

Package ‘redist’

December 15, 2018

Version 1.3-3

Date 2018-12-15

Title Markov Chain Monte Carlo Methods for Redistricting Simulation

Author Ben Fifield <bfifield@princeton.edu>, Alexander Tarr
<atarr@princeton.edu>, Michael Higgins <mikehiggins@k-state.edu>, and Kosuke
Imai <imai@harvard.edu>

Maintainer Ben Fifield <bfifield@princeton.edu>

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 <<https://imai.fas.harvard.edu/research/files/redist.pdf>>.

Depends R (>= 3.1.0)

Imports Rcpp (>= 0.11.0), spdep, sp, coda, parallel, doParallel, foreach

Suggests testthat, Rmpi

LinkingTo Rcpp, RcppArmadillo

License GPL (>= 2)

SystemRequirements gmp, libxml2

NeedsCompilation yes

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

RoxygenNote 6.0.1

Repository CRAN

Date/Publication 2018-12-15 20:40:03 UTC

R topics documented:

| | |
|------------------------------|-----------|
| redist-package | 2 |
| algdat.p10 | 3 |
| algdat.p20 | 4 |
| algdat.pfull | 5 |
| redist.combine | 6 |
| redist.combine.mpi | 8 |
| redist.diagplot | 10 |
| redist.enumerate | 11 |
| redist.findparams | 12 |
| redist.ipw | 14 |
| redist.mcmc | 17 |
| redist.mcmc.mpi | 20 |
| redist.rsg | 22 |
| redist.segcalc | 24 |
| Index | 26 |

| | |
|----------------|----------------------------------------------------------------------|
| redist-package | <i>Markov Chain Monte Carlo Methods for Redistricting Simulation</i> |
|----------------|----------------------------------------------------------------------|

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: 1.3-2
 Date: 2018-10-15
 License: GPL (>= 2)

Author(s)

Benjamin Fifield, Department of Politics, Princeton University <bfifield@princeton.edu>, <http://www.benfifield.com>

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

Maintainer: Ben Fifield <bfifield@princeton.edu>

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 | <i>All Partitions of 25 Precincts into 3 Congressional Districts (No Population Constraint)</i> |
|--------------|-------------------------------------------------------------------------------------------------|

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)
```

| | |
|----------------|-----------------------------------------------|
| redist.combine | <i>Combine successive runs of redist.mcmc</i> |
|----------------|-----------------------------------------------|

Description

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

Usage

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

Arguments

| | |
|----------|---------------------------------------------------------------------------------------|
| savename | The name (without the loop or .RData suffix) of the saved simulations. |
| nsims | The number of simulations in each loop. |
| nloop | The number of loops being combined. |
| nthin | How much to thin the simulations being combined. |
| nunits | The number of geographic units from the simulations. |
| temper | Whether 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 following 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$pop,
initcds = initcds,
nsims = 10000, nloops = 2, savename = "test")
out <- redist.combine(savename = "test", nsims = 10000, nloop = 2,
nthin = 10, nunits = length(algdat.pfull$adjlist))
```

```
## End(Not run)
```

```
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, nsims, nloop, nthin, nunits, tempadj)
```

Arguments

| | |
|----------|------------------------------------------------------------------------|
| savename | The name (without the loop or .RData suffix) of the saved simulations. |
| nsims | The number of simulations in each loop. |
| nloop | The number of loops being combined. |
| nthin | How much to thin the simulations being combined. |
| nunits | The number of geographic units from the simulations. |
| 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 following 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$pop,
initcds = initcds,
nsims = 10000, nloops = 2, savename = "test")
out <- redist.combine.mpi(savename = "test", nsims = 10000, nloop = 2,
nthin = 10, nunits = length(algdat.pfull$adjlist), tempadj = tempAdjMat)

## 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)
initcids <- 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,
initcids = initcids, 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")

## 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 users 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,
nconstrainhigh = 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,
  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 <code>redist.rsg</code> . |
| 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 edgcut 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 <code>expand.grid</code> . Parameters accepted for params include <code>eprob</code> , <code>lambda</code> , <code>popcons</code> , <code>beta</code> , and <code>constraint</code> . |
| 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. |
| 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)
initcids <- 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,
initcids = initcids, nsims = 10000, params = params)

## 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,
resampleconstraint = c("pop", "compact", "segregation", "similar"),
targetbeta, targetpop = NULL, temper = 0)
```

Arguments

| | |
|---------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>algout</code> | An object of class "redist". |
| <code>resampleconstraint</code> | The constraint implemented in the simulations: one of "pop", "compact", "segregation", or "similar". |
| <code>targetbeta</code> | The target value of the constraint. |
| <code>targetpop</code> | The desired level of population parity. <code>targetpop = 0.01</code> means that the desired distance from population parity is 1%. The default is NULL. |
| <code>temper</code> | A flag for whether simulated tempering was used to improve the mixing of the Markov Chain. The default is 1. |

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 following components (the inclusion of some components is dependent on whether tempering techniques are used):

| | |
|-------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>partitions</code> | Matrix of congressional district assignments generated by the algorithm. Each row corresponds to a geographic unit, and each column corresponds to a simulation. |
| <code>distance_parity</code> | Vector containing the maximum distance from parity for a particular simulated redistricting plan. |
| <code>mhdecisions</code> | A vector specifying whether a proposed redistricting plan was accepted (1) or rejected (0) in a given iteration. |
| <code>mhprob</code> | A vector containing the Metropolis-Hastings acceptance probability for each iteration of the algorithm. |
| <code>pparam</code> | A vector containing the draw of the p parameter for each simulation, which dictates the number of swaps attempted. |
| <code>constraint_pop</code> | A vector containing the value of the population constraint for each accepted redistricting plan. |
| <code>constraint_compact</code> | A vector containing the value of the compactness constraint for each accepted redistricting plan. |
| <code>constraint_segregation</code> | A vector containing the value of the segregation constraint for each accepted redistricting plan. |
| <code>constraint_similar</code> | 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,
resampleconstraint = "pop",
targetbeta = -5.4,
targetpop = .2, temper = 1)

## 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, ssdmat = NULL,
beta = 0, temper = "none", constraint = "none",
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 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 for the algorithm. The number of swaps each iteration is equal to $\text{Pois}(\lambda) + 1$. 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. |
| ssdmat | A matrix of squared distances between geographic units. The default is NULL. |
| beta | The strength of the target strength in the MH ratio. The default is 0. |

| | |
|----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>temper</code> | Type of tempering algorithm to use. Accepts <code>simulated</code> , <code>parallel</code> , or <code>none</code> (no tempering applied). The default is <code>none</code> . |
| <code>constraint</code> | Which constraint to apply. Accepts <code>compact</code> , <code>segregation</code> , <code>population</code> , <code>similarity</code> , or <code>none</code> (no constraint applied). The default is <code>none</code> . |
| <code>betaseq</code> | Sequence of beta values for tempering. The default is <code>powerlaw</code> (see Fifield et. al (2015) for details). |
| <code>betaseqlength</code> | Length of beta sequence desired for tempering. The default is <code>10</code> . |
| <code>betaweights</code> | Sequence of weights for different values of beta. Allows the user to upweight certain values of beta over others. The default is <code>NULL</code> (equal weighting). |
| <code>adjswaps</code> | Flag to restrict swaps of beta so that only values adjacent to current constraint are proposed. The default is <code>TRUE</code> . |
| <code>rngseed</code> | Allows the user to set the seed for the simulations. Default is <code>NULL</code> . |
| <code>maxiterrsg</code> | Maximum number of iterations for random seed-and-grow algorithm to generate starting values. Default is <code>5000</code> . |
| <code>adapt_lambda</code> | Whether to adaptively tune the lambda parameter so that the Metropolis-Hastings acceptance probability falls between 20% and 40%. Default is <code>FALSE</code> . |
| <code>adapt_eprob</code> | Whether to adaptively tune the edgcut probability parameter so that the Metropolis-Hastings acceptance probability falls between 20% and 40%. Default is <code>FALSE</code> . |
| <code>contiguitymap</code> | Use queens or rooks distance criteria for generating an adjacency list from a "SpatialPolygonsDataFrame" data type. Default is "rooks". |
| <code>exact_mh</code> | Whether to use the approximate (0) or exact (1) Metropolis-Hastings ratio calculation for accept-reject rule. Default is <code>FALSE</code> . |
| <code>savename</code> | Filename to save simulations. Default is <code>NULL</code> . |
| <code>verbose</code> | Whether to print initialization statement. Default is <code>TRUE</code> . |

Details

This function allows users to simulate redistricting plans using Markov Chain Monte Carlo methods. Several constraints corresponding 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 following components (the inclusion of some components is dependent on whether tempering techniques are used):

| | |
|------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>partitions</code> | Matrix of congressional district assignments generated by the algorithm. Each row corresponds to a geographic unit, and each column corresponds to a simulation. |
| <code>distance_parity</code> | 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.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,
ssdmat = NA, rngseed = NA,
beta = -10, constraint = "population",
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 $\text{Pois}(\lambda) + 1$. 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. |
| ssdmat | A matrix of squared distances between geographic units. The default is NULL. |
| rngseed | Allows the user to set the seed for the simulations. Default is NULL. |

| | |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| beta | The strength of the target strength in the MH ratio. The default is 0. |
| constraint | Which constraint to apply. Accepts compact, segregation, population, similarity, or none (no constraint applied). The default is none. |
| 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 corresponding 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 following 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)
initcgs <- 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,
initcgs = initcgs,
nsims = 10000, savename = "test")

## 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 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. |
| 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 <bfifield@princeton.edu>, <http://www.benfifield.com>

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

Kosuke Imai, Department of Politics, Princeton University <kimai@princeton.edu>, <http://imai.princeton.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.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)
initcids <- 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,
initcids = initcids, 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)
```

Index

*Topic **package**

[redist-package](#), [2](#)

[almdat.p10](#), [3](#)

[almdat.p20](#), [4](#)

[almdat.pfull](#), [5](#)

[redist \(redist-package\)](#), [2](#)

[redist-package](#), [2](#)

[redist.combine](#), [6](#)

[redist.combine.mpi](#), [8](#)

[redist.diagplot](#), [10](#)

[redist.enumerate](#), [11](#)

[redist.findparams](#), [12](#)

[redist.ipw](#), [14](#)

[redist.mcmc](#), [17](#)

[redist.mcmc.mpi](#), [20](#)

[redist.rsg](#), [22](#)

[redist.segcalc](#), [24](#)