Package 'GERGM'

May 15, 2018

Type Package

Title Estimation and Fit Diagnostics for Generalized Exponential Random Graph Models

Version 0.13.0

Date 2018-05-14

Author Matthew J. Denny <mdenny@psu.edu>, James D. Wilson <jdwilson1212@gmail.com>, Skyler Cranmer <cranmer.12@osu.edu >, Bruce A. Desmarais <bdesmarais@psu.edu>, Shankar Bhamidi <bhamidi@email.unc.edu>

Maintainer Matthew J. Denny <mdenny@psu.edu>

Description Estimation and diagnosis of the convergence of Generalized Exponential Random Graph Models via Gibbs sampling or Metropolis Hastings with exponential down weighting.

URL https://github.com/matthewjdenny/GERGM

License GPL-2

Imports Rcpp, ggplot2, methods, stringr, igraph, plyr, parallel, coda, vegan, scales, RcppParallel, slackr, matrixcalc

Depends R (>= 3.2.0)

LinkingTo BH, Rcpp, RcppArmadillo, RcppParallel

RoxygenNote 6.0.1

Suggests testthat, knitr, rmarkdown

SystemRequirements GNU make

VignetteBuilder knitr

NeedsCompilation yes

Repository CRAN

Date/Publication 2018-05-15 18:05:28 UTC

R topics documented:

`calculate_network_statistics`
A Function to estimate calculate (weighted) network statistics

---

Description

Calculates out2stars, in2stars, ctriads, mutual, ttriads, and edges statistics (with or without exponential down weighting) for a real-valued network.

Usage

```
calculate_network_statistics(network, weights = NULL,
  downweight_statistics_together = TRUE, include_diagonal = FALSE)
```

Arguments

`network`	A square numeric matrix (sociomatrix or adjacency matrix) representing the network.
`weights`	If you wish to provide your own weights, you must provide a vector of length 6 with terms corresponding to out2stars, in2stars, ctriads, mutual, ttriads, edges in that order.

`downweight_statistics_together`
: Logical indicating whether exponential down weighting should be done together or separately. Defaults to TRUE.

`include_diagonal`
: Logical indicating whether the diagonal should be included in the network statistics. Defaults to FALSE.

Value

A gergm object containing parameter estimates.

`conditional_edge_prediction` *A Function to predict edge weights from a GERGM fit object.*

Description

Performs edgewise predictions from a GERGM model fit.

Usage

```
conditional_edge_prediction(GERGM_Object, simulation_method = c("Metropolis",
  "Gibbs"), number_of_networks_to_simulate = 500, thin = 1,
  proposal_variance = 0.1, MCMC_burnin = 100, seed = 123,
  return_constrained_networks = FALSE, optimize_proposal_variance = FALSE,
  target_accept_rate = 0.25, use_stochastic_MH = FALSE,
  stochastic_MH_proportion = 1)
```

Arguments

`GERGM_Object`
: A GERGM object output by the gergm() estimation function. The following terms must still be specified: number_of_networks_to_simulate, thin, and MCMC_burnin. proposal_variance may also be specified, or if set equal to NULL, then the proposal variance from parameter estimation will be instead (this option is likely preferred in most situations).

`simulation_method`
: Default is "Metropolis" which allows for exponential down weighting, can also be "Gibbs".

`number_of_networks_to_simulate`
: Number of simulations generated for estimation via MCMC. Default is 500.

`thin`
: The proportion of samples that are kept from each simulation. For example, thin = 1/200 will keep every 200th network in the overall simulated sample. Default is 1.

`proposal_variance`
: The variance specified for the Metropolis Hastings simulation method. This parameter is inversely proportional to the average acceptance rate of the M-H sampler and should be adjusted so that the average acceptance rate is approximately 0.25. Default is 0.1.

`MCMC_burnin` Number of samples from the MCMC simulation procedure that will be discarded before drawing the samples used for estimation. Default is 100.

`seed` Seed used for reproducibility. Default is 123.

`return_constrained_networks`
Logical argument indicating whether simulated networks should be transformed back to observed scale or whether constrained [0,1] networks should be returned. Defaults to FALSE, in which case networks are returned on observed scale.

`optimize_proposal_variance`
Logical indicating whether proposal variance should be optimized if using Metropolis Hastings for simulation. Defaults to FALSE.

`target_accept_rate`
Defaults to 0.25, can be used to optimize Metropolis Hastings simulations.

`use_stochastic_MH`
A logical indicating whether a stochastic approximation to the h statistics should be used under Metropolis Hastings in-between thinned samples. This may dramatically speed up estimation. Defaults to FALSE. HIGHLY EXPERIMENTAL!

`stochastic_MH_proportion`
Percentage of dyads/triads to use for approximation, defaults to 0.25

Value

A list object containing simulated networks.

`conditional_edge_prediction_correlation_plot`
A Function generate a scatter plot of observed vs predicted edge values.

Description

Scatter plot of observed vs. predicted edge values generated by the 'conditional_edge_prediction()' function. Also calculated correlation coefficient.

Usage

```
conditional_edge_prediction_correlation_plot(edge_prediction_results,
  xlim = NULL, ylim = NULL, plot_max_entropy_predictions = FALSE)
```

Arguments

`edge_prediction_results`
A list object returned by the 'conditional_edge_prediction()' function.

`xlim` Defaults to NULL, but can be set by the user if desired to a numeric vector of the form c(lower,upper) for observed edge values.

`ylim` Defaults to NULL, but can be set by the user if desired to a numeric vector of the form c(lower,upper) for predicted edge values.

`plot_max_entropy_predictions` Defaults to FALSE, can be set to TRUE to show maximum entropy edge predictions as red X's.

Value

Saves a PDF plot to the current working directory

`conditional_edge_prediction_MSE`

A Function to calculate Mean Edgewise MSE to evaluate edge predictions.

Description

Calculates mean edgewise MSE for predicted edge values.

Usage

```
conditional_edge_prediction_MSE(edge_prediction_results)
```

Arguments

`edge_prediction_results` A list object returned by the 'conditional_edge_prediction()' function.

Value

A list of MSE's.

`conditional_edge_prediction_plot`

A Function generate edge prediction plots.

Description

Plots edgewise predictions generated by the 'conditional_edge_prediction()' function.

Usage

```
conditional_edge_prediction_plot(edge_prediction_results, filename, plot_size,
  node_name_cex = 1)
```

Arguments

edge_prediction_results	A list object returned by the 'conditional_edge_prediction()' function.
filename	The name of the file the user would like to save the plots in. Should be something along the lines of "edge_plots.pdf"
plot_size	A single number specifying the dimensions of the PDF output file. This will automatically be square so only one number is required.
node_name_cex	The cex for node names (printed on the diagonal). Defaults to 1, but can be increased to make these easier to see.

Value

Saves a PDF plot to the current working directory

---

convert_simulated_networks_to_observed_scale

Transforms simulated networks to observed scale. In general, do not use this function.

Description

Transforms simulated networks to observed scale. In general, do not use this function.

Usage

```
convert_simulated_networks_to_observed_scale(GERGM_Object)
```

Arguments

GERGM_Object	A GERGM object returned by the 'gergm()' function. In general, this function should not be used except in the case where you are working with a GERGM object where the '@MCMC_output$Networks' field is still on the [0,1] unconstrained space, and you wish to transform it to the observed scale.

Value

A GERGM Object

covariate_data_2005 *GDP and G8 membership data for 17 large countries from 2005.*

Description

Metadata for each country in "lending_2005", including GDP, logged GDP, and a G8 membership indicator.

Usage

```
covariate_data_2005
```

Format

A data frame with 17 rows and 3 columns

Source

http://arxiv.org/abs/1505.04015

Estimate_Plot *Generate parameter estimate plot with 95 percent CI's from a GERGM object.*

Description

Generate parameter estimate plot with 95 percent CI's from a GERGM object.

Usage

```
Estimate_Plot(GERGM_Object, normalize_coefficients = FALSE,
  coefficients_to_plot = c("both", "covariate", "structural"),
  coefficient_names = NULL, leave_out_coefficients = NULL,
  comparison_model = NULL, model_names = NULL, text_size = 12)
```

Arguments

GERGM_Object	The object returned by the estimation procedure using the GERGM function.
normalize_coefficients	Defaults to FALSE, if TRUE then parameter estimates will be converted be divided by their standard deviations with and displayed with 95 percent confidence intervals. These coefficients will no longer be comparable, but make graphical interpretation of significance and sign easier.

`coefficients_to_plot`
: An optional argument indicating which kind of parameters to plot. Can be one of "both","covariate", or "structural". Useful for creating separate parameter plots for covariates and structural parameters when these parameters are on very different scales.

`coefficient_names`
: Defaults to NULL. Can be a string vector of names for coefficients to be used in making publication quality plots.

`leave_out_coefficients`
: Defaults to NULL. Can be a string vector of coefficient names as they appear in the plot. These coefficients will be removed from the final plot. Useful if the intercept term is much larger in magnitude than other estimates, and the user wishes to clarify the other parameter estimates without normalizing. given

`comparison_model`
: A GERGM_Object produced by an alternative model whose parameter estimates are to be compared to the existing model. Defaults to NULL.

`model_names`
: If a comparison_model is provided, then each model must be given a name via the model_names parameter. Defaults to NULL.

`text_size`
: The base size for axis text. Defaults to 12.

Value

A parameter estimate plot.

`find_example_simulated_network`
: *Find an example simulated network that is the minimum Frobenius distance from the observed network.*

Description

Find an example simulated network that is the minimum Frobenius distance from the observed network.

Usage

```
find_example_simulated_network(GERGM_Object, observed_network,
  objective = c("Frobenius", "Likelihood"))
```

Arguments

`GERGM_Object`
: The object returned by the estimation procedure using the GERGM function.

`observed_network`
: The observed network used as the dependent variable in the original gergm() specification.

objective — Defaults to "Frobenius", in which case the Frobenius norm is used to find the simulated network that is most similar to the observed network. Can also be "Likelihood", in which case the model log likelihood is used to select the network to display.

Value

A simulated network (as a matrix object).

---

gergm *A Function to estimate a GERGM.*

Description

The main function provided by the package.

Usage

```
gergm(formula, covariate_data = NULL, beta_correlation_model = FALSE,
  distribution_estimator = c("none", "rowwise-marginal", "joint"),
  network_is_directed = TRUE, include_diagonal = FALSE,
  number_of_networks_to_simulate = 500, MCMC_burnin = 100, thin = 1,
  proposal_variance = 0.1, target_accept_rate = 0.25, seed = 123,
  hyperparameter_optimization = FALSE, convex_hull_proportion = 0.9,
  convex_hull_convergence_proportion = 0.9, sample_edges_at_a_time = 0,
  parallel = FALSE, parallel_statistic_calculation = FALSE, cores = 1,
  use_stochastic_MH = FALSE, stochastic_MH_proportion = 0.25,
  slackr_integration_list = NULL, convergence_tolerance = 0.5,
  MPLE_gain_factor = 0, acceptable_fit_p_value_threshold = 0.05,
  normalization_type = c("log", "division"),
  transformation_type = c("Cauchy", "LogCauchy", "Gaussian", "LogNormal"),
  estimation_method = c("Metropolis", "Gibbs"),
  downweight_statistics_together = TRUE, force_x_theta_updates = 1,
  force_x_lambda_updates = 1, output_directory = NULL, output_name = NULL,
  generate_plots = TRUE, verbose = TRUE, use_previous_thetas = TRUE,
  fine_grained_pv_optimization = FALSE, use_MPLE_only = c(FALSE, TRUE),
  start_with_zeros = FALSE, stop_for_degeneracy = FALSE,
  maximum_number_of_lambda_updates = 10,
  maximum_number_of_theta_updates = 10,
  user_specified_initial_thetas = NULL, integration_intervals = 150,
  distribution_mple_regularization_weight = 0.05,
  theta_grid_optimization_list = NULL, weighted_MPLE = FALSE,
  estimate_model = TRUE, optimization_method = c("BFGS", "L-BFGS-B",
  "Nelder-Mead", "CG", "SANN", "Brent"), ...)
```

Arguments

`formula` A formula object that specifies the relationship between statistics and the observed network. Currently, the user may specify a model using any combination of the following statistics: ‘out2stars(alpha = 1)‘, ‘in2stars(alpha = 1)‘, ‘ctriads(alpha = 1)‘, ‘mutual(alpha = 1)‘, ‘ttriads(alpha = 1)‘, ‘absdiff(covariate = "MyCov")‘, ‘sender(covariate = "MyCov")‘, ‘reciever(covariate = "MyCov")‘, ‘nodematch(covariate)‘, ‘nodemix(covariate, base = "MyBase")‘, ‘netcov(network)‘ and ‘edges(alpha = 1, method = c("regression","endogenous"))‘. If the user specifies ‘nodemix(covariate, base = NULL)‘, then all levels of the covariate will be matched on. Note that the ‘edges‘ term must be specified if the user wishes to include an intercept (strongly recommended). The user may select the "regression" method (default) to include an intercept in the lambda transformation of the network, or "endogenous" to include the intercept as in a traditional ERGM model. To use exponential down-weighting for any of the network level terms, simply specify a value for alpha less than 1. The ‘(alpha = 1)‘ term may be omitted from the structural terms if no exponential down weighting is required. In this case, the terms may be provided as: ‘out2star‘, ‘in2star‘, ‘ctriads‘, ‘mutual‘, ‘ttriads‘. If the network is undirected the user may only specify the following terms: ‘twostars(alpha = 1)‘, ‘ttriads(alpha = 1)‘, ‘absdiff(covariate = "MyCov")‘, ‘sender(covariate = "MyCov")‘, ‘nodematch(covariate)‘, ‘nodemix(covariate, base = "MyBase")‘, ‘netcov(network)‘ and ‘edges(alpha = 1, method = c("regression","endogenous"))‘. In some cases, the user may only wish to calculate endogenous statistics for edges between some subset of the nodes in the network. For each of the endogenous statistics, the user may optionally specify a ‘covariate‘ and ‘base‘ field such as in ‘in2stars(covariate = "Type", base = "C")‘. This will add an in-2star statistic for the subnetwork defined by actors who match each level of the categorical variable "Type" (in this example), and exclude the subnetwork for type "C", if the ‘base‘ argument is provided. If the ‘base‘ argument is excluded, then terms will be added to the model for all levels of the statistic. This can be a useful option if the user believes that a network property varies with some property of nodes.

`covariate_data` A data frame containing node level covariates the user wished to transform into sender or receiver effects. It must have row names that match every entry in colnames(raw_network), should have descriptive column names. If left NULL, then no sender or receiver effects will be added.

`beta_correlation_model`
Defaults to FALSE. If TRUE, then the beta correlation model is estimated. A correlation network must be provided, but all covariates and undirected statistics may be supplied as normal.

`distribution_estimator`
Provides an option to estimate the structure of row-wise marginal and joint distribtuions using a uniform-dirichlet proposal distribution. THIS FEATURE IS EXPERIMENTAL. Defaults to "none", in which case a normal GERGM is estimated, but can be set to one of "rowwise-marginal" and "joint" to propose either row-wsie marginal distribtuions or joint distributions. If an option other than "none" is selected, the beta correlation model will be turned off, estimation will automatically be set to Metropolis, no covariate data will be allowed, and the network will be set to directed. Furthermore, a "diagonal" statistic will be added

to the model which simply records the sum of the diagonal of the network. The "mutual" statistic will also be adapted to include the diagonal elements. In the future, more statistics which take account of the network diagonal will be included.

`network_is_directed`
: Logical specifying whether or not the observed network is directed. Default is TRUE.

`include_diagonal`
: Logical indicating whether the diagonal should be included in the estimation proceedure. If TRUE, then a "diagonal" statistic is added to the model. Defaults to FALSE.

`number_of_networks_to_simulate`
: Number of simulations generated for estimation via MCMC. Default is 500.

`MCMC_burnin`
: Number of samples from the MCMC simulation procedure that will be discarded before drawing the samples used for estimation. Default is 100.

`thin`
: The proportion of samples that are kept from each simulation. For example, thin = 1/200 will keep every 200th network in the overall simulated sample. Default is 1.

`proposal_variance`
: The variance specified for the Metropolis Hastings simulation method. This parameter is inversely proportional to the average acceptance rate of the M-H sampler and should be adjusted so that the average acceptance rate is approximately 0.25. Default is 0.1.

`target_accept_rate`
: The target Metropolis Hastings acceptance rate. Defaults to 0.25

`seed`
: Seed used for reproducibility. Default is 123.

`hyperparameter_optimization`
: Logical indicating whether automatic hyperparameter optimization should be used. Defaults to FALSE. If TRUE, then the algorithm will automatically seek to find an optimal burnin and number of networks to simulate, and if using Metropolis Hasings, will attempt to select a proposal variance that leads to a acceptance rate within +-0.05 of target_accept_rate. Furthermore, if degeneracy is detected, the algorithm will attempt to adress the issue automatically. WARNING: This feature is experimental, and may greatly increase runtime. Please monitor console output!

`convex_hull_proportion`
: Defaults to 0.9. Must be a number between 0 and 1, with values between 0.5 and 0.9 preferred. Setting a value for this parameter makes use of the initialization method introduced by Hummel, Hunter and Handcock (2012), which can provide a much more stable initialization method than MPLE. Selecting this method will not supercede other initialization methods, as it will be run after MPLE or grid search. However, in practice, it should often preclude the need for other initialization methods. If NULL, then standard MPLE an grid search methods may be used. Not recommended

`convex_hull_convergence_proportion`
: Defaults to 0.9. This must be a number between 0 and 1, with values between 0.7 and 0.95 preferred. This parameter controls the stopping behavior of the

convex hull initialization proceedure, with a higher value requiring a better fit before moving to standard estimation.

`sample_edges_at_a_time`
: Defaults to 0. If greater than zero, then this is the number of edges to be updated at once during MCMCMLE. The lower this number is set, the higher the Metropolis Hastings acceptance rate should be. This option will primarily be relevant for large networks.

`parallel`
: Logical indicating whether the weighted MPLE objective and any other operations that can be easily parallelized should be calculated in parallel. Defaults to FALSE. If TRUE, a significant speedup in computation may be possible.

`parallel_statistic_calculation`
: Logical indicating whether network statistics should be calculated in parallel. This will tend to be slower for networks with les than ~30 nodes but may provide a substantial speedup for larger networks.

`cores`
: Numeric value defaulting to 1. Can be set to any number up to the number of threads/cores available on your machine. Will be used to speed up computations if parallel = TRUE.

`use_stochastic_MH`
: A logical indicating whether a stochastic approximation to the h statistics should be used under Metropolis Hastings in-between thinned samples. This may dramatically speed up estimation. Defaults to FALSE. HIGHLY EXPERIMENTAL!

`stochastic_MH_proportion`
: Percentage of dyads/triads to use for approximation, defaults to 0.25.

`slackr_integration_list`
: An optional list object that contains information necessary to provide updates about model fitting progress to a Slack channel (https://slack.com/). This can be useful if models take a long time to run, and you wish to receive updates on their progress (or if they become degenerate). The list object must be of the following form: list(model_name = "descriptive model name", channel = "#yourchannelname", incoming_webhook_url = "https://hooks.slack.com/services/XX/YY/ZZ"). You will need to set up incoming webhook integration for your slack channel and then paste in the URL you get from slack into the incoming_webhook_url field. If all goes well, and the computer you are running the GERGM estimation on has internet access, your slack channel will receive updates when you start estimation, after each lambda/theta parameter update, if the model becomes degenerate, and when it completes running.

`convergence_tolerance`
: Threshold designated for stopping criterion. If the difference of parameter estimates from one iteration to the next all have a p -value (under a paired t-test) greater than this value, the parameter estimates are declared to have converged. Default is 0.5, which is quite conservative.

`MPLE_gain_factor`
: Multiplicative constant between 0 and 1 that controls how far away the initial theta estimates will be from the standard MPLEs via a one step Fisher update. In the case of strongly dependent data, it is suggested to use a value of 0.10. Default is 0.

`acceptable_fit_p_value_threshold`
: A p-value threshold for how closely statistics of observed network conform to statistics of networks simulated from GERGM parameterized by converged final parameter estimates. Default value is 0.05.

`normalization_type`
: If only a raw_network is provided the function will automatically check to determine if all edges fall in the [0,1] interval. If edges are determined to fall outside of this interval, then a trasformation onto the interval may be specified. If "division" is selected, then the data will have a value added to them such that the minimum value is at least zero (if necessary) and then all edge values will be divided by the maximum to ensure that the maximum value is in [0,1]. If "log" is selected, then the data will have a value added to them such that the minimum value is at least zero (if necessary), then 1 will be added to all edge values before they are logged and then divided by the largest value, again ensuring that the resulting network is on [0,1]. Defaults to "log" and need not be set to NULL if providing covariates as it will be ignored.

`transformation_type`
: Specifies how covariates are transformed onto the raw network. When working with heavy tailed data that are not strictly positive, select "Cauchy" to transform the data using a Cauchy distribution. If data are strictly positive and heavy tailed (such as financial data) it is suggested the user select "LogCauchy" to perform a Log-Cauchy transformation of the data. For a tranformation of the data using a Gaussian distribution, select "Gaussian" and for strictly positive raw networks, select "LogNormal". The Default value is "Cauchy".

`estimation_method`
: Simulation method for MCMC estimation. Default is "Metropolis", which allows for the most flexible model specifications, but may also be set to "Gibbs", if the user wishes to use Gibbs sampling.

`downweight_statistics_together`
: Logical specifying whether or not the weights should be applied inside or outside the sum. Default is TRUE and user should not select FALSE under normal circumstances.

`force_x_theta_updates`
: Defaults to 1 where theta estimation is not allowed to converge until thetas have updated for x iterations . Useful when model is not degenerate but simulated statistics do not match observed network well when algorithm stops after first y updates.

`force_x_lambda_updates`
: Defaults to 1 where lambda estimation is not allowed to converge until lambdas have updated for x iterations . Useful when model is not degenerate but simulated statistics do not match observed network well when algorithm stops after first y updates.

`output_directory`
: The directory where you would like output generated by the GERGM estimation procedure to be saved (if output_name is specified). This includes, GOF, trace, and parameter estimate plots, as well as a summary of the estimation procedure and an .Rdata file containing the GERGM object returned by this function. May

be left as NULL if the user would prefer all plots be printed to the graphics device.

`output_name` The common name stem you would like to assign to all objects output by the gergm() function. Default value of NULL will not save any output directly to .pdf files, it will be printed to the console instead. Must be a character string or NULL. For example, if "Test" is supplied as the output_name, then 4 files will be output: "Test_GOF.pdf", "Test_Parameter_Estim ates.pdf", "Test_GERGM_Object.Rdata", "Test_Estimation_Log.txt", and "Test_Trace_Plot.pdf"

`generate_plots` Defaults to TRUE, if FALSE, then no diagnostic or parameter plots are generated.

`verbose` Defaults to TRUE (providing lots of output while model is running). Can be set to FALSE if the user wishes to see less output.

`use_previous_thetas`
Logical, defaults to TRUE. IF TRUE, then at each iteration of covariate parameter updates beyond the first, the MPLE initialization for theta parameters will be skipped, and the previous iteration thetas will be used as a starting point. This option will only work with convex hull initialization. The intuition here is that if MPLE is poor, the previous theta estimates may be a better starting point for convex hull initialization. In some cases this can lead to a very large speedup in estimation. However, this can lead to arbitrarily poor performance in some situations.

`fine_grained_pv_optimization`
Logical indicating whether fine grained proposal variance optimization should be used. This will often slow down proposal variance optimization, but may provide better results. Highly recommended if running a correlation model.

`use_MPLE_only` Logical specifying whether or not only the maximum pseudo likelihood estimates should be obtained. In this case, no simulations will be performed. Default is FALSE.

`start_with_zeros`
Defaults to FALSE. IF TRUE, then no MPLE is used and allendogenous parameters are initialized to zero. This method will still work with convex_hull_proportion.

`stop_for_degeneracy`
When TRUE, automatically stops estimation when degeneracy is detected, even when hyperparameter_optimization is set to TRUE. Defaults to FALSE.

`maximum_number_of_lambda_updates`
Maximum number of iterations of outer MCMC loop which alternately estimates transform parameters and ERGM parameters. In the case that data_transformation = NULL, this argument is ignored. Default is 10.

`maximum_number_of_theta_updates`
Maximum number of iterations within the MCMC inner loop which estimates the ERGM parameters. Default is 100.

`user_specified_initial_thetas`
Optional numeric vector of user specified theta values to be used for initialization (instead of MPLE). This option should not be used exept when MPLE performance is poor. Defaults to NULL.

integration_intervals
: The number of intervals to be used for numerical integration in weighted MPLE and in MPLE for the distribution estimator. Defaults to 150 but may be increased if more accuracy is required.

distribution_mple_regularization_weight
: L2 regularization of the MPLE theta estimates may be necessary to provide an adequate starting position when using the distribtuion estimator. We suggest a value of 0.05, but the optimal L2 penalty will be application specific. Setting to zero removes all L2 regularization.

theta_grid_optimization_list
: Defaults to NULL. This highly experimental feature may allow the user to address model degeneracy arising from a suboptimal theta initialization. It performs a grid search around the theta values calculated via MPLE to select a potentially improved initialization. The runtime complexity of this feature grows exponentially in the size of the grid and number of parameters – use with great care. This feature may only be used if hyperparameter_optimization = TRUE, and if a list object of the following form is provided: list(grid_steps = 2, step_size = 0.5, cores = 2, iteration_fraction = 0.5). grid_steps indicates the number of steps out the grid search will perform, step_size indicates the fraction of the MPLE theta estimate that each grid search step will change by, cores indicates the number of cores to be used for parallel optimization, and iteration_fraction indicates the fraction of the number of MCMC iterations that will be used for each grid point (should be set less than 1 to speed up optimization). In general grid_steps should be smaller the more structural parameters the user wishes to specify. For example, with 5 structural parameters (mutual, ttriads, etc.), grid_steps = 3 will result in a (2*3+1)^5 = 16807 parameter grid search. Again this feature is highly experimental and should only be used as a last resort (after playing with exponential down weighting and the MPLE_gain_factor).

weighted_MPLE
: Defaults to FALSE. Should be used whenever the user is specifying statistics with alpha down weighting. Tends to provide better initialization when downweight_statistics_together = FALSE.

estimate_model
: Logical indicating whether a model should be estimated. Defaults to TRUE, but can be set to FALSE if the user simply wishes to return a GERGM object containing the model specification. Useful for debugging.

optimization_method
: The optimization method used by the 'optim' function in estimating theta and lambda parameter estimates. Defualts to "BFGS", but can also be any one of "L-BFGS-B", "Nelder-Mead", "CG", "SANN", or "Brent". "L-BFGS-B" is preferred for fitting beta correlation models.

...
: Optional arguments, currently unsupported.

Value

A gergm object containing parameter estimates.

Examples

```
## Not run:
```

```
set.seed(12345)
net <- matrix(rnorm(100,0,20),10,10)
colnames(net) <- rownames(net) <- letters[1:10]
formula <- net ~  mutual + ttriads

test <- gergm(formula,
              normalization_type = "division",
              network_is_directed = TRUE,
              use_MPLE_only = FALSE,
              estimation_method = "Metropolis",
              number_of_networks_to_simulate = 40000,
              thin = 1/10,
              proposal_variance = 0.5,
              downweight_statistics_together = TRUE,
              MCMC_burnin = 10000,
              seed = 456,
              convergence_tolerance = 0.01,
              MPLE_gain_factor = 0,
              force_x_theta_updates = 4)

## End(Not run)
```

GOF *Generate Goodness Of Fit plot from a GERGM object.*

Description

Generate Goodness Of Fit plot from a GERGM object.

Usage

```
GOF(GERGM_Object, column_names = NULL, modularity_group_memberships = NULL,
  return_GERGM_Object = FALSE, observed_support = FALSE, ...)
```

Arguments

GERGM_Object	The object returned by the estimation procedure using the GERGM function.
column_names	Optional argument allowing the user to specify names for statistics to be plotted.
modularity_group_memberships	Optional numeric vector of node group memberships indexed from 1, which will be used to calculate network modularities.
return_GERGM_Object	Optional argument to return the GERGM Object that was passed in, but now with additional GOF statistics such as modularity included in the '@simulated_statistics_for_GOF' and '@additional_stats' fields
observed_support	logical indicating whether GOF plots should use observed support. Defaults to FALSE.
...	Additional Arguments can be passed in. Included for eventual compatibility with XERGM package.

Value

A set of box plots where of simulated network statistics centered at the observed value for those statistics and normalized by their standard deviation. This aids in interpretation as the y-axis can be interpreted as the number of simulated-sample standard deviations above or below the observed statistic. Optionally also returns a GERGM object with updated statistics.

`hysteresis` *A function to automatically generate hysteresis plots for all structural parameter estimates.*

Description

Hysteresis plots were introduced by: Snijders, Tom AB, et al. "New specifications for exponential random graph models." Sociological methodology 36.1 (2006): 99-153. They can tell the user about sensitivity of the parameter estimates and whether they should worry about degeneracy.

Usage

```
hysteresis(GERGM_Object, networks_to_simulate = 1000, burnin = 500,
  range = 4, steps = 20, initial_density = 0.2,
  simulation_method = c("Gibbs", "Metropolis"), proposal_variance = 0.1,
  seed = 12345, thin = 1, output_directory = NULL, output_name = NULL,
  parallel = FALSE)
```

Arguments

`GERGM_Object`	A GERGM object returned by the gergm() estimation function.
`networks_to_simulate`	Number of simulations per unique parameter value used in the hysteresis plots. Default is 1000.
`burnin`	Number of samples from the MCMC simulation procedure that will be discarded before drawing the samples used for hysteresis plots. Default is 500.
`range`	The magnitude of the interval over which theta parameter values will be varied for the hysteresis plots. The actual range will be vary from a minimum of theta_value - range * theta_std_error to a maximum of theta_value + range * theta_std_error so the actual parameter ranges will be scaled to the magnitude of the parameter estimate standard errors. Defaults to 4.
`steps`	The number of theta values to simulate above and below the estimated theta value within the given range. The total number of simulations is then = 2 * steps + 1. Defaults to 20.
`initial_density`	The initial network density used in simulations, can range from 0 to 1. Defaults to 0.2.
`simulation_method`	Simulation method for MCMC estimation. Default is "Gibbs" but can also be set to "Metropolis".

proposal_variance
: The variance specified for the Metropolis Hastings simulation method. This parameter is inversely proportional to the average acceptance rate of the M-H sampler and should be adjusted so that the average acceptance rate is approximately 0.25. Default is 0.1.

seed
: Seed used for reproducibility. Default is 123.

thin
: The proportion of samples that are kept from each simulation. For example, thin = 1/200 will keep every 200th network in the overall simulated sample. Default is 1.

output_directory
: The directory where you would like output generated by this function to be saved. If NULL, then the current working directory will be used. Defaults to NULL.

output_name
: The common name stem you would like to assign to all plots generated by this function. If NULL, then no output will be saved to pdf and plots will only be plotted to the graphics device.

parallel
: Logical indicating whether hysteresis plots for each theta parameter should be simulated in parallel. Can greatly reduce runtime, but the computer must have at least as many cores as theta parameters. Defaults to FALSE.

Value

A list object containing network densities for simulated networks.

Examples

```
## Not run:
set.seed(12345)
net <- matrix(rnorm(100,0,20),10,10)
colnames(net) <- rownames(net) <- letters[1:10]
formula <- net ~  mutual + ttriads

test <- gergm(formula,
              normalization_type = "division",
              network_is_directed = TRUE,
              use_MPLE_only = FALSE,
              estimation_method = "Metropolis",
              number_of_networks_to_simulate = 40000,
              thin = 1/10,
              proposal_variance = 0.5,
              downweight_statistics_together = TRUE,
              MCMC_burnin = 10000,
              seed = 456,
              convergence_tolerance = 0.01,
              MPLE_gain_factor = 0,
              force_x_theta_updates = 4)

hysteresis_results <- hysteresis(
    test,
    networks_to_simulate = 1000,
```

```
    burnin = 500,
    range = 2,
    steps = 20,
    simulation_method = "Metropolis",
    proposal_variance = 0.5)

## End(Not run)
```

`hysteresis_plot` *Generate hysteresis plots for theta parameter estimates*

Description

Generate hysteresis plots for theta parameter estimates

Usage

```
hysteresis_plot(hysteresis_output, text_size = 12, ...)
```

Arguments

`hysteresis_output`	The list object output from the hysteresis function.
`text_size`	The base size for axis text. Defaults to 12.
`...`	Additional arguments currently not supported.

`lending_2005` *Logged international lending volumes for 17 large countries from 2005.*

Description

A dataset of logged aggregate public and private lending volumes between 17 large industrialized countries.

Usage

```
lending_2005
```

Format

A matrix with 17 rows and columns and a zero diagonal.

Source

http://arxiv.org/abs/1505.04015

net_exports_2005 *Normalized net exports between 17 large countries from 2005.*

Description

Aggregate net export volume between countries has been normalized to lie between 0 and 1.

Usage

```
net_exports_2005
```

Format

A matrix with 17 rows and columns and a zero diagonal.

Source

http://arxiv.org/abs/1505.04015

parallel_gergm *A Function to estimate a number of GERGMs in parallel, each with its own equation.*

Description

Allows the user to run multiple specifications at once in parallel. All variables (excluding formula_list, observed_network_list, covariate_data_list, network_data_list, cores and generate_plots) be be either specified as a single value or as a vector of values equal to the length of formula_list, if the user wishes to use different values for each specification.

Usage

```
parallel_gergm(formula_list, observed_network_list,
  covariate_data_list = NULL, network_data_list = NULL, cores = 1,
  normalization_type = c("log", "division"), network_is_directed = TRUE,
  use_MPLE_only = FALSE, transformation_type = c("Cauchy", "LogCauchy",
  "Gaussian", "LogNormal"), estimation_method = c("Gibbs", "Metropolis"),
  maximum_number_of_lambda_updates = 10,
  maximum_number_of_theta_updates = 10,
  number_of_networks_to_simulate = 500, thin = 1, proposal_variance = 0.1,
  downweight_statistics_together = TRUE, MCMC_burnin = 100, seed = 123,
  convergence_tolerance = 0.01, MPLE_gain_factor = 0,
  acceptable_fit_p_value_threshold = 0.05, force_x_theta_updates = 1,
  force_x_lambda_updates = 1, output_directory = NULL, output_name = NULL,
  generate_plots = TRUE, verbose = TRUE,
  hyperparameter_optimization = FALSE, stop_for_degeneracy = FALSE,
```

```
  target_accept_rate = 0.25, theta_grid_optimization_list = NULL,
  beta_correlation_model = FALSE, weighted_MPLE = FALSE,
  fine_grained_pv_optimization = FALSE, parallel = FALSE,
  parallel_statistic_calculation = FALSE, cores_per_model = 1,
  use_stochastic_MH = FALSE, stochastic_MH_proportion = 0.25, ...)
```

Arguments

formula_list	A list of formula objects that specifies the relationship between statistics and the observed network for each gergm. See the gergm() documentation for more details.
observed_network_list	A list of observed networks (as numeric matrices to be used with each specification).
covariate_data_list	An optional list of covariate data frames (may include NULL entries if no covariates are needed in some specifications)
network_data_list	An optional list of of lists of network covariates to be included in each specification (one list per specification – may also be left NULL for some specifications). The list object corresponding to each specification must have entries for network covariates named as they appear in the corresponding equation. For example if the user specified a 'netcov(distance)' term, the corresponding list object for that specification would need a $distance entry containing the corresponding matrix object.
cores	The number of cores to be used for parallelization.
normalization_type	If only a raw_network is provided the function will automatically check to determine if all edges fall in the [0,1] interval. If edges are determined to fall outside of this interval, then a trasformation onto the interval may be specified. If "division" is selected, then the data will have a value added to them such that the minimum value is at least zero (if necessary) and then all edge values will be divided by the maximum to ensure that the maximum value is in [0,1]. If "log" is selected, then the data will have a value added to them such that the minimum value is at least zero (if necessary), then 1 will be added to all edge values before they are logged and then divided by the largest value, again ensuring that the resulting network is on [0,1]. Defaults to "log" and need not be set to NULL if providing covariates as it will be ignored.
network_is_directed	Logical specifying whether or not the observed network is directed. Default is TRUE.
use_MPLE_only	Logical specifying whether or not only the maximum pseudo likelihood estimates should be obtained. In this case, no simulations will be performed. Default is FALSE.
transformation_type	Specifies how covariates are transformed onto the raw network. When working with heavy tailed data that are not strictly positive, select "Cauchy" to transform

the data using a Cauchy distribution. If data are strictly positive and heavy tailed (such as financial data) it is suggested the user select "LogCauchy" to perform a Log-Cauchy transformation of the data. For a tranformation of the data using a Gaussian distribution, select "Gaussian" and for strictly positive raw networks, select "LogNormal". The Default value is "Cauchy".

`estimation_method`
: Simulation method for MCMC estimation. Default is "Gibbs" which will generally be faster with well behaved networks but will not allow for exponential down weighting.

`maximum_number_of_lambda_updates`
: Maximum number of iterations of outer MCMC loop which alternately estimates transform parameters and ERGM parameters. In the case that data_transformation = NULL, this argument is ignored. Default is 10.

`maximum_number_of_theta_updates`
: Maximum number of iterations within the MCMC inner loop which estimates the ERGM parameters. Default is 100.

`number_of_networks_to_simulate`
: Number of simulations generated for estimation via MCMC. Default is 500.

`thin`
: The proportion of samples that are kept from each simulation. For example, thin = 1/200 will keep every 200th network in the overall simulated sample. Default is 1.

`proposal_variance`
: The variance specified for the Metropolis Hastings simulation method. This parameter is inversely proportional to the average acceptance rate of the M-H sampler and should be adjusted so that the average acceptance rate is approximately 0.25. Default is 0.1.

`downweight_statistics_together`
: Logical specifying whether or not the weights should be applied inside or outside the sum. Default is TRUE and user should not select FALSE under normal circumstances.

`MCMC_burnin`
: Number of samples from the MCMC simulation procedure that will be discarded before drawing the samples used for estimation. Default is 100.

`seed`
: Seed used for reproducibility. Default is 123.

`convergence_tolerance`
: Threshold designated for stopping criterion. If the difference of parameter estimates from one iteration to the next all have a p -value (under a paired t-test) greater than this value, the parameter estimates are declared to have converged. Default is 0.01.

`MPLE_gain_factor`
: Multiplicative constant between 0 and 1 that controls how far away the initial theta estimates will be from the standard MPLEs via a one step Fisher update. In the case of strongly dependent data, it is suggested to use a value of 0.10. Default is 0.

`acceptable_fit_p_value_threshold`
: A p-value threshold for how closely statistics of observed network conform to statistics of networks simulated from GERGM parameterized by converged final parameter estimates. Default value is 0.05.

`force_x_theta_updates`
: Defaults to 1 where theta estimation is not allowed to converge until thetas have updated for x iterations . Useful when model is not degenerate but simulated statistics do not match observed network well when algorithm stops after first y updates.

`force_x_lambda_updates`
: Defaults to 1 where lambda estimation is not allowed to converge until lambdas have updated for x iterations . Useful when model is not degenerate but simulated statistics do not match observed network well when algorithm stops after first y updates.

`output_directory`
: The directory where you would like output generated by the GERGM estimation procedure to be saved (if output_name is specified). This includes, GOF, trace, and parameter estimate plots, as well as a summary of the estimation procedure and an .Rdata file containing the GERGM object returned by this function. May be left as NULL if the user would prefer all plots be printed to the graphics device.

`output_name`
: The common name stem you would like to assign to all objects output by the gergm function. Default value of NULL will not save any output directly to .pdf files, it will be printed to the console instead. Must be a character string or NULL. For example, if "Test" is supplied as the output_name, then 4 files will be output: "Test_GOF.pdf", "Test_Parameter_Estim ates.pdf", "Test_GERGM_Object.Rdata", "Test_Estimation_Log.txt", and "Test_Trace_Plot.pdf". Must be the same length as the number of specifications or specification_i will be automatically used to distinguish between specifications.

`generate_plots`
: Defaults to TRUE, if FALSE, then no diagnostic or parameter plots are generated.

`verbose`
: Defaults to TRUE (providing lots of output while model is running). Can be set to FALSE if the user wishes to see less output.

`hyperparameter_optimization`
: Logical indicating whether automatic hyperparameter optimization should be used. Defaults to FALSE. If TRUE, then the algorithm will automatically seek to find an optimal burnin and number of networks to simulate, and if using Metropolis Hasings, will attempt to select a proposal variance that leads to a acceptance rate within +-0.05 of target_accept_rate. Furthermore, if degeneracy is detected, the algorithm will attempt to adress the issue automatically. WARNING: This feature is experimental, and may greatly increase runtime. Please monitor console output!

`stop_for_degeneracy`
: When TRUE, automatically stops estimation when degeneracy is detected, even when hyperparameter_optimization is set to TRUE. Defaults to FALSE. SPECIFY SINGLE VALUE, MUST BE CONSTANT ACROSS SPECIFICATIONS.

`target_accept_rate`
: The target Metropolis Hastings acceptance rate. Defaults to 0.25

`theta_grid_optimization_list`
: Defaults to NULL. This highly experimental feature may allow the user to address model degeneracy arising from a suboptimal theta initialization. It

performs a grid search around the theta values calculated via MPLE to select a potentially improved initialization. The runtime complexity of this feature grows exponentially in the size of the grid and number of parameters – use with great care. This feature may only be used if hyperparameter_optimization = TRUE, and if a list object of the following form is provided: list(grid_steps = 2, step_size = 0.5, cores = 2, iteration_fraction = 0.5). grid_steps indicates the number of steps out the grid search will perform, step_size indicates the fraction of the MPLE theta estimate that each grid search step will change by, cores indicates the number of cores to be used for parallel optimization, and iteration_fraction indicates the fraction of the number of MCMC iterations that will be used for each grid point (should be set less than 1 to speed up optimization). In general grid_steps should be smaller the more structural parameters the user wishes to specify. For example, with 5 structural parameters (mutual, ttriads, etc.), grid_steps = 3 will result in a (2*3+1)^5 = 16807 parameter grid search. Again this feature is highly experimental and should only be used as a last resort (after playing with exponential down weighting and the MPLE_gain_factor). SPECIFY SINGLE VALUE, MUST BE CONSTANT ACROSS SPECIFICATIONS.

`beta_correlation_model`
: Defaults to FALSE. If TRUE, then the beta correlation model is estimated. A correlation network must be provided, but all covariates and undirected statistics may be supplied as normal. SPECIFY SINGLE VALUE, MUST BE CONSTANT ACROSS SPECIFICATIONS.

`weighted_MPLE`
: Defaults to FALSE. Should be used whenever the user is specifying statistics with alpha down weighting. Tends to provide better initialization when downweight_statistics_together = FALSE. SPECIFY SINGLE VALUE, MUST BE CONSTANT ACROSS SPECIFICATIONS.

`fine_grained_pv_optimization`
: Logical indicating whether fine grained proposal variance optimization should be used. This will often slow down proposal variance optimization, but may provide better results. Highly recommended if running a correlation model. SPECIFY SINGLE VALUE, MUST BE CONSTANT ACROSS SPECIFICATIONS.

`parallel`
: Logical indicating whether the weighted MPLE objective and any other operations that can be easily parallelized should be calculated in parallel. Defaults to FALSE. If TRUE, a significant speedup in computation may be possible. SPECIFY SINGLE VALUE, MUST BE CONSTANT ACROSS SPECIFICATIONS.

`parallel_statistic_calculation`
: Logical indicating whether network statistics should be calculated in parallel. This will tend to be slower for networks with les than ~30 nodes but may provide a substantial speedup for larger networks. SPECIFY SINGLE VALUE, MUST BE CONSTANT ACROSS SPECIFICATIONS.

`cores_per_model`
: Numeric value defaulting to 1. Can be set to any number up to the number of threads/cores available on your machine. Will be used to speed up computations if parallel = TRUE. Note that this will be the number of croes requested by EACH model, so plan accordingly! SPECIFY SINGLE VALUE, MUST BE CONSTANT ACROSS SPECIFICATIONS.

use_stochastic_MH
: A logical indicating whether a stochastic approximation to the h statistics should be used under Metropolis Hastings in-between thinned samples. This may dramatically speed up estimation. Defualts to FALSE. HIGHLY EXPERIMENTAL! SPECIFY SINGLE VALUE, MUST BE CONSTANT ACROSS SPECIFICATIONS.

stochastic_MH_proportion
: Percentage of dyads/triads to use for approximation, defaults to 0.25. SPECIFY SINGLE VALUE, MUST BE CONSTANT ACROSS SPECIFICATIONS.

...
: Optional arguments, currently unsupported.

Value

A list of gergm objects for each model specified.

Examples

```
## Not run:
set.seed(12345)
net <- matrix(runif(100,0,1),10,10)
colnames(net) <- rownames(net) <- letters[1:10]
node_level_covariates <- data.frame(Age = c(25,30,34,27,36,39,27,28,35,40),
                           Height = c(70,70,67,58,65,67,64,74,76,80),
                           Type = c("A","B","B","A","A","A","B","B","C","C"))
rownames(node_level_covariates) <- letters[1:10]
network_covariate <- net + matrix(rnorm(100,0,.5),10,10)

network_data_list <- list(network_covariate = network_covariate)

formula <- net ~ edges + sender("Age") +
  netcov("network_covariate") + nodematch("Type",base = "A")
formula2 <- net ~ edges +
  netcov("network_covariate") + nodemix("Type",base = "A")

form_list <- list(f1 = formula,
                  f2 = formula2)

testl <- parallel_gergm(formula_list = form_list,
                        observed_network_list = net,
                        covariate_data_list = node_level_covariates,
                        network_data_list = network_data_list,
                        cores = 2,
                        number_of_networks_to_simulate = 10000,
                        thin = 1/100,
                        proposal_variance = 0.1,
                        MCMC_burnin = 5000)

## End(Not run)
```

plot_network *Plots of value-edged networks.*

Description

Generates a visualization of a value-edged network.

Usage

```
plot_network(sociomatrix, threshold = 0.5, save_pdf = FALSE,
  pdf_name = "Test.pdf", output_directory = "./",
  comparison_network = NULL, comparison_names = NULL, seed = NULL,
  white_background = FALSE, show_legend = TRUE, title = "",
  identical_node_positions = FALSE)
```

Arguments

sociomatrix	A square numeric matrix (sociomatrix) with real valued edges (no NA's).
threshold	The threshold for removing edges from the network in order to calculate the positions for the nodes using the Futcherman-Reingold algorithm. The value is multiplied against max(abs(sociomatrix)) to determine the threshold. Defaults to 0.5.
save_pdf	Logical indicating whether the plot should be saved to a PDF.
pdf_name	The name we would like to give to the output file. Be sure to include a ".pdf" extension.
output_directory	The directory where the user would like to output the PDF if save_pdf == TRUE.
comparison_network	An optional argument providing a second square numeric matrix (sociomatrix) with real valued edges (no NA's) to be visually compared to sociomatrix. The second network will be Procrustes transformed so that it appears most similar without changing the relativel positions of nodes. Defaults to NULL.
comparison_names	An optional string vector of length two providing titles for each of the two networks to be compared. Defaults to NULL.
seed	Optional argument to set the seed for the network layout algorithm so that plots look the same across multiple runs. Defaults to NULL but can be a positive integer (eg. 12345).
white_background	Defaults to FALSE. If TRUE, then network is plotted on a white background with black lettering.
show_legend	Logical indicating whether a legend with extremal edge values should be shown. Defaults to TRUE.
title	The title we wish to give our plot.

identical_node_positions
: Logical indicating whether node positions should be fixed to be the same when comparing networks. Defaults to FALSE.

Examples

```
set.seed(12345)
sociomatrix <- matrix(rnorm(400,0,20),20,20)
colnames(sociomatrix) <- rownames(sociomatrix) <- letters[1:20]
plot_network(sociomatrix)
```

---

simulate_networks *A Function to simulate networks from a GERGM with given theta parameters.*

Description

Simulates networks from a GERGM for a given set of parameter values.

Usage

```
simulate_networks(formula, thetas, simulation_method = c("Metropolis",
  "Gibbs"), network_is_directed = TRUE,
  number_of_networks_to_simulate = 500, thin = 1, proposal_variance = 0.1,
  downweight_statistics_together = TRUE, MCMC_burnin = 100, seed = 123,
  GERGM_Object = NULL, return_constrained_networks = FALSE,
  optimize_proposal_variance = FALSE, target_accept_rate = 0.25,
  use_stochastic_MH = FALSE, stochastic_MH_proportion = 1,
  beta_correlation_model = FALSE, distribution_estimator = c("none",
  "rowwise-marginal", "joint"), covariate_data = NULL, lambdas = NULL,
  include_diagonal = FALSE, ...)
```

Arguments

formula
: A formula object that specifies which statistics the user would like to include while simulating the network, and the network the user is providing as the initial network. Currently, the following statistics can be specified: c("edges", "out2stars", "in2stars", "ctriads", "mutual", "ttriads").

thetas
: A vector of theta parameters given in the same order as the formula terms, which the user would like to use to parameterize the model.

simulation_method
: Default is "Metropolis" which allows for exponential down weighting, can also be "Gibbs".

network_is_directed
: Logical specifying whether or not the observed network is directed. Default is TRUE.

number_of_networks_to_simulate
: Number of simulations generated for estimation via MCMC. Default is 500.

thin
: The proportion of samples that are kept from each simulation. For example, thin = 1/200 will keep every 200th network in the overall simulated sample. Default is 1.

proposal_variance
: The variance specified for the Metropolis Hastings simulation method. This parameter is inversely proportional to the average acceptance rate of the M-H sampler and should be adjusted so that the average acceptance rate is approximately 0.25. Default is 0.1.

downweight_statistics_together
: Logical specifying whether or not the weights should be applied inside or outside the sum. Default is TRUE and user should not select FALSE under normal circumstances.

MCMC_burnin
: Number of samples from the MCMC simulation procedure that will be discarded before drawing the samples used for estimation. Default is 100.

seed
: Seed used for reproducibility. Default is 123.

GERGM_Object
: Optional argument allowing the user to supply a GERGM object output by the gergm() estimation function in order to simulate further networks. Defaults to NULL. If a GERGM object is provided, any user specified parameter values will be ignored and the final parameter estimates from the gergm() function will be used instead. When using this option, the following terms must still be specified: number_of_networks_to_simulate, thin, and MCMC_burnin. proposal_variance may also be specified, or if set equal to NULL, then the proposal variance from parameter estimation will be used instead (this option is likely preferred in most situations).

return_constrained_networks
: Logical argument indicating whether simulated networks should be transformed back to observed scale or whether constrained [0,1] networks should be returned. Defaults to FALSE, in which case networks are returned on observed scale.

optimize_proposal_variance
: Logical indicating whether proposal variance should be optimized if using Metropolis Hastings for simulation. Defaults to FALSE.

target_accept_rate
: Defaults to 0.25, can be used to optimize Metropolis Hastings simulations.

use_stochastic_MH
: A logical indicating whether a stochastic approximation to the h statistics should be used under Metropolis Hastings in-between thinned samples. This may dramatically speed up estimation. Defaults to FALSE. HIGHLY EXPERIMENTAL!

stochastic_MH_proportion
: Percentage of dyads/triads to use for approximation, defaults to 0.25

beta_correlation_model
: Defaults to FALSE. If TRUE, then the beta correlation model is estimated. A correlation network must be provided, but all covariates and undirected statistics may be supplied as normal.

distribution_estimator
: Provides an option to estimate the structure of row-wise marginal and joint distribtuions using a uniform-dirichlet proposal distribution. THIS FEATURE IS

EXPERIMENTAL. Defaults to "none", in which case a normal GERGM is estimated, but can be set to one of "rowwise-marginal" and "joint" to propose either row-wsie marginal distribtuions or joint distributions. If an option other than "none" is selected, the beta correlation model will be turned off, estimation will automatically be set to Metropolis, no covariate data will be allowed, and the network will be set to directed. Furthermore, a "diagonal" statistic will be added to the model which simply records the sum of the diagonal of the network. The "mutual" statistic will also be adapted to include the diagonal elements. In the future, more statistics which take account of the network diagonal will be included.

`covariate_data` A data frame containing node level covariates the user wished to transform into sender or receiver effects. It must have row names that match every entry in colnames(raw_network), should have descriptive column names. If left NULL, then no sender or receiver effects will be added.

`lambdas` A vector of lambda parameters given in the same order as the formula terms, which the user would like to use to parameterize the model. Covariate effects should be specified after endogenous effects.

`include_diagonal`
Logical indicating whether the diagonal should be included in the estimation proceedure. If TRUE, then a "diagonal" statistic is added to the model. Defaults to FALSE.

`...` Optional arguments, currently unsupported.

Value

A list object containing simulated networks and parameters used to specify the simulation. See the $MCMC_Output field for simulated networks. If GERGM_Object is provided, then a GERGM object will be returned instead.

Examples

```
## Not run:
set.seed(12345)
net <- matrix(runif(100),10,10)
diag(net) <- 0
colnames(net) <- rownames(net) <- letters[1:10]
formula <- net ~ edges + ttriads + in2stars

test <- simulate_networks(formula,
 thetas = c(0.6,-0.8),
 lambdas = 0.2,
 number_of_networks_to_simulate = 100,
 thin = 1/10,
 proposal_variance = 0.5,
 MCMC_burnin = 100,
 seed = 456)

# preferred method for specifying a null model
formula <- net ~ edges(method = "endogenous")
```

```
test <- simulate_networks(
 formula,
 thetas = 0,
 number_of_networks_to_simulate = 100,
 thin = 1/10,
 proposal_variance = 0.5,
 MCMC_burnin = 100,
 seed = 456)

## End(Not run)
```

`Thin_Statistic_Samples` *A function to thin out simulated GOF statistics so that they have an MCMC autocorrelation of less tahn 0.01.*

Description

Takes as input the '@simulated_statistics_for_GOF' field of the GERGM object (a data.frame) and returns the same data frame but now thinned to reduce autocorrelation in the samples. Useful for exactly replicating the statistics used in the GOF() function.

Usage

```
Thin_Statistic_Samples(statistics)
```

Arguments

`statistics`	A data.frame stored in the '@simulated_statistics_for_GOF' field of the GERGM object.

Value

A data.frame that has been thinned to reduce autocorrelation.

`Trace_Plot` *Generate trace plot of network density from a GERGM object.*

Description

Generate trace plot of network density from a GERGM object.

Usage

```
Trace_Plot(GERGM_Object)
```

Arguments

GERGM_Object	The object returned by the estimation procedure using the GERGM function.

Value

A trace plot of network density.

Index