <- 4^i}

## Run simulations - tempering population constraint
alg_253_20_st <- redist.mcmc(adjobj = algdat.p20$adjlist,
popvec = algdat.p20$precinct.data$pop,
initcds = initcds, nsims = 10000, betapop = -5.4,
betaweights = betaweights, temperbetapop = 1)

## Resample using inverse probability weighting.
## Target distance from parity is 20%
alg_253_20_st <- redist.ipw(alg_253_20_st, targetpop = .2)


## End(Not run)
```

`redist.map` *Creates a map with optional graph overlay*

Description

Creates a map with optional graph overlay

Usage

```
redist.map(
  shp = NULL,
  district_membership = NULL,
  centroids = TRUE,
  edges = TRUE,
  drop = FALSE,
  title = ""
)
```

Arguments

shp	A SpatialPolygonsDataFrame or sf object. Required.
district_membership	A numeric vector with one row for each precinct in shp. Used to color the districts. Default is NULL. Optional.
centroids	A logical indicating if centroids should be plotted. Default is TRUE.
edges	A logical indicating if edges should connect adjacent centroids. Default is TRUE.
drop	A logical indicating if edges that cross districts should be dropped. Default is FALSE.
title	A string title of plot. Defaults to empty string. Optional.

Value

ggplot map

Examples

```
## Not run:
library(redist)
data("fl25")
data("algdat.p10")
cds <- algdat.p10$cdmat[,100]
redist.map(shp = fl25, district_membership = cds)

## End(Not run)
```

---

redist.mcmc *MCMC Redistricting Simulator*

Description

redist.mcmc is used to simulate Congressional redistricting plans using Markov Chain Monte Carlo methods.

Usage

```
redist.mcmc(
  adjobj,
  popvec,
  nsims,
  ndists = NULL,
  initcds = NULL,
  loopscompleted = 0,
  nloop = 1,
  nthin = 1,
  eprob = 0.05,
  lambda = 0,
  popcons = NULL,
  grouppopvec = NULL,
  areasvec = NULL,
  countymembership = NULL,
  borderlength_mat = NULL,
  ssdmat = NULL,
  temper = FALSE,
  constraint = NULL,
  constraintweights = NULL,
  compactness_metric = "fryer-holden",
  ssd_denom = 1,
  betaseq = "powerlaw",
  betaseqlength = 10,
  betaweights = NULL,
  adjswaps = TRUE,
  rngseed = NULL,
  maxiterrsg = 5000,
  adapt_lambda = FALSE,
  adapt_eprob = FALSE,
  contiguitymap = "rooks",
  exact_mh = FALSE,
  savename = NULL,
  verbose = TRUE
)
```

Arguments

adjobj	An adjacency matrix, list, or object of class "SpatialPolygonsDataFrame."
popvec	A vector containing the populations of each geographic unit
nsims	The number of simulations run before a save point.
ndists	The numbe of congressional districts. The default is NULL.
initcds	A vector containing the congressional district labels of each geographic unit. The default is NULL. If not provided, random and contiguous congressional district assignments will be generated using redist.rsg.
loopscompleted	Number of save points reached by the algorithm. The default is 0.

`nloop` The total number of save points for the algorithm. The default is `1`. Note that the total number of simulations run will be `nsims` * `nloop`.

`nthin` The amount by which to thin the Markov Chain. The default is `1`.

`eprob` The probability of keeping an edge connected. The default is `0.05`.

`lambda` The parameter detmerining the number of swaps to attempt each iteration fo the algoirhtm. The number of swaps each iteration is equal to Pois(`lambda`) + 1. The default is `0`.

`popcons` The strength of the hard population constraint. `popcons` = 0.05 means that any proposed swap that brings a district more than 5% away from population parity will be rejected. The default is `NULL`.

`grouppopvec` A vector of populations for some sub-group of interest. The default is `NULL`.

`areasvec` A vector of precinct areas for discrete Polsby-Popper. The default is `NULL`.

`countymembership`
A vector of county membership assignments. The default is `NULL`.

`borderlength_mat`
A matrix of border length distances, where the first two columns are the indices of precincts sharing a border and the third column is its distance. Default is `NULL`.

`ssdmat` A matrix of squared distances between geographic units. The default is `NULL`.

`temper` Whether to use simulated tempering algorithm. Default is FALSE.

`constraint` Which constraint to apply. Accepts any combination of `compact`, `segregation`, `population`, `similarity`, or `none` (no constraint applied). The default is NULL.

`constraintweights`
The weights to apply to each constraint. Should be a vector the same length as constraint. Default is NULL.

`compactness_metric`
The compactness metric to use when constraining on compactness. Default is `fryer-holden`, the other implemented option is `polsby-popper`.

`ssd_denom` The normalizing constant for the sum-of-squared distance Fryer-Holden metric. Default is 1.0 (unnormalized).

`betaseq` Sequence of beta values for tempering. The default is `powerlaw` (see Fifield et. al (2015) for details).

`betaseqlength` Length of beta sequence desired for tempering. The default is `10`.

`betaweights` Sequence of weights for different values of beta. Allows the user to upweight certain values of beta over others. The default is `NULL` (equal weighting).

`adjswaps` Flag to restrict swaps of beta so that only values adjacent to current constraint are proposed. The default is `TRUE`.

`rngseed` Allows the user to set the seed for the simulations. Default is `NULL`.

`maxiterrsg` Maximum number of iterations for random seed-and-grow algorithm to generate starting values. Default is 5000.

`adapt_lambda` Whether to adaptively tune the lambda parameter so that the Metropolis-Hastings acceptance probability falls between 20% and 40%. Default is FALSE.

adapt_eprob	Whether to adaptively tune the edgecut probability parameter so that the Metropolis-Hastings acceptance probability falls between 20% and 40%. Default is FALSE.
contiguitymap	Use queens or rooks distance criteria for generating an adjacency list from a "SpatialPolygonsDataFrame" data type. Default is "rooks".
exact_mh	Whether to use the approximate (0) or exact (1) Metropolis-Hastings ratio calculation for accept-reject rule. Default is FALSE.
savename	Filename to save simulations. Default is NULL.
verbose	Whether to print initialization statement. Default is TRUE.

Details

This function allows users to simulate redistricting plans using Markov Chain Monte Carlo methods. Several constraints correspoding to substantive requirements in the redistricting process are implemented, including population parity and geographic compactness. In addition, the function includes multiple-swap and simulated tempering functionality to improve the mixing of the Markov Chain.

Value

redist.mcmc returns an object of class "redist". The object redist is a list that contains the folowing components (the inclusion of some components is dependent on whether tempering techniques are used):

partitions	Matrix of congressional district assignments generated by the algorithm. Each row corresponds to a geographic unit, and each column corresponds to a simulation.
distance_parity	Vector containing the maximum distance from parity for a particular simulated redistricting plan.
mhdecisions	A vector specifying whether a proposed redistricting plan was accepted (1) or rejected (0) in a given iteration.
mhprob	A vector containing the Metropolis-Hastings acceptance probability for each iteration of the algorithm.
pparam	A vector containing the draw of the p parameter for each simulation, which dictates the number of swaps attempted.
constraint_pop	A vector containing the value of the population constraint for each accepted redistricting plan.
constraint_compact	A vector containing the value of the compactness constraint for each accepted redistricting plan.
constraint_segregation	A vector containing the value of the segregation constraint for each accepted redistricting plan.
constraint_similar	A vector containing the value of the similarity constraint for each accepted redistricting plan.

`beta_sequence` A vector containing the value of beta for each iteration of the algorithm. Returned when tempering is being used.

`mhdecisions_beta` A vector specifying whether a proposed beta value was accepted (1) or rejected (0) in a given iteration of the algorithm. Returned when tempering is being used.

`mhprob_beta` A vector containing the Metropolis-Hastings acceptance probability for each iteration of the algorithm. Returned when tempering is being used.

References

Fifield, Benjamin, Michael Higgins, Kosuke Imai and Alexander Tarr. (2016) "A New Automated Redistricting Simulator Using Markov Chain Monte Carlo." Working Paper. Available at http://imai.princeton.edu/research/files/redist.pdf.

Examples

```
## Not run:
data(algdat.pfull)
## Code to run the simulations in Figure 4 in Fifield, Higgins,
## Imai and Tarr (2015)

## Get an initial partition
set.seed(1)
initcds <- algdat.pfull$cdmat[,sample(1:ncol(algdat.pfull$cdmat), 1)]

## Run the algorithm
alg_253 <- redist.mcmc(adjobj = algdat.pfull$adjlist,
popvec = algdat.pfull$precinct.data$pop,
initcds = initcds,
nsims = 10000)

## End(Not run)
```

---

redist.mcmc.anneal *MCMC Redistricting Simulator using Simulated Annealing*

Description

`redist.mcmc.anneal` simulates congressional redistricting plans using Markov chain Monte Carlo methods coupled with simulated annealing.

Usage

```
redist.mcmc.anneal(adjobj, popvec, ndists,
initcds, num_hot_steps, num_annealing_steps,
num_cold_steps,
eprob, lambda, popcons, grouppopvec,
areasvec, countymembership, borderlength_mat,
```

```
ssdmat, constraint, constraintweights,
compactness_metric, rngseed, maxiterrsg,
adapt_lambda, adapt_eprob,
contiguitymap, exact_mh,
savename, verbose, ncores)
```

Arguments

adjobj	An adjacency matrix, list, or object of class "SpatialPolygonsDataFrame."
popvec	A vector containing the populations of each geographic unit
ndists	The numbe of congressional districts. The default is NULL.
initcds	A vector containing the congressional district labels of each geographic unit. The default is NULL. If not provided, random and contiguous congressional district assignments will be generated using redist.rsg.
num_hot_steps	The number of steps to run the simulator at beta = 0. Default is 40000.
num_annealing_steps	The number of steps to run the simulator with linearly changing beta schedule. Default is 60000
num_cold_steps	The number of steps to run the simulator at beta = 1. Default is 20000.
eprob	The probability of keeping an edge connected. The default is 0.05.
lambda	The parameter detmerining the number of swaps to attempt each iteration fo the algoirhtm. The number of swaps each iteration is equal to Pois(lambda) + 1. The default is 0.
popcons	The strength of the hard population constraint. popcons = 0.05 means that any proposed swap that brings a district more than 5% away from population parity will be rejected. The default is NULL.
grouppopvec	A vector of populations for some sub-group of interest. The default is NULL.
areasvec	A vector of precinct areas for discrete Polsby-Popper. The default is NULL.
countymembership	A vector of county membership assignments. The default is NULL.
borderlength_mat	A matrix of border length distances, where the first two columns are the indices of precincts sharing a border and the third column is its distance. Default is NULL.
ssdmat	A matrix of squared distances between geographic units. The default is NULL.
constraint	Which constraint to apply. Accepts any combination of compact, segregation, population, similarity, or none (no constraint applied). The default is NULL.
constraintweights	The weights to apply to each constraint. Should be a vector the same length as constraint. Default is NULL.
compactness_metric	The compactness metric to use when constraining on compactness. Default is fryer-holden, the other implemented option is polsby-popper.
rngseed	Allows the user to set the seed for the simulations. Default is NULL.

maxiterrsg	Maximum number of iterations for random seed-and-grow algorithm to generate starting values. Default is 5000.
adapt_lambda	Whether to adaptively tune the lambda parameter so that the Metropolis-Hastings acceptance probability falls between 20% and 40%. Default is FALSE.
adapt_eprob	Whether to adaptively tune the edgecut probability parameter so that the Metropolis-Hastings acceptance probability falls between 20% and 40%. Default is FALSE.
contiguitymap	Use queens or rooks distance criteria for generating an adjacency list from a "SpatialPolygonsDataFrame" data type. Default is "rooks".
exact_mh	Whether to use the approximate (0) or exact (1) Metropolis-Hastings ratio calculation for accept-reject rule. Default is FALSE.
savename	Filename to save simulations. Default is NULL.
verbose	Whether to print initialization statement. Default is TRUE.
ncores	The number of cores available to parallelize over. Default is 1.

`redist.mcmc.mpi` *MCMC Redistricting Simulator using MPI*

Description

`redist.mcmc.mpi` is used to simulate Congressional redistricting plans using Markov Chain Monte Carlo methods.

Usage

```
redist.mcmc.mpi(adjobj, popvec, nsims, ndists = NA, initcds = NULL,
loopscompleted = 0, nloop = 1, nthin = 1,
eprob = 0.05,
lambda = 0, popcons = NA, grouppopvec = NA,
areasvec = NA, countymembership = NA,
borderlength_mat = NA, ssdmat = NA,
compactness_metric = "fryer-holden", rngseed = NA,
constraint = NA, constraintweights = NA,
betaseq = "powerlaw",
betaseqlength = 10, adjswaps = TRUE,
freq = 100, savename = NA, maxiterrsg = 5000,
contiguitymap = "rooks", verbose = FALSE)
```

Arguments

adjobj	An adjacency matrix, list, or object of class "SpatialPolygonsDataFrame."
popvec	A vector containing the populations of each geographic unit.
nsims	The number of simulations run before a save point.
ndists	The number of congressional districts. The default is NULL.

`initcds` A vector containing the congressional district labels of each geographic unit. The default is `NULL`. If not provided, random and contiguous congressional district assignments will be generated using `redist.rsg`.

`loopscompleted` Number of save points reached by the algorithm. The default is `0`.

`nloop` The total number of save points for the algorithm. The default is `1`. Note that the total number of simulations run will be `nsims` * `nloop`.

`nthin` The amount by which to thin the Markov Chain. The default is `1`.

`eprob` The probability of keeping an edge connected. The default is `0.05`.

`lambda` The parameter determining the number of swaps to attempt each iteration of the algorithm. The number of swaps each iteration is equal to Pois(`lambda`) + 1. The default is `0`.

`popcons` The strength of the hard population constraint. `popcons` = 0.05 means that any proposed swap that brings a district more than 5% away from population parity will be rejected. The default is `NULL`.

`grouppopvec` A vector of populations for some sub-group of interest. The default is `NULL`.

`areasvec` A vector of precinct areas for discrete Polsby-Popper. The default is `NULL`.

`countymembership`
A vector of county membership assignments. The default is `NULL`.

`borderlength_mat`
A matrix of border length distances, where the first two columns are the indices of precincts sharing a border and the third column is its distance. Default is `NULL`.

`ssdmat` A matrix of squared distances between geographic units. The default is `NULL`.

`compactness_metric`
The compactness metric to use when constraining on compactness. Default is `fryer-holden`, the other implemented option is `polsby-popper`.

`rngseed` Allows the user to set the seed for the simulations. Default is `NULL`.

`constraint` Which constraint to apply. Accepts any combination of `compact`, `segregation`, `population`, `similarity`, or `none` (no constraint applied). The default is NULL.

`constraintweights`
The weights to apply to each constraint. Should be a vector the same length as constraint. Default is NULL.

`betaseq` Sequence of beta values for tempering. The default is `powerlaw` (see Fifield et. al (2015) for details).

`betaseqlength` Length of beta sequence desired for tempering. The default is `10`.

`adjswaps` Flag to restrict swaps of beta so that only values adjacent to current constraint are proposed. The default is `TRUE`.

`freq` Frequency of between-chain swaps. Default to once every 100 iterations

`savename` Filename to save simulations. Default is `NULL`.

`maxiterrsg` Maximum number of iterations for random seed-and-grow algorithm to generate starting values. Default is 5000.

`contiguitymap` Use queens or rooks distance criteria for generating an adjacency list from a "SpatialPolygonsDataFrame" data type. Default is "rooks".

`verbose` Whether to print initialization statement. Default is `TRUE`.

Details

This function allows users to simulate redistricting plans using Markov Chain Monte Carlo methods. Several constraints correspoding to substantive requirements in the redistricting process are implemented, including population parity and geographic compactness. In addition, the function includes multiple-swap and parallel tempering functionality in MPI to improve the mixing of the Markov Chain.

Value

`redist.mcmc.mpi` returns an object of class "redist". The object `redist` is a list that contains the folowing components (the inclusion of some components is dependent on whether tempering techniques are used):

`partitions` Matrix of congressional district assignments generated by the algorithm. Each row corresponds to a geographic unit, and each column corresponds to a simulation.

`distance_parity` Vector containing the maximum distance from parity for a particular simulated redistricting plan.

`mhdecisions` A vector specifying whether a proposed redistricting plan was accepted (1) or rejected (0) in a given iteration.

`mhprob` A vector containing the Metropolis-Hastings acceptance probability for each iteration of the algorithm.

`pparam` A vector containing the draw of the `p` parameter for each simulation, which dictates the number of swaps attempted.

`constraint_pop` A vector containing the value of the population constraint for each accepted redistricting plan.

`constraint_compact` A vector containing the value of the compactness constraint for each accepted redistricting plan.

`constraint_segregation` A vector containing the value of the segregation constraint for each accepted redistricting plan.

`constraint_similar` A vector containing the value of the similarity constraint for each accepted redistricting plan.

`beta_sequence` A vector containing the value of beta for each iteration of the algorithm. Returned when tempering is being used.

`mhdecisions_beta` A vector specifying whether a proposed beta value was accepted (1) or rejected (0) in a given iteration of the algorithm. Returned when tempering is being used.

`mhprob_beta` A vector containing the Metropolis-Hastings acceptance probability for each iteration of the algorithm. Returned when tempering is being used.

References

Fifield, Benjamin, Michael Higgins, Kosuke Imai and Alexander Tarr. (2016) "A New Automated Redistricting Simulator Using Markov Chain Monte Carlo." Working Paper. Available at http://imai.princeton.edu/research/files/redist.pdf.

Examples

```
## Not run:
data(algdat.pfull)
## Code to run the simulations in Figure 4 in Fifield, Higgins,
## Imai and Tarr (2015)

## Get an initial partition
set.seed(1)
initcds <- algdat.pfull$cdmat[,sample(1:ncol(algdat.pfull$cdmat), 1)]

## Run the algorithm
redist.mcmc.mpi(adjobj = algdat.pfull$adjlist,
popvec = algdat.pfull$precinct.data$pop,
initcds = initcds,
nsims = 10000, savename = "test")

## End(Not run)
```

`redist.metrics` *Calculate gerrymandering metrics for a set of districts*

Description

`redist.metrics` is used to compute different gerrymandering metrics for a set of maps.

Usage

```
redist.metrics(
  district_membership,
  measure = "DSeats",
  rvote,
  dvote,
  tau = 1,
  biasV = 0.5,
  respV = 0.5,
  bandwidth = 0.01,
  nloop = 1,
  ncores = 1
)
```

Arguments

district_membership	A numeric vector (if only one map) or matrix with one row for each precinct and one column for each map. Required.
measure	A vector with a string for each measure desired from list "DSeats", "DVS", "EffGap", "EffGapEqPop", "TauGap", "MeanMedian", "Bias", "BiasV", "Declination", "Responsiveness", and "LopsidedWins". Use "all" to get all metrics. "Dseats" and "DVS" are always computed, so it is recommended to always return those values.
rvote	A numeric vector with the Republican vote for each precinct.
dvote	A numeric vector with the Democratic vote for each precinct.
tau	A non-negative number for calculating Tau Gap. Only used with option "TauGap". Defaults to 1.
biasV	A value between 0 and 1 to compute bias at. Only used with option "BiasV". Defaults to 0.5.
respV	A value between 0 and 1 to compute responsiveness at. Only used with option "Responsiveness". Defaults to 0.5.
bandwidth	A value between 0 and 1 for computing responsiveness. Only used with option "Responsiveness." Defaults to 0.01.
nloop	A numeric to specify loop number. Defaults to 1 if only one map provided and the column number if multiple maps given.
ncores	Number of cores to use for parallel computing. Default is 1.

Details

This function computes specified compactness scores for a map. If there is more than one precinct specified for a map, it aggregates to the district level and computes one score.

DSeats is computed as the expected number of Democratic seats with no change in votes. DVS is the Democratic Vote Share, which is the two party vote share with Democratic votes as the numerator. EffGap is the Efficiency Gap, calculated with votes directly. EffGapEqPop is the Efficiency Gap under an Equal Population assumption, calculated with the DVS. TauGap is the Tau Gap, comptued with the Equal Population assumption. MeanMedian is the Mean Median difference. Bias is the Partisan Bias computed at 0.5. BiasV is the Partisan Bias computed at value V. Declination is the value of declination at 0.5. Responsiveness is the responsiveness at the user-supplied value with the user-supplied bandwidth. LopsidedWins computed the Lopsided Outcomes value, but does not produce a test statistic.

Value

A tibble with a column for each specified measure and a column that specifies the map number.

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.

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

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

Examples

```
## Not run:
data("algdat.p10")
dists <- algdat.p10$cdmat[,1:100]
redist.metrics(dists, measure = 'all', rvote = algdat.p10$precinct.data$repvote,
dvote = algdat.p10$precinct.data$demvote)

## End(Not run)
```

---

redist.parity *Calculates Population Parity*

Description

`redist.parity` computes the population parity of a matrix of maps.

Usage

```
redist.parity(district_membership, population, ncores = 1)
```

Arguments

`district_membership`	A matrix with one row for each precinct and one column for each map. Required.
`population`	A numeric vector with the population for every precinct.
`ncores`	Number of cores to use for parallel computing. Default is 1.

Value

numeric vector with the population parity for each column

redist.prep.enumpart *Prepares a run of the enumpart algorithm by ordering edges*

Description

Prepares a run of the enumpart algorithm by ordering edges

Usage

```
redist.prep.enumpart(adjlist, unordered_path, ordered_path)
```

Arguments

adjlist	zero indexed adjacency list
unordered_path	valid path to output the unordered adjacency map to
ordered_path	valid path to output the ordered adjacency map to

Value

0 on success

References

Benjamin Fifield, Kosuke Imai, Jun Kawahara, and Christopher T Kenny. "The Essential Role of Empirical Validation in Legislative Redistricting Simulation." Forthcoming, Statistics and Public Policy.

Examples

```
## Not run:
data("algdat.p10")
redist.prep.enumpart(adjlist = algdat.p10$adjlist, unordered_path = '../unordered',
ordered_path = '../ordered')

## End(Not run)
```

redist.read.enumpart *Read Results from enumpart*

Description

Read Results from enumpart

Usage

```
redist.read.enumpart(out_path, skip = 0, n_max = -1L)
```

Arguments

out_path	out_path specified in redist.run.enumpart
skip	number of lines to skip
n_max	max number of lines to read

Value

district_membership matrix

References

Benjamin Fifield, Kosuke Imai, Jun Kawahara, and Christopher T Kenny. "The Essential Role of Empirical Validation in Legislative Redistricting Simulation." Forthcoming, Statistics and Public Policy.

Examples

```
## Not run:
cds <- redist.read.enumpart(out_path = '../enumerated')

## End(Not run)
```

---

redist.rsg *Redistricting via Random Seed and Grow Algorithm*

Description

redist.rsg generates redistricting plans using a random seed a grow algorithm. This is the non-compact districting algorithm described in Chen and Rodden (2013). The algorithm can provide start values for the other redistricting routines in this package.

Usage

```
redist.rsg(adj.list, population, ndists, thresh,
verbose = TRUE, maxiter=5000)
```

Arguments

adj.list	List of length N, where N is the number of precincts. Each list element is an integer vector indicating which precincts that precinct is adjacent to. It is assumed that precinct numbers start at 0.
population	numeric vector of list N, where N is the number of precincts. Each element lists the population total of the corresponding precinct, and is used to enforce population constraints.
ndists	integer, the number of districts we want to partition the precincts into.

thresh	numeric, indicating how close district population targets have to be to the target population before algorithm converges. thresh=0.05 for example means that all districts must be between 0.95 and 1.05 times the size of target.pop in population size.
verbose	boolean, indicating whether the time to run the algorithm is printed.
maxiter	integer, indicating maximum number of iterations to attempt before convergence to population constraint fails. If it fails once, it will use a different set of start values and try again. If it fails again, redist.rsg() returns an object of all NAs, indicating that use of more iterations may be advised.

Value

list, containing three objects containing the completed redistricting plan.

- district_membership A vector of length N, indicating the district membership of each precinct.
- district_list A list of length Ndistrict. Each list contains a vector of the precincts in the respective district.
- district_pop A vector of length Ndistrict, containing the population totals of the respective districts.

Author(s)

Benjamin Fifield, Department of Politics, Princeton University <benfifield@gmail.com>, https://www.benfifield.com/

Michael Higgins, Department of Statistics, Kansas State University <mikehiggins@k-state.edu>, http://www-personal.k-state.edu/~mikehiggins/

Kosuke Imai, Department of Politics, Princeton University <imai@harvard.edu>, http://imai.fas.harvard.edu

James Lo, <jameslo@princeton.edu>

Alexander Tarr, Department of Electrical Engineering, Princeton University <atarr@princeton.edu>

References

Jowei Chen and Jonathan Rodden (2013) "Unintentional Gerrymandering: Political Geography and Electoral Bias in Legislatures." Quarterly Journal of Political Science. 8(3): 239-269.

Examples

```
### Real data example from test set
data("algdat.pfull")
res <- redist.rsg(algdat.pfull$adjlist, algdat.pfull$precinct.data$pop, 3, 0.05)

## Not run:
### Example that generates test data from a square map with equal population
districts
### Number of precincts is Nrows*Ncols
### getTest() outputs an adjacency list out of specified rows and columns
```

```
genTest <- function(Nrows,Ncols){
NN <- Nrows * Ncols
geog <- matrix(NA,nrow=Nrows+2, ncol=Ncols+2)
geog[2:(Nrows+1), 2:(Ncols+1)] <- 0:(NN-1)

adj.list <- vector("list", NN)

for(i in 2:(Nrows+1)){
    for(j in 2:(Ncols+1)){
       adj.list[[ geog[i,j] + 1 ]] <- c(geog[i-1,j],geog[i+1,j],geog[i,j-1],geog[i,j+1])
    }
}
adj.list <- lapply(adj.list, na.omit)
adj.list <- lapply(adj.list, as.numeric)
return(adj.list)
}

### Generate a 100x100 precinct map and redistrict it into 10 districts
adj.list <- genTest(100,100)
population <- rep(300,length(adj.list))
tmp <- redist.rsg(adj.list, population, 10, 0.05)

## End(Not run)
```

`redist.run.enumpart` *Runs the enumpart algorithm*

Description

Runs the enumpart algorithm

Usage

```
redist.run.enumpart(
  ordered_path,
  out_path,
  ndist = 2,
  all = TRUE,
  n = NULL,
  options = NULL
)
```

Arguments

`ordered_path`	Path used in redist.prep.enumpart
`out_path`	Valid path to output the enumerated districts
`ndist`	number of districts to enumerate

all	boolean. TRUE outputs all districts. FALSE samples n districts.
n	integer. Number of districts to output if all is FALSE. Returns districts selected from uniform random distribution.
options	Additional enumpart arguments. Not recommended for use.

Value

0 on success

References

Benjamin Fifield, Kosuke Imai, Jun Kawahara, and Christopher T Kenny. "The Essential Role of Empirical Validation in Legislative Redistricting Simulation." Forthcoming, Statistics and Public Policy.

Examples

```
## Not run:
redist.run.enumpart(ordered_path = '../ordered', out_path = '../enumerated')

## End(Not run)
```

---

redist.samplepart *Sample partitions using spanning trees*

Description

redist.samplepart uses a spanning tree method to randomly sample redistricting plans.

Usage

```
redist.samplepart(adjobj, ndists, popvec, pop_filter, pop_constraint,
contiguitymap, nsamp, n_cores)
```

Arguments

adjobj	An adjacency list, matrix, or object of class SpatialPolygonsDataFrame.
ndists	The desired number of congressional districts
popvec	Population vector for adjacency object. Provide if filtering by population
pop_filter	Boolean. Whether or not to filter on population parity. Default is FALSE.
pop_constraint	Strength of population filter if filtering on distance to parity.
contiguitymap	Use queens or rooks distance criteria for generating an adjacency list from a "SpatialPolygonsDataFrame" data type. Default is "rooks".
nsamp	Number of samples to draw. Default is 1000.
n_cores	Number of cores to parallelize over for parity calculation and compactness calculation. Default is 1.

Value

`redist.samplepart` returns a list where the first entry is the randomly sampled redistricting plan, and the second entry is the number of possible redistricting plans from the implied spanning tree.

---

redist.segcalc *Segregation index calculation for MCMC redistricting.*

Description

`redist.segcalc` calculates the dissimilarity index of segregation (see Massey \& Denton 1987 for more details) for a specified subgroup under any redistricting plan.

Usage

```
redist.segcalc(algout, grouppop, fullpop)
```

Arguments

`algout`	A matrix of congressional district assignments or a redist object.
`grouppop`	A vector of populations for some subgroup of interest.
`fullpop`	A vector containign the populations of each geographic unit.

Value

`redist.segcalc` returns a vector where each entry is the dissimilarity index of segregation (Massey & Denton 1987) for each redistricting plan in `algout`.

References

Fifield, Benjamin, Michael Higgins, Kosuke Imai and Alexander Tarr. (2016) "A New Automated Redistricting Simulator Using Markov Chain Monte Carlo." Working Paper. Available at http://imai.princeton.edu/research/files/redist.pdf.

Massey, Douglas and Nancy Denton. (1987) "The Dimensions of Social Segregation". Social Forces.

Examples

```
## Not run:
data(algdat.pfull)

## Code to run the simulations in Figure 4 of Fifield, Higgins,
## Imai and Tarr (2015)

## Get an initial partition
set.seed(1)
initcds <- algdat.pfull$cdmat[,sample(1:ncol(algdat.pfull$cdmat), 1)]
```

```
## Run simulations
alg_253 <- redist.mcmc(adjobj = algdat.pfull$adjlist,
popvec = algdat.pfull$precinct.data$pop,
initcds = initcds, nsims = 10000)

## Get Republican Dissimilarity Index from simulations
rep_dmi_253 <- redist.segcalc(alg_253,
algdat.pfull$precinct.data$repvote,
algdat.pfull$precinct.data$pop)

## End(Not run)
```

`redist.smc` *SMC Redistricting Sampler*

Description

`redist.smc` uses a Sequential Monte Carlo algorithm to generate nearly independent congressional or legislative redistricting plans according to contiguity, population, compactness, and administrative boundary constraints.

Usage

```
redist.smc(
  adjobj,
  popvec,
  nsims,
  ndists,
  counties = NULL,
  popcons = 0.01,
  compactness = 1,
  resample = TRUE,
  constraint_fn = function(m) rep(0, ncol(m)),
  adapt_k_thresh = 0.95,
  seq_alpha = 0.1 + 0.2 * compactness,
  truncate = (compactness != 1),
  trunc_fn = function(x) pmin(x, 0.01 * nsims^0.4),
  max_oversample = 20,
  verbose = TRUE,
  silent = FALSE
)
```

Arguments

`adjobj`	An adjacency matrix, list, or object of class "SpatialPolygonsDataFrame."
`popvec`	A vector containing the populations of each geographic unit.
`nsims`	The number of samples to draw.

ndists	The number of districts in each redistricting plan.
counties	A vector containing county (or other administrative or geographic unit) labels for each unit, which must be integers ranging from 1 to the number of counties. If provided, the algorithm will only generate maps which split up to ndists-1 counties. If no county-split constraint is desired, this parameter should be left blank.
popcons	The desired population constraint. All sampled districts will have a deviation from the target district size no more than this value in percentage terms, i.e., popcons=0.01 will ensure districts have populations within 1% of the target population.
compactness	Controls the compactness of the generated districts, with higher values preferring more compact districts. Must be nonnegative. See the 'Details' section for more information, and computational considerations.
resample	Whether to perform a final resampling step so that the generated plans can be used immediately. Set this to FALSE to perform direct importance sampling estimates, or to adjust the weights manually.
constraint_fn	A function which takes in a matrix where each column is a redistricting plan and outputs a vector of log-weights, which will be added the the final weights.
adapt_k_thresh	The threshold value used in the heuristic to select a value k_i for each splitting iteration. Set to 0.9999 or 1 if the algorithm does not appear to be sampling from the target distribution. Must be between 0 and 1.
seq_alpha	The amount to adjust the weights by at each resampling step; higher values prefer exploitation, while lower values prefer exploration. Must be between 0 and 1.
truncate	Whether to truncate the importance sampling weights at the final step by trunc_fn. Recommended if compactness is not 1.
trunc_fn	A function which takes in a vector of weights and returns a truncated vector. Recommended to specify this manually if truncating weights.
max_oversample	How much oversampling to allow at each stage; used to control memory and computation time. If the algorithm is not producing the desired nubmer of samples, this should be increased.
verbose	Whether to print out intermediate information while sampling. Recommended.
silent	Whether to supress all diagnostic information.

Details

This function draws nearly-independent samples from a specific target measure, controlled by the popcons, compactness, and constraint_fn parameters.

Higher values of compactness sample more compact districts; setting this parameter to 1 is computationally efficient and generates nicely compact districts. Values of other than 1 may lead to highly variable importance sampling weights. By default these weights are truncated at nsims^0.04 / 100 to stabilize the resulting estimates, but if truncation is used, a specific truncation function should probably be chosen by the user.

Because of the randomness inherent in the algorithm and the way it samples, this function is not guaranteed to produce exactly nsims samples. Failure to do so is usually a result of a hard-to-meet

population constraint, especially when there are many districts. Increasing `max_oversample` should generally alleviate this problem.

Value

`redist.smc` returns an object of class `redist`, which is a list containing the following components:

`aList`	The adjacency list used to sample
`cdvec`	The matrix of sampled plans. Each row is a geographical unit, and each column is a sample.
`wgt`	The importance sampling weights, normalized to sum to 1.
`nsims`	The number of plans sampled.
`pct_dist_parity`	The population constraint.
`compactness`	The compactness constraint.
`maxdev`	The maximum population deviation of each sample.
`popvec`	The provided vector of unit populations.
`counties`	The provided county vector.
`adapt_k_thresh`	The provided control parameter.
`seq_alpha`	The provided control vector.
`max_oversample`	The provided control vector.
`algorithm`	The algorithm used, here `"smc"`.

Examples

```
## Not run:
data(algdat.p10)
sampled_plans = redist.smc(algdat.pfull$adjlist, algdat.pfull$precinct.data$pop,
                           nsims=10000, ndists=3, popcons=0.1)

## End(Not run)
```

---

redist.smc_is_ci *Confidence Intervals for Importance Sampling Estimates*

Description

Builds a confidence interval for a quantity of interest, given importance sampling weights.

Usage

```
redist.smc_is_ci(x, wgt, conf = 0.99)
```

Arguments

x	A numeric vector containing the quantity of interest
wgt	A numeric vector containing the nonnegative importance weights. Will be normalized automatically.
conf	The confidence level for the interval.

Value

A two-element vector of the form [lower, upper] containing the importance sampling confidence interval.

Index