Package 'mlergm'

July 23, 2025

Title Multilevel Exponential-Family Random Graph Models

Version 0.8.1

Description Estimates exponential-family random graph models for multilevel network data, assuming the multilevel structure is observed. The scope, at present, covers multilevel models where the set of nodes is nested within known blocks. The estimation method uses Monte-Carlo maximum likelihood estimation (MCMLE) methods to estimate a variety of canonical or curved exponential family models for binary random graphs. MCMLE methods for curved exponential-family random graph models can be found in Hunter and Handcock (JCGS, 2006). The package supports parallel computing, and provides methods for assessing goodness-of-fit of models and visualization of networks.

Depends R (>= 4.0.0), ergm (>= 4.2.2), network (>= 1.17.2)

Imports parallel, Matrix, stringr, stats, GGally, ggplot2, cowplot, reshape2, plyr, methods, graphics, lpSolve, sna, statnet.common

Suggests RColorBrewer, knitr, rmarkdown

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.2.1

NeedsCompilation no

VignetteBuilder knitr

Author Jonathan Stewart [cre, aut], Michael Schweinberger [ctb]

Maintainer Jonathan Stewart <jrstewart@fsu.edu>

Repository CRAN

Date/Publication 2025-05-22 15:50:10 UTC

Contents

`classes` *Polish school classes data set.*

Description

The Polish school classes data set `classes` is a subset of a larger data set which was generated as part of a Polish study on adolescent youth. The network data, obtained via a nomination processes, results in a binary, directed random graph where a directed edge from i to j indicates that student i nominated student j as a playmate. A further description of the data as well as a demonstration of an analysis with curved ERGMs can be found in Stewart, Schweinberger, Bojanowski, and Morris (2018).

Usage

```
data(classes)
```

Format

An `mlnet` object.

Details

A dataset containing network data for 9 school classes as part of a Polish educational study. The nodes of the network are students with nodal covariate `sex` and known class membership of the students.

References

Dolata, R. (ed). (2014). Czy szkoła ma znaczenie? Zróżnicowanie wyników nauczania po pierwszym etapie edukacyjnym oraz jego pozaszkolne i szkolne uwarunkowania. Vol. 1. Warsaw: Instytut Badań Edukacyjnych.

Dolata, R. and Rycielski, P. (2014). Wprowadzenie: założenia i cele badania szkolnych uwarunkowań efektywności kształcenia SUEK.

Stewart, J., Schweinberger, M., Bojanowski, M., and M. Morris (2019). Multilevel network data facilitate statistical inference for curved ERGMs with geometrically weighted terms. Social Networks, to appear.

compute_CI *Multilevel Exponential-Family Random Graph Models*

Description

This function estimates an exponential-family random graph model for multilevel network data. At present, `mlergm` covers network data where the set of nodes is nested within known blocks (see, e.g., Schweinberger and Handcock, 2015). An example is groups of students nested within classrooms, which is covered in the `classes` data set. It is assumed that the node membership, that to which block each node is associated, is known (or has been previously estimated).

Usage

```
compute_CI(object, alpha)

mlergm(
  form,
  node_memb,
  parameterization = "standard",
  options = set_options(),
  theta_init = NULL,
  verbose = 0,
  eval_loglik = TRUE,
  seed = NULL
)

## S3 method for class 'mlergm'
print(x, ...)

## S3 method for class 'mlergm'
summary(object, ...)

## S3 method for class 'mlergm'
vcov(object, ...)
```

Arguments

`object`	An object of class `mlergm`, probably produced by `mlergm`.
`alpha`	Desired significance level for the confidence interval.
`form`	Formula of the form: `network ~ term1 + term2 + ...`; allowable model terms are a subset of those in R package ergm, see `ergm.terms`.
`node_memb`	Vector (length equal to the number of nodes in the network) indicating to which block or group the nodes belong. If the network provided in `form` is an object of class `mlnet`, then `node_memb` can be exctracted directly from the network and need not be provided.

`parameterization`
: Parameterization options include 'standard', 'offset', or 'size'.
  - 'standard' : Does not adjust the individual block parameters for size.
  - 'offset' : The offset parameterization uses edge and mutual offsets along the lines of Krivitsky, Handcock, and Morris (2011) and Krivitsky and Kolaczyk (2015). The edge parameter is offset by $-logn(k)$ and the mutual parameter is offset by $+logn(k)$, where $n(k)$ is the size of the kth block.
  - 'size' : Multiplies the block parameters by $logn(k)$, where $n(k)$ is the size of the kth block.

`options`
: See `set_options` for details.

`theta_init`
: Parameter vector of initial estimates for theta to be used.

`verbose`
: Controls the level of output. A value of `0` corresponds to no output, except for warnings; a value of `1` corresponds to minimal output, and a value of `2` corresponds to full output.

`eval_loglik`
: (Logical `TRUE` or `FALSE`) If set to `TRUE`, the bridge estimation procedure of Hunter and Handcock (2006) is used to estimate the loglikelihood for BIC calculations, otherwise the loglikelihood and therefore the BIC is not estimated.

`seed`
: For reproducibility, an integer-valued seed may be specified.

`x`
: An object of class `mlergm`, probably produced by `mlergm`.

`...`
: Additional arguments to be passed if necessary.

Details

The estimation procedures performs Monte-Carlo maximum likelihood for the specified ERGM using a version of the Fisher scoring method detailed by Hunter and Handcock (2006). Settings governing the MCMC procedure (such as `burnin`, `interval`, and `sample_size`) as well as more general settings for the estimation procedure can be adjusted through `set_options`. The estimation procedure uses the the stepping algorithm of Hummel, et al., (2012) for added stability.

Value

`mlergm` returns an object of class `mlergm` which is a list containing:

`theta`
: Estimated parameter vector of the exponential-family random graph model.

`between_theta`
: Estimated parameter vector of the between group model.

`se`
: Standard error vector for theta.

`vcovmat`
: Variance-covariance matrix of the sufficient statistics of the model. Also the Fisher Information matrix.

`between_se`
: Standard error vector for between_theta.

`pvalue`
: A vector of p-values for the estimated parameter vector.

`between_pvalue`
: A vector of p-values for the estimated parameter vector.

`logLikval`
: The loglikelihood for at the estimated MLE.

`bic`
: The BIC for the estimated model.

`mcmc_chain`	The MCMC sample used in the final estimation step, which can be used to diagnose non-convergence.
`estimation_status`	Indicator of whether the estimation procedure had `succcess` or `failed`.
`parameterization`	The model parameterization (either `standard` or `offset`).
`etamap`	Object defining the evaluation of the canonical parameters. See `eta` for more details.
`formula`	The model formula.
`network`	The network for which the model is estimated.
`node_memb`	Vector indicating to which group or block the nodes belong.
`size_quantiles`	The quantiles of the block sizes.

Methods (by generic)

- `print(mlergm)`: Print method for objects of class `mlergm`. Indicates whether the model was succesfully estimated, as well as the model formula provided.
- `summary(mlergm)`: Prints a summary of the estimated `mlergm` model.
- `vcov(mlergm)`: Extracts the estimated Fisher information matrix of the model from an estimated `mlergm` model. When the model is a canonical exponential family (i.e., no curved terms such as gwesp or gwdegree), this is equal to the variance-covariance matrix of the canonical statistics. In the case that specified model is a curved exponential family, the Fisher information matrix is a transformation of the variance-covariance matrix of the exponential family and is given by Equation (3.2) of Hunter & Handcock (Journal of Computational and Graphical Statistics, 2006, DOI: 10.1198/106186006X133069).

Functions

- `compute_CI()`: Constructs a confidence interval at a desird significance level for an estimated parameter vector.

References

Schweinberger, M. and Stewart, J. (2019) Concentration and consistency results for canonical and curved exponential-family random graphs. The Annals of Statistics, to appear.

Schweinberger, M. and Handcock, M. S. (2015). Local dependence in random graph models: characterization, properties and statistical inference. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 77(3), 647-676.

Hunter, D. R., and Handcock, M. S. (2006). Inference in curved exponential family models for networks. Journal of Computational and Graphical Statistics, 15(3), 565-583.

Hummel, R. M., Hunter, D. R., and Handcock, M. S. (2012). Improving simulation-based algorithms for fitting ERGMs. Journal of Computational and Graphical Statistics, 21(4), 920-939.

Krivitsky, P. N., Handcock, M. S., & Morris, M. (2011). Adjusting for network size and composition effects in exponential-family random graph models. Statistical methodology, 8(4), 319-339.

Krivitsky, P.N, and Kolaczyk, E. D. (2015). On the question of effective sample size in network modeling: An asymptotic inquiry. Statistical science: a review journal of the Institute of Mathematical Statistics, 30(2), 184.

Hunter D., Handcock M., Butts C., Goodreau S., and Morris M. (2008). ergm: A Package to Fit, Simulate and Diagnose Exponential-Family Models for Networks. Journal of Statistical Software, 24(3), 1-29.

Butts, C. (2016). sna: Tools for Social Network Analysis. R package version 2.4. https://CRAN.R-project.org/package=sna.

Butts, C. (2008). network: a Package for Managing Relational Data in R. Journal of Statistical Software, 24(2).

Stewart, J., Schweinberger, M., Bojanowski, M., and M. Morris (2019). Multilevel network data facilitate statistical inference for curved ERGMs with geometrically weighted terms. Social Networks, 59, 98-119.

Schweinberger, M., Krivitsky, P. N., Butts, C.T. and J. Stewart (2018). Exponential-family models of random graphs: Inference in finite-, super-, and infinite-population scenarios. https://arxiv.org/abs/1707.04800

Hunter, D. R., and Handcock, M. S. (2006). Inference in curved exponential family models for networks. Journal of Computational and Graphical Statistics, 15(3), 565-583.

See Also

gof.mlergm, mlnet

Examples

```
### Load the school classes data-set
data(classes)

# Estimate a curved multilevel ergm model with offset parameter
# Approximate run time (2 cores): 1.29m, Run time (5 cores): 1.01m
model_est <- mlergm(classes ~ edges + mutual + nodematch("sex") +  gwesp(fixed = FALSE),
                    seed = 123,
                    options = set_options(number_cores = 2))

# To access a summary of the fitted model, call the 'summary' function
summary(model_est)

# Goodness-of-fit can be run by calling the 'gof.mlergm' method
# Approximate run time (2 cores): 32.7s, Run time (5 cores): 18.4s
gof_res <- gof(model_est, options = set_options(number_cores = 2))
plot(gof_res, cutoff = 15)

# Additional information can be obtained by setting verbose = 1,2.
# Approximate run time (2 cores): 6.7s, Run time (5 cores): 5.6s
model_est <- mlergm(classes ~ edges + mutual + nodematch("sex"),
                    seed = 123,
                    verbose = 2,
                    options = set_options(number_cores = 2))
```

`eta` *Extracts the natural parameters from an estimated* `mlergm` *model. When the model is a canonical exponential family, this is the the identity mapping from the parameter vector. However, when the model is a curved exponential family this returns the resulting canonical parameters of the curved exponential family.*

Description

Extracts the natural parameters from an estimated `mlergm` model. When the model is a canonical exponential family, this is the the identity mapping from the parameter vector. However, when the model is a curved exponential family this returns the resulting canonical parameters of the curved exponential family.

Usage

```
eta(object, block = NULL)
```

Arguments

`object`	An object of class `mlergm`, probably produced by `mlergm`.
`block`	Block identifier which should be an element of node_memb in the network.

References

Hunter, D. R., and Handcock, M. S. (2006). Inference in curved exponential family models for networks. Journal of Computational and Graphical Statistics, 15(3), 565-583.

`extract_block` *Multilevel Network*

Description

Function creates a multilevel network object of class `mlnet`. The object inherits the `network` class, with additional information concerning the multilevel structure.

Usage

```
extract_block(net, block)

mlnet(network, node_memb, directed = FALSE)

## S3 method for class 'mlnet'
plot(
  x,
```

```
  node_size = 2.5,
  palette = NULL,
  memb_colors = NULL,
  arrow.gap = 0.015,
  arrow.size = 4,
  color_legend_title = "",
  legend = TRUE,
  legend.position = "right",
  layout_type = "kamadakawai",
  ...
)
```

Arguments

net	An object of class mlnet, possibly produced by mlnet or simulate_mlnet.
block	Block identifier which should be an element of node_memb in the network.
network	Either a network object, an adjacency matrix, or an edge list.
node_memb	Vector (length equal to the number of nodes in the network) indicating to which block or group the nodes belong.
directed	(TRUE or FALSE) Indicates whether the supplied network is directed or undirected. Default is FALSE.
x	An object of class mlnet, possibly produced by mlnet or simulate_mlnet.
node_size	Controls the size of nodes.
palette	If package RColorBrewer is installed, then the name of an R color brewer pallete can be specified and used for the block colors. See brewer.pal for details on RColorBrewer palletes.
memb_colors	Specifies the named colors to be used for the membership colors.
arrow.gap	(Directed graphs only) Controls the amount of space between arrowheads and the nodes.
arrow.size	(Directed graphs only) Controls the size of the arrowhead.
color_legend_title	Name for the node color legend title.
legend	(TRUE or FALSE) Controls whether the block membership legend is printed.
legend.position	The position of the legend in the plot. Defaults to the "right" position.
layout_type	Viable layout options. See gplot.layout for options.
...	Additional arguments to be passed to ggnet2.

Details

The mlnet function creates an object of class mlnet which is used to access methods designed specifically for multilevel networks, including visualization methods as well as direct interface with some of the main functions, such as mlergm. Presently, the mlnet function and object class cover multilevel structure where the set of nodes is nested within known block structure.

Value

mlnet returns an object of class mlnet which inherits the network class, with the additional vector attribute node_memb, which encodes the block membership of the multilevel netwrok.

Methods (by generic)

- plot(mlnet): Plots network objects of type mlnet.

Functions

- extract_block(): Extracts a specified block subgraph from a network object of type mlnet.

Examples

```
# Show how the sampson dataset can be turned into an mlnet object
data(sampson)
net <- mlnet(samplike, get.vertex.attribute(samplike, "group"))
```

---

gof.mlergm *Evaluate the goodness-of-fit of an estimated model.*

Description

Performs a Goodness-of-Fit procedure along the lines of Hunter, Goodreau, and Handcock (2008). Statistics are simulated from an fitted mlergm object, which can then be plotted and visualized. An example is given in the documentation of mlergm.

Usage

```
## S3 method for class 'mlergm'
gof(object, ..., options = set_options(), seed = NULL, gof_form = NULL)
```

Arguments

object	An object of class mlergm, likely produced by function mlergm.
...	Additional arguments to be passed if necessary.
options	See set_options for details.
seed	A seed to be provided to ensure reproducibility of results.
gof_form	A formula object of the form ~ term1 + term2 + ... for statistics to compute for the GOF procedure.

Value

gof.mlergm returns an object of class gof_mlergm which is a list containing:

obs_stats	The GOF statistic values of the observed network.
gof_stats	The GOF statistic values simulated from the the estimated mlergm objeect provided.

References

Hunter, D. R., Goodreau, S. M., and Handcock, M. S. (2008). Goodness of fit of social network models. Journal of the American Statistical Association, 103(481), 248-258.

See Also

plot.gof_mlergm

---

is.gof_mlergm *Check if object is of class* gof_mlergm

Description

Function checks if a provided object is of class gof_mlergm (see gof.mlergm for details).

Usage

```
is.gof_mlergm(x)
```

Arguments

x	An object to be checked.

Value

TRUE if the provided object x is of class gof_mlergm, FALSE otherwise.

See Also

mlergm, gof.mlergm

---

is.inCHv3.9 *Determine whether a vector is in the closure of the convex hull of some sample of vectors*

Description

is.inCH returns TRUE if and only if p is contained in the convex hull of the points given as the rows of M. If p is a matrix, each row is tested individually, and TRUE is returned if all rows are in the convex hull.

Usage

```
is.inCHv3.9(p, M, verbose = FALSE, ...)
```

Arguments

p	A d-dimensional vector or a matrix with d columns
M	An r by d matrix. Each row of M is a d-dimensional vector.
verbose	A logical vector indicating whether to print progress
...	arguments passed directly to linear program solver

Details

The d-vector p is in the convex hull of the d-vectors forming the rows of M if and only if there exists no separating hyperplane between p and the rows of M. This condition may be reworded as follows:

Letting $q = (1p')'$ and $L = (1M)$, if the maximum value of $z'q$ for all z such that $z'L \leq 0$ equals zero (the maximum must be at least zero since z=0 gives zero), then there is no separating hyperplane and so p is contained in the convex hull of the rows of M. So the question of interest becomes a constrained optimization problem.

Solving this problem relies on the package lpSolve to solve a linear program. We may put the program in "standard form" by writing $z = a - b$, where a and b are nonnegative vectors. If we write $x = (a'b')'$, we obtain the linear program given by:

Minimize $(-q'q')x$ subject to $x'(L - L) \leq 0$ and $x \geq 0$. One additional constraint arises because whenever any strictly negative value of $(-q'q')x$ may be achieved, doubling x arbitrarily many times makes this value arbitrarily large in the negative direction, so no minimizer exists. Therefore, we add the constraint $(q' - q')x \leq 1$.

This function is used in the "stepping" algorithm of Hummel et al (2012).

Value

Logical, telling whether p is (or all rows of p are) in the closed convex hull of the points in M.

References

- Hummel, R. M., Hunter, D. R., and Handcock, M. S. (2012), Improving Simulation-Based Algorithms for Fitting ERGMs, Journal of Computational and Graphical Statistics, 21: 920-939.

---

is.mlergm *Check if the object is of class* mlergm

Description

Function checks if a provided object is of class mlergm (see mlergm for details).

Usage

```
is.mlergm(x)
```

Arguments

x	An objected to be checked.

Value

TRUE if the provided object x is of class mlergm, FALSE otherwise.

See Also

mlergm

---

is.mlnet *Check if object is of class* mlnet

Description

Function checks if a provided object is of class mlnet (see mlnet for details).

Usage

```
is.mlnet(x)
```

Arguments

x	An object to be checked.

Value

TRUE if the provided object x is of class mlnet, FALSE otherwise.

See Also

mlnet

`plot.gof_mlergm` *Plot goodness-of-fit results*

Description

Produces goodness-of-fit plots for a `gof_mlergm` object in order to visualize and assess the fit of an estimated model produced by `mlergm`.

Usage

```
## S3 method for class 'gof_mlergm'
plot(
  x,
  ...,
  individual_plots = FALSE,
  save_plots = FALSE,
  show_plots = TRUE,
  width = 8,
  height = 4.5,
  cutoff = NULL,
  x_labels = NULL,
  x_angle = 0,
  x_axis_label = NULL,
  y_axis_label = "Count",
  plot_title = "",
  title_size = 18,
  axis_label_size = 14,
  axis_size = 10,
  line_size = 1,
  x_axis_label_size = NULL,
  y_axis_label_size = NULL,
  x_axis_size = NULL,
  y_axis_size = NULL,
  pretty_x = TRUE
)
```

Arguments

`x`	An object of class `gof_mlergm`, produced by `gof.mlergm`.
`...`	Additional argument to be passed if necessary.
`individual_plots`	(Logical `TRUE` or `FALSE`) If `TRUE`, individual gof plots are produced. Defaults to `FALSE`.
`save_plots`	(Logical `TRUE` or `FALSE`) If `TRUE`, the individual GOF plots are saved.
`show_plots`	(Logical `TRUE` or `FALSE`) If `TRUE`, the plots are printed to the screen, and if `FALSE` no plots are displayed. This may be helpful when the only desire is to save the individual GOF plots.

width	If save_plots == TRUE, controls the plot width dimension saved.
height	If save_plots == TRUE, controls the plot height dimension saved.
cutoff	For statistics that are distributions (e.g., degree distributions), specifies a cutoff point. Dimensions past the cutoff are ignored and not plotted.
x_labels	Character vector specifying the statistic names or labels.
x_angle	Adjusts the angle of the x axis tick labels (typically the statistic names).
x_axis_label	Label for the x axis.
y_axis_label	Label for the y aixs.
plot_title	Title for the plot.
title_size	Font size for the plot title.
axis_label_size	Font size for the axis labels. Individual axes label sizes can be changed using x_axis_label_size and y_axis_label_size which are detailed below.
axis_size	Font size for the axis tick labels. Individual axes tick label sizes can be changed using x_axis_size and y_axis_size which are detailed below.
line_size	(Numeric, non-negative) If line_size is positive, then a red line will be plotted to indicate the observed network value of the statistic. If line_size is equal to zero, then the observed data line will not be plotted.
x_axis_label_size	The font size of the x axis label. When NULL, axis_label_size is used. Defaults to NULL.
y_axis_label_size	The font size of the y axis label. When NULL, axis_label_size is used. Defaults to NULL.
x_axis_size	The font size of the x axis tick labels. When NULL, axis_size is used. Defaults to NULL.
y_axis_size	The font size of the y acis tick labels. When NULL, axis_size is used. Defaults to NULL.
pretty_x	(Logical TRUE or FALSE) If set to TRUE, the link[base]{pretty} function will be called to format the x-axis breaks. This can be useful for when the x-axis range is large.

`print.gof_mlergm` *Print summary of a* `gof_mlergm` *object.*

Description

Prints a formatted summary output for `gof_mlergm` object which was produced by `gof.mlergm`.

Usage

```
## S3 method for class 'gof_mlergm'
print(x, ...)
```

Arguments

x	An object of class gof_mlergm, probably produced by gof.mlergm.
...	Additional arguments to be passed if necessary.

See Also

gof.mlergm

---

set_options *Set and adjust options and settings.*

Description

Function allows for specification of options and settings for simulation and estimation procedures.

Usage

```
set_options(
  burnin = 1e+05,
  interval = 2000,
  sample_size = 1000,
  NR_tol = 1e-04,
  NR_max_iter = 50,
  MCMLE_max_iter = 10,
  do_parallel = TRUE,
  number_cores = detectCores(all.tests = FALSE, logical = TRUE) - 1,
  adaptive_step_len = TRUE,
  step_len_multiplier = 0.5,
  step_len = 1,
  bridge_num = 10,
  bridge_burnin = 10000,
  bridge_interval = 500,
  bridge_sample_size = 5000
)
```

Arguments

burnin	The burnin length for MCMC chains.
interval	The sampling interval for MCMC chains.
sample_size	The number of points to sample from MCMC chains for the MCMLE procedure.
NR_tol	The convergence tolerance for the Newton-Raphson optimization (implemented as Fisher scoring).
NR_max_iter	The maximum number of Newton-Raphson updates to perform.
MCMLE_max_iter	The maximum number of MCMLE steps to perform.
do_parallel	(logical) Whether or not to use parallel processesing (defaults to TRUE).

`number_cores` The number of parallel cores to use for parallel computations.

`adaptive_step_len`
(logical) If `TRUE`, an adaptive steplength procedure is used for the Newton-Raphson procedure. Arguments `NR_step_len` and `NR_step_len_multiplier` are ignored when `adaptive_step_len` is `TRUE`.

`step_len_multiplier`
The step_len adjustment multplier when convergence fails.

`step_len` The step length adjustment default to be used for the Newton-Raphson updates.

`bridge_num` The number of bridges to use for likelihood computations.

`bridge_burnin` The burnin length for the bridge MCMC chain for approximate likelihood computation.

`bridge_interval`
The sampling interval for the brdige MCMC chain for approximate likelihood computation.

`bridge_sample_size`
The number of points to sample from the bridge MCMC chain for approximate likelihood computation.

Details

The main simulation settings are `burnin`, `interval`, and `sample_size`. For estimation of the log-likelihood value, options include `bridge_num` which controls the number of bridges to be used for approximating the loglikelihood (see, e.g., Hunter and Handcock (2006) for a discussion). The main estimation settings and options include `NR_tol`, `NR_max_iter`, `MCMLE_max_iter`, `adaptive_step_len`, and `step_len`. Parameters `NR_tol` and `NR_max_iter` control the convergence tolerance and maximum number of iterations for the Newton-Raphson, or Fisher scoring, optimization. When the L2 norm of the incremenet in the Newton-Raphson procedure is under the specified tolerance `NR_tol` convergence is reached; and, no more than `NR_max_iter` iterations are performed. The MCMLE procedure uses the stepping algorithn of Hummel, et al., (2012) to give stabiity to the estimation procedure. Each MCMLE iteration draws samples from an MCMC chain, and `MCMLE_max_iter` controls how many iterations are performed before termination. Most functions support parallel computing for efficiency; by default `do_parallel` is `TRUE`. The number of computing cores can be adjusted by `number_cores`, and the default is one less than the number of cores available.

References

Hunter, D. R., and Handcock, M. S. (2006). Inference in curved exponential family models for networks. Journal of Computational and Graphical Statistics, 15(3), 565-583.

Hummel, R. M., Hunter, D. R., and Handcock, M. S. (2012). Improving simulation-based algorithms for fitting ERGMs. Journal of Computational and Graphical Statistics, 21(4), 920-939.

`simulate_mlnet` *Simulate a multilevel network*

Description

Function simulates a multilevel network by specifying a network size, node block memberships, and within-block and between-block models. The function currently only suppports block-models where between-block edges are dyad-independent.

Usage

```
simulate_mlnet(
  form,
  node_memb,
  theta,
  parameterization = "standard",
  seed = NULL,
  between_form = NULL,
  between_theta = NULL,
  between_prob = NULL,
  options = set_options()
)
```

Arguments

`form`	A `formula` object of the form `network ~ model terms` which specifies how the within-block subgraphs are modeled.
`node_memb`	Vector of node block memberships.
`theta`	A vector of model parameters (coefficients) for the ERGM governing the within-subgraph edges.
`parameterization`	Parameterization options include 'standard', 'offset', or 'size'. • 'standard' : Does not adjust the individual block parameters for size. • 'offset' : The offset parameterization uses edge and mutual offsets along the lines of Krivitsky, Handcock, and Morris (2011) and Krivitsky and Kolaczyk (2015). The edge parameter is offset by $-logn(k)$ and the mutual parameter is offset by $+logn(k)$, where $n(k)$ is the size of the kth block. • 'size' : Multiplies the block parameters by $logn(k)$, where $n(k)$ is the size of the kth block.
`seed`	Seed to be provided for reproducibility.
`between_form`	A `formula` object of the form `~ model terms` which specifies how the within-block subgraphs are modeled.
`between_theta`	A vector of model parameters (coefficients) for the ERGM governing the between-subgraph edges.

between_prob	A probability which specifies how edges between blocks are governerd. An ERGM (between_form and between_theta) cannot be specified together with between_prob.
options	Use set_options to change the simulation options. Note that some options are only valid for estimation using mlergm.

Details

Simulation of multilevel block networks is done with a Monte-Carlo Markov chain (MCMC) and can be done in parallel where set_options can be used to adjust the simulation settings (such as burnin, interval, and sample_size). Each within-block subgraph is given its own Markov chain, and so these settings are the settings to be used for each within-block chain.

Value

simulate_mlnet returns an objects of class mlnet.

Examples

```
# Create a K = 2 block network with edge + gwesp term
net <- simulate_mlnet(form = network.initialize(30, directed = FALSE) ~ edges + gwesp,
                      node_memb = c(rep(1, 15), rep(2, 15)),
                      theta = c(-3, 0.5, 1.0),
                      between_prob = 0.01,
                      options = set_options(number_cores = 2, burnin = 2000))

# Simulate a K = 2 block directed network, specifying a formula for between edges
net <- simulate_mlnet(form = network.initialize(30, directed = TRUE) ~ edges + gwesp,
                      node_memb = c(rep(1, 15), rep(2, 15)),
                      theta = c(-3, 0.5, 1.0),
                      between_form = ~ edges + mutual,
                      between_theta = c(-4, 2),
                      options = set_options(number_cores = 2, burnin = 2000))
```

Index