

# Package ‘latentnet’

May 8, 2026

**Version** 2.12.0

**Date** 2025-09-11

**Title** Latent Position and Cluster Models for Statistical Networks

**Description** Fit and simulate latent position and cluster models for statistical networks. See Krivitsky and Handcock (2008) <[doi:10.18637/jss.v024.i05](https://doi.org/10.18637/jss.v024.i05)> and Krivitsky, Handcock, Raftery, and Hoff (2009) <[doi:10.1016/j.socnet.2009.04.001](https://doi.org/10.1016/j.socnet.2009.04.001)>.

**Depends** R (>= 3.6.0), network (>= 1.19.0), ergm (>= 4.10.1)

**Imports** sna (>= 2.8), mvtnorm (>= 1.3-3), abind (>= 1.4-8), coda (>= 0.19-4.1), statnet.common (>= 4.12.0), tools

**Suggests** snowFT (>= 1.6-1), rgl (>= 1.3.24), heplots (>= 1.7.8), rlecuyer (>= 0.3-8), covr (>= 3.6.4), KernSmooth

**SystemRequirements** GNU make

**RdMacros** ergm

**BugReports** <https://github.com/statnet/latentnet/issues>

**License** GPL-3 + file LICENSE

**URL** <https://statnet.org>

**Encoding** UTF-8

**RoxygenNote** 7.3.2.9000

**NeedsCompilation** yes

**Author** Pavel N. Krivitsky [aut, cre] (ORCID:  
<<https://orcid.org/0000-0002-9101-3362>>),  
Mark S. Handcock [aut],  
Susan M. Shortreed [ctb],  
Jeremy Tantrum [ctb],  
Peter D. Hoff [ctb],  
Li Wang [ctb],  
Kirk Li [ctb],  
Jake Fisher [ctb],  
Jordan T. Bates [ctb]

**Maintainer** Pavel N. Krivitsky <[pavel@statnet.org](mailto:pavel@statnet.org)>

**Repository** CRAN

**Date/Publication** 2025-09-11 08:10:02 UTC

## Contents

latentnet-package . . . . .	2
as.mcmc.list.ergmm . . . . .	4
bilinear-ergmTerm . . . . .	5
control.ergmm . . . . .	7
davis . . . . .	9
ergmm . . . . .	11
ergmm-class . . . . .	14
ergmm-families . . . . .	15
ergmm.drawpie . . . . .	16
ergmm.par.list . . . . .	17
ergmm.prior . . . . .	18
euclidean-ergmTerm . . . . .	19
euclidean2-ergmTerm . . . . .	20
gof.ergmm . . . . .	21
Intercept-ergmTerm . . . . .	23
latentcov-ergmTerm . . . . .	24
loopcov-ergmTerm . . . . .	25
loopfactor-ergmTerm . . . . .	25
loops-ergmTerm . . . . .	26
mcmc.diagnostics.ergmm . . . . .	27
merge.ergmm . . . . .	28
plot.ergmm . . . . .	29
predict.ergmm . . . . .	33
receivercov-ergmTerm . . . . .	34
rreceiver-ergmTerm . . . . .	35
rsender-ergmTerm . . . . .	36
rsociality-ergmTerm . . . . .	37
sendercov-ergmTerm . . . . .	38
simulate.ergmm . . . . .	39
socialitycov-ergmTerm . . . . .	40
summary.ergmm . . . . .	41
tribes . . . . .	43
<b>Index</b>	<b>45</b>

---

latentnet-package

*latentnet: Latent Position and Cluster Models for Statistical Networks*

---

### Description

Fit and simulate latent position and cluster models for statistical networks. See Krivitsky and Handcock (2008) [doi:10.18637/jss.v024.i05](https://doi.org/10.18637/jss.v024.i05) and Krivitsky, Handcock, Raftery, and Hoff (2009) [doi:10.1016/j.socnet.2009.04.001](https://doi.org/10.1016/j.socnet.2009.04.001).

## Details

The package **latentnet** is used to fit latent cluster random effect models, where the probability of a network  $g$ , on a set of nodes is a product of dyad probabilities, each of which is a GLM with linear component  $\eta_{i,j} = \sum_{k=1}^p \beta_k X_{i,j,k} + d(Z_i, Z_j) + \delta_i + \gamma_j$ , where  $X$  is an array of dyad covariates,  $\beta$  is a vector of covariate coefficients,  $Z_i$  is the latent space position of node  $i$ ,  $d(\cdot, \cdot)$  is a function of the two positions: either negative Euclidean ( $-||Z_i - Z_j||$ ) or bilinear ( $Z_i \cdot Z_j$ ), and  $\delta$  and  $\gamma$  are vectors of sender and receiver effects. (Note that these are different from the eigenmodel of Hoff (2007) “Modeling homophily and stochastic equivalence in symmetric relational data”, fit by package `eigenmodel`.)

`ergmm()` specifies models via:  $g \sim \langle \text{model terms} \rangle$  where  $g$  is a network object. For the list of possible `<model terms>`, see `ergmTerm`. For the list of the possible dyad distribution families, see `families.ergmm`.

The arguments in the `ergmm()` function specific to latent variable models are `ergmm.control()`. See the help page for `ergmm()` for the details.

The result of a latent variable model fit is an `ergmm()` object. Hence the `summary()`, `print()`, and `plot()` methods apply to the fits. The `plot.ergmm()` function has many options specific to latent variable models.

## Author(s)

**Maintainer:** Pavel N. Krivitsky <pavel@statnet.org> ([ORCID](#))

Authors:

- Mark S. Handcock <handcock@stat.ucla.edu>

Other contributors:

- Susan M. Shortreed [contributor]
- Jeremy Tantrum [contributor]
- Peter D. Hoff [contributor]
- Li Wang <lwxwang@gmail.com> [contributor]
- Kirk Li <kirkli@uw.edu> [contributor]
- Jake Fisher <jcf26@duke.edu> [contributor]
- Jordan T. Bates <jtbates@gmail.com> [contributor]

## References

Mark S. Handcock, Adrian E. Raftery and Jeremy Tantrum (2007). *Model-Based Clustering for Social Networks*. Journal of the Royal Statistical Society: Series A (Statistics in Society), 170(2), 301-354.

Peter D. Hoff (2005). *Bilinear Mixed Effects Models for Dyadic Data*. Journal of the American Statistical Association, 100(469), 286-295.

Peter D. Hoff, Adrian E. Raftery and Mark S. Handcock (2002). *Latent space approaches to social network analysis*. Journal of the American Statistical Association, 97(460), 1090-1098.

Pavel N. Krivitsky, Mark S. Handcock, Adrian E. Raftery, and Peter D. Hoff (2009). *Representing degree distributions, clustering, and homophily in social networks with latent cluster random effects models*. *Social Networks*, 31(3), 204-213.

Pavel N. Krivitsky and Mark S. Handcock (2008). *Fitting Position Latent Cluster Models for Social Networks with **latentnet***. *Journal of Statistical Software*, 24(5). doi:10.18637/jss.v024.i05

Susan M. Shortreed, Mark S. Handcock, and Peter D. Hoff (2006). *Positional Estimation within the Latent Space Model for Networks*. *Methodology*, 2(1), 24-33.

## See Also

Useful links:

- <https://statnet.org>
- Report bugs at <https://github.com/statnet/latentnet/issues>

---

as.mcmc.list.ergmm      *Convert an ERGMM Object to an MCMC list object for Diagnostics.*

---

## Description

Functions to extract a subset of MCMC-sampled variables from an object of class `ergmm` and construct an `mcmc.list` object.

## Usage

```
## S3 method for class 'list.ergmm'
as.mcmc(x, burnin = FALSE, which.vars = NULL, vertex.i = c(1), ...)
```

## Arguments

<code>x</code>	An object of class <code>ergmm</code> .
<code>burnin</code>	If TRUE, generates an <code>mcmc.list</code> object for the burnin (if stored) instead of the main sampling run.
<code>which.vars</code>	A named list mapping variable names to the indices to include. If given, overrides the defaults and all arguments that follow.
<code>vertex.i</code>	A numeric vector of vertices whose latent space coordinates and random effects to include.
<code>...</code>	Not used at this time.

## Details

Unless `which.vars` is specified, the `mcmc.list` returned also includes all of the covariate coefficients.

Regardless of whether the MCMC run was single- or multi-threaded, this function returns an `mcmc.list`, with a single thread, if necessary.

**Value**

A `mcmc.list` object with the sample of the selected subset of the variables.

**See Also**

`ergmm`, `mcmc.list`, `mcmc.diagnostics.ergmm`

**Examples**

```
library(coda)
data(sampson)
monks.fit<-ergmm(samplike~euclidean(d=2,G=3))
monks.fit.mcmc<-as.mcmc.list(monks.fit)
plot(monks.fit.mcmc)
raftery.diag(monks.fit.mcmc)
```

---

bilinear-ergmTerm

*Bilinear (inner-product) latent space, with optional clustering*


---

**Description**

Adds a term to the model equal to the inner product of the latent positions:  $Z_i \cdot Z_j$ , where  $Z_i$  and  $Z_j$  are the positions of their respective actors in an unobserved social space. These positions may optionally have a finite spherical Gaussian mixture clustering structure. *Note: For a bilinear latent space effect, two actors being closer in the clustering sense does not necessarily mean that the expected value of a tie between them is higher. Thus, a warning is printed when this model is combined with clustering.*

**Important:** This term works in **latentnet**'s `ergmm()` only. Using it in `ergm()` will result in an error.

**Usage**

```
# binary: bilinear(d, G=0, var.mul=1/8, var=NULL, var.df.mul=1, var.df=NULL,
#             mean.var.mul=1, mean.var=NULL, pK.mul=1, pK=NULL)

# valued: bilinear(d, G=0, var.mul=1/8, var=NULL, var.df.mul=1, var.df=NULL,
#             mean.var.mul=1, mean.var=NULL, pK.mul=1, pK=NULL)
```

**Arguments**

<code>d</code>	The dimension of the latent space.
<code>G</code>	The number of groups (0 for no clustering).
<code>var.mul</code>	In the absence of <code>var</code> , this argument will be used as a scaling factor for a function of average cluster size and latent space dimension to set <code>var</code> . To set it in the prior argument to <code>ergmm</code> , use <code>Z.var.mul</code> .

<code>var</code>	If given, the scale parameter for the scale-inverse-chi-squared prior distribution of the within-cluster variance. To set it in the prior argument to <a href="#">ergmm</a> , use <code>Z.var</code> .
<code>var.df.mul</code>	In the absence of <code>var.df</code> , this argument is the multiplier for the square root of average cluster size, which serves in place of <code>var.df</code> . To set it in the prior argument to <a href="#">ergmm</a> , use <code>Z.var.df.mul</code> .
<code>var.df</code>	The degrees of freedom parameter for the scale-inverse-chi-squared prior distribution of the within-cluster variance. To set it in the prior argument to <a href="#">ergmm</a> , use <code>Z.var.df</code> .
<code>mean.var.mul</code>	In the absence of <code>mean.var</code> , the multiplier for a function of number of vertices and latent space dimension to set <code>mean.var</code> . To set it in the prior argument to <a href="#">ergmm</a> , use <code>Z.mean.var.mul</code> .
<code>mean.var</code>	The variance of the spherical Gaussian prior distribution of the cluster means. To set it in the prior argument to <a href="#">ergmm</a> , use <code>Z.mean.var</code> .
<code>pK.mul</code>	In the absence of <code>pK</code> , this argument is the multiplier for the square root of the average cluster size, which is used as <code>pK</code> . To set it in the prior argument to <a href="#">ergmm</a> , use <code>Z.pK</code> .
<code>pK</code>	The parameter of the Dirichlet prior distribution of cluster assignment probabilities. To set it in the prior argument to <a href="#">ergmm</a> , use <code>Z.pK</code> .

### Details

The following parameters are associated with this term:

`Z` Numeric matrix with rows being latent space positions.

`Z.K` (**when** `G > 0`) Integer vector of cluster assignments.

`Z.mean` (**when** `G > 0`) Numeric matrix with rows being cluster means.

`Z.var` (**when** `G > 0`) Depending on the model, either a numeric vector with within-cluster variances or a numeric scalar with the overall latent space variance.

`Z.pK` (**when** `G > 0`) Numeric vector of probabilities of a vertex being in a particular cluster.

### See Also

[ergmTerm](#) for index of model terms currently visible to the package.

**Keywords:** None

---

`control.ergmm`*Auxiliary for Controlling ERGMM Fitting*

---

**Description**

Auxiliary function as user interface for `ergmm` fitting. Typically only used when calling `ergmm`. It is used to set parameters that affect the sampling but do not affect the posterior distribution.

**Usage**

```
control.ergmm(  
  sample.size = 4000,  
  burnin = 10000,  
  interval = 10,  
  threads = 1,  
  kl.threads = 1,  
  mle.maxit = 100,  
  Z.delta = 0.6,  
  RE.delta = 0.6,  
  group.deltas = 0.4,  
  pilot.runs = 4,  
  pilot.factor = 0.8,  
  pilot.discard.first = 0.5,  
  target.acc.rate = 0.234,  
  backoff.threshold = 0.05,  
  backoff.factor = 0.2,  
  accept.all = FALSE,  
  store.burnin = FALSE,  
  refine.user.start = TRUE  
)  
  
ergmm.control(  
  sample.size = 4000,  
  burnin = 10000,  
  interval = 10,  
  threads = 1,  
  kl.threads = 1,  
  mle.maxit = 100,  
  Z.delta = 0.6,  
  RE.delta = 0.6,  
  group.deltas = 0.4,  
  pilot.runs = 4,  
  pilot.factor = 0.8,  
  pilot.discard.first = 0.5,  
  target.acc.rate = 0.234,  
  backoff.threshold = 0.05,  
  backoff.factor = 0.2,
```

```

accept.all = FALSE,
store.burnin = FALSE,
refine.user.start = TRUE
)

```

## Arguments

<code>sample.size</code>	The number of draws to be taken from the posterior distribution.
<code>burnin</code>	The number of initial MCMC iterations to be discarded.
<code>interval</code>	The number of iterations between consecutive draws.
<code>threads</code>	The number of chains to run. If greater than 1, package <code>snowFT</code> is used to take advantage of any multiprocessing or distributed computing capabilities that may be available. Currently, only PVM (via <code>rpvm</code> ) has been tested. Note, also, that PVM daemon needs to be started before the package is loaded.
<code>kl.threads</code>	If greater than 1, uses an experimental parallelized label-switching algorithm. This is not guaranteed to work.
<code>mle.maxit</code>	Maximum number of iterations for computing the starting values, posterior modes, MLEs, MKL estimates, etc..
<code>Z.delta</code>	Standard deviation of the proposal for the jump in the individual latent space position, or its starting value for the tuner.
<code>RE.delta</code>	Standard deviation of the proposal for the jump in the individual random effects values, or its starting value for the tuner.
<code>group.deltas</code>	A scalar, a vector, or a matrix of an appropriate size, giving the initial proposal structure for the “group proposal” of a jump in covariate coefficients, scaling of latent space positions, and a shift in random effects. If a matrix of an appropriate size is given, it is used as a matrix of coefficients for a correlated proposal. If a vector is given, an independent proposal is used with the corresponding elements being proposal standard deviations. If a scalar is given, it is used as a multiplier for an initial heuristic for the proposal structure. It is usually best to leave this argument alone and let the adaptive sampling be used.
<code>pilot.runs</code>	Number of pilot runs into which to split the burn-in period. After each pilot run, the proposal standard deviations and coefficients <code>Z.delta</code> , <code>RE.delta</code> , and <code>group.deltas</code> are reevaluated. If set to 0, disables adaptive sampling, and only makes a single burn-in run.
<code>pilot.factor</code>	Initial value for the factor by which the coefficients gotten by a Cholesky decomposition of the pilot sample covariance matrix are multiplied.
<code>pilot.discard.first</code>	Proportion of draws from the pilot run to discard for estimating acceptance rate and group proposal covariance.
<code>target.acc.rate</code>	Target acceptance rate for the proposals used. After a pilot run, the proposal variances are adjusted upward if the acceptance rate is above this, and downward if below.
<code>backoff.threshold</code>	If a pilot run’s acceptance rate is below this, redo it with drastically reduced proposal standard deviation. Set to 0 to disable this behavior.

`backoff.factor` Factor by which to multiply the relevant proposal standard deviation if its acceptance rate fell below the backoff threshold.

`accept.all` Forces all proposals to be accepted unconditionally. Use only in debugging proposal distributions!

`store.burnin` If TRUE, the samples from the burnin are also stored and returned, to be used in MCMC diagnostics.

`refine.user.start`  
If TRUE, the values passed to `ergmm` in the `user.start` argument can be updated by the mode-finding algorithm.

**Value**

A list with the arguments as components.

**See Also**

[ergmm](#)

**Examples**

```
data(sampson)
## Shorter run than default.
samp.fit<-ergmm(samplike~euclidean(d=2,G=3)+rreceiver,
control=ergmm.control(burnin=1000,sample.size= 2000,interval=5))
```

---

davis

*Southern Women Data Set (Davis) as a bipartite “network” object*

---

**Description**

This is a data set of 18 women observed over a nine-month period. During that period, various subsets of these women had met in a series of 14 informal social events. The data recored which women met for which events. The data is originally from Davis, Gardner and Gardner (1941) via UCINET and stored as a [network](#) object.

**Details**

This documentation is taken from Freeman (2003) in his usual lucid description. See the reference to the paper below:

In the 1930s, five ethnographers, Allison Davis, Elizabeth Stubbs Davis, Burleigh B. Gardner, Mary R. Gardner and J. G. St. Clair Drake, collected data on stratification in Natchez, Mississippi (Warner, 1988, p. 93). They produced the book cited below (DGG) that reported a comparative study of social class in black and in white society. One element of this work involved examining the correspondence between people’s social class levels and their patterns of informal interaction.

DGG was concerned with the issue of how much the informal contacts made by individuals were established solely (or primarily) with others at approximately their own class levels. To address this question the authors collected data on social events and examined people's patterns of informal contacts.

In particular, they collected systematic data on the social activities of 18 women whom they observed over a nine-month period. During that period, various subsets of these women had met in a series of 14 informal social events. The participation of women in events was uncovered using interviews, the records of participant observers, guest lists, and the newspapers' ' (DGG, p. 149). Homans' day's work behind the counter of a store, a meeting of a women's club, a church supper, a card party, a supper party, a meeting of the Parent-Teacher Association, etc."

This data set has several interesting properties. It is small and manageable. It embodies a relatively simple structural pattern, one in which, according to DGG, the women seemed to organize themselves into two more or less distinct groups. Moreover, they reported that the positions - core and peripheral - of the members of these groups could also be determined in terms of the ways in which different women had been involved in group activities. At the same time, the DGG data set is complicated enough that some of the details of its patterning are less than obvious. As Homans (1950, p. 84) put it, The pattern is frayed at the edges. ' ' And, finally, this data set comes to us in a two-mode woman by event" form. Thus, it provides an opportunity to explore methods designed for direct application to two-mode data. But at the same time, it can easily be transformed into two one-mode matrices (woman by woman or event by event) that can be examined using tools for one-mode analysis.

Because of these properties, this DGG data set has become something of a touchstone for comparing analytic methods in social network analysis. Davis, Gardner and Gardner presented an intuitive interpretation of the data, based in part on their ethnographic experience in the community. Then the DGG data set was picked up by Homans (1950) who provided an alternative intuitive interpretation. In 1972, Phillips and Conviser used an analytic tool, based on information theory, that provided a systematic way to reexamine the DGG data. Since then, this data set has been analyzed again and again. It reappears whenever any network analyst wants to explore the utility of some new tool for analyzing data.

If the source of the data set does not specified otherwise, this data set is protected by the Creative Commons License <https://creativecommons.org/licenses/by-nc-nd/2.5/>.

When publishing results obtained using this data set the original authors should be cited. In addition this package should be cited.

## Source

Linton C. Freeman (2003). *Finding Social Groups: A Meta-Analysis of the Southern Women Data*, In Ronald Breiger, Kathleen Carley and Philippa Pattison, eds. *Dynamic Social Network Modeling and Analysis*. Washington: The National Academies Press.

## References

Davis, A., Gardner, B. B. and M. R. Gardner (1941) *Deep South*, Chicago: The University of Chicago Press.

Linton C. Freeman (2003). *Finding Social Groups: A Meta-Analysis of the Southern Women Data*, In Ronald Breiger, Kathleen Carley and Philippa Pattison, eds. *Dynamic Social Network Modeling and Analysis*. Washington: The National Academies Press.

**See Also**

statnet, network, ergm, ergm

**Examples**

```
data(davis)
# Fit a 2D 2-cluster fit and plot.
davis.fit<-ergmm(davis~euclidean(d=2,G=2)+sociality)
plot(davis.fit, pie=TRUE, rand.eff="sociality")
```

---

 ergmm

*Fit a Latent Space Random Graph Model*


---

**Description**

`ergmm()` is used to fit latent space and latent space cluster random network models, as described by Hoff, Raftery and Handcock (2002), Handcock, Raftery and Tantrum (2005), and Krivitsky, Handcock, Raftery, and Hoff (2009). `ergmm()` can return either a Bayesian model fit or the two-stage MLE.

**Usage**

```
ergmm(
  formula,
  response = NULL,
  family = "Bernoulli",
  fam.par = NULL,
  control = control.ergmm(),
  user.start = list(),
  prior = ergmm.prior(),
  tofit = c("mcmc", "mkl", "mkl.mbc", "procrustes", "klswitch"),
  Z.ref = NULL,
  Z.K.ref = NULL,
  seed = NULL,
  verbose = FALSE
)
```

**Arguments**

formula	An formula object, of the form $g \sim \langle \text{term 1} \rangle + \langle \text{term 2} \rangle \dots$ , where $g$ is a network object or a matrix that can be coerced to a network object, and $\langle \text{term 1} \rangle$ , $\langle \text{term 2} \rangle$ , etc., are each terms for the model. See <a href="#">ergmTerm</a> for the terms that can be fitted, though note the section on fixed effects below. To create a network object in <code>R</code> , use the <code>network</code> function, then add nodal attributes to it using <code>set.vertex.attribute</code> if necessary.
---------	---

Note that, as in `lm()`, the model will include an `intercept` term. This behavior can be overridden by including a `-1` or `+0` term in the formula, and a `1(mean=...,var=...)` term can be used to set a prior different from default.

response	An optional edge attribute that serves as the response variable. By default, presence (1) or absence (0) of an edge in <code>g</code> is used.
family	A character vector specifying the conditional distribution of each edge value. See <a href="#">families.ergmm</a> for the currently implemented families.
fam.par	For those families that require additional parameters, a list.
control	The MCMC parameters that do not affect the posterior distribution such as the sample size, the proposal variances, and tuning parameters, in the form of a named list. See <a href="#">control.ergmm</a> for more information and defaults.
user.start	An optional initial configuration parameters for MCMC in the form of a list. By default, posterior mode conditioned on cluster assignments is used. It is permitted to only supply some of the parameters of a configuration. If this is done, the remaining parameters are fitted conditional on those supplied.
prior	The prior parameters for the model being fitted in the form of a named list. See term help for the terms to use. If given, will override those given in the formula terms, making it useful as a convenient way to store and reproduce a prior distribution. The list or prior parameters can also be extracted from an <a href="#">ERGMM fit object</a> . See <a href="#">ergmm.prior</a> for more information.
tofit	A character vector listing some subset of "pmode", "mcmc", "mkl", "mkl.mbc", "mle", "procrustes", and "kls witch", defaulting to all of the above, instructing <a href="#">ergmm</a> what should be returned as a part of the <a href="#">ERGMM fit object</a> . Omitting can be used to skip particular steps in the fitting process. If the requested procedure or output depends on some other procedure or output not explicitly listed, the dependency will be resolved automatically.
Z.ref	If given, used as a reference for Procrustes analysis.
Z.K.ref	If given, used as a reference for label-switching.
seed	If supplied, random number seed.
verbose	If this is TRUE (or 1), causes information to be printed out about the progress of the fitting, particularly initial value generation. Higher values lead to greater verbosity.

## Value

`ergmm` returns an object of class `ergmm` containing the information about the posterior.

## Specifying fixed effects

Each coefficient for a fixed effect covariate has a normal prior whose mean and variance are set by the `mean` and `var` parameters of the term. For those formula terms that add more than one covariate, a vector can be given for mean and variance. If not, the vectors given will be repeated until the needed length is reached.

`ergmm` can use model terms implemented for the `ergm` package and via the `ergm.userterms` API (in [GitHub repository statnet/ergm.userterms](#)). See [ergmTerm](#) for a list of available terms. If

you wish to specify the prior mean and variance, you can add them to the call. E.g., `TERMNAME(..., mean=0, var=9)`, where ... are the arguments for the `ergm` term, will initialize `TERMNAME` with prior mean of 0 and prior variance of 9.

Some caveats:

- **ergm** has a binary and a valued mode. Regardless of the `family` used, the *binary* variant of the **ergm** term will be used in the linear predictor of the model.
- **ergm** does not support modeling self-loops, so terms imported in this way will always have predictor `x[i, i]==0`. This should not affect most situations, but if you absolutely must model self-loops and non-self-edges in one term, use the deprecated terms below.
- **latentnet** only fits models with dyadic independence. Terms that induce dyadic dependence (e.g., `triangles`) can be used, but then the likelihood of the model will, effectively, be replaced with pseudolikelihood. (Note that under dyadic independence, the two are equal.)

## References

Mark S. Handcock, Adrian E. Raftery and Jeremy Tantrum (2002). *Model-Based Clustering for Social Networks*. Journal of the Royal Statistical Society: Series A, 170(2), 301-354.

Peter D. Hoff, Adrian E. Raftery and Mark S. Handcock (2002). *Latent space approaches to social network analysis*. Journal of the American Statistical Association, 97(460), 1090-1098.

Pavel N. Krivitsky, Mark S. Handcock, Adrian E. Raftery, and Peter D. Hoff (2009). *Representing degree distributions, clustering, and homophily in social networks with latent cluster random effects models*. Social Networks, 31(3), 204-213.

Pavel N. Krivitsky and Mark S. Handcock (2008). *Fitting Position Latent Cluster Models for Social Networks with latentnet*. Journal of Statistical Software, 24(5). doi:10.18637/jss.v024.i05

## See Also

[network](#), [summary.ergmm](#), [ergmTerm](#), [families.ergmm](#)

## Examples

```
#
# Use 'data(package = "latentnet")' to list the data sets in a
#
data(package="latentnet")
#
# Using Sampson's Monk data, lets fit a
# simple latent position model
#
data(sampson)
samp.fit <- ergmm(samplike ~ euclidean(d=2))
#
# See if we have convergence in the MCMC
mcmc.diagnostics(samp.fit)
#
# Plot the fit
```

```

#
plot(samp.fit)
#
# Using Sampson's Monk data, lets fit a latent clustering random effects model
#
samp.fit2 <- ergmm(samplike ~ euclidean(d=2, G=3)+rreceiver)
#
# See if we have convergence in the MCMC
mcmc.diagnostics(samp.fit2)
#
# Plot the fit.
#
plot(samp.fit2, pie=TRUE)

```

---

ergmm-class

*Class of Fitted Exponential Random Graph Mixed Models*


---

## Description

A class `ergmm` to represent a fitted exponential random graph mixed model. The output of `ergmm()`.

## Details

There are methods `summary.ergmm()`, `print.ergmm()`, `plot.ergmm()`, `predict.ergmm()`, and `as.mcmc.list.ergmm()`.

The structure of `ergmm` is as follows:

`sample` An object of class `ergmm.par.list` containing the MCMC sample from the posterior. If the run had multiple threads, their output is concatenated.

`mcmc.mle` A list containing the parameter configuration of the highest-likelihood MCMC iteration.

`mcmc.pmode` A list containing the parameter configuration of the highest-joint-density (conditional on cluster assignments) MCMC iteration.

`mk1` A list containing the MKL estimate.

`model` A list containing the model that was fitted.

`prior` A list containing the information about the prior distribution used. It can be passed as parameter `prior` to `ergmm()` to reproduce the prior in a new fit.

`control` A list containing the information about the model fit settings that do not affect the posterior distribution. It can be passed as parameter `control` to `ergmm()` to reproduce control parameters in a new fit.

`mle` A list containing the MLE, conditioned on cluster assignments.

`pmode` A list containing the posterior mode, conditioned on cluster assignments.

`burnin.start` A list containing the starting value for the burnin.

`main.start` A list (or a list of lists, for a multithreaded run) containing the starting value for the sampling.

**See Also**

[ergmm\(\)](#), [summary.ergmm\(\)](#), [plot.ergmm\(\)](#), [predict.ergmm\(\)](#), [as.mcmc.list.ergmm\(\)](#)

---

 ergmm-families

---

*Edge Weight Distribution Families*


---

**Description**

Family-link combinations supported by [ergmm\(\)](#).

**Details**

Each supported family has a family of functions, of the form `pY.-`, `lpY.-`, `EY.-`, `dIpY.deta.-`, `dIpY.ddispersion.-`, `lpYc.-`, `rsm.-`, followed by the family's name, for the respective family's name, representing the family's likelihood, log-likelihood, expectation, derivative of log-likelihood with respect to the linear predictor, derivative of log-likelihood with respect to the dispersion parameter, log-normalizing-constant, and random sociomatrix generation functions.

On the C side, similar functions exist, but because of static typing, are also provided for "continuous" versions of those families. These should not be used on their own, but are used in estimating MKL positions from the posterior distribution.

**Family-link combinations**

ID	C name	R name	Type	Family	Link
1	Bernoulli_logit	Bernoulli.logit	Discrete	Bernoulli	logit
2	binomial_logit	binomial.logit	Discrete	binomial	logit
3	Poisson_log	Poisson.log	Discrete	Poisson	log
4	Bernoulli_cont_logit	NA	Continuous	Bernoulli	logit
5	binomial_cont_logit	NA	Continuous	binomial	logit
6	Poisson_cont_log	NA	Continuous	Poisson	log
7	normal_identity	normal.identity	Continuous	normal	identity

`.link` can be omitted when not ambiguous. Some families require an appropriate `fam.par` argument to be supplied to [ergmm\(\)](#):

**binomial families** A mandatory `trials` parameter for the number of trials whose success the response counts represent. This can be a single number if the same for all dyads, or a matrix if not. For non-bipartite networks, the matrix must be a square matrix with dimension equal to the number of nodes; for a bipartite network, it can also be a rectangular incidence matrix. If the network is undirected, the lower triangle is used.

**normal** Mandatory `prior.var` and `prior.var.df` parameters for the prior scale and degrees of freedom of the variance of the dyad values.

---

ergmm.drawpie	<i>Draw a pie chart at a specified location.</i>
---------------	--

---

### Description

Used by `plot.ergmm` to draw pie charts to visualize soft clusterings when `pie=TRUE`. Exported as a courtesy to dependent packages.

### Usage

```
ergmm.drawpie(center, radius, probs, n = 50, cols = seq_along(probs), ...)
```

### Arguments

<code>center</code>	A numeric vector of length 2, specifying the horizontal and the vertical coordinates of its center.
<code>radius</code>	Radius of the pie chart.
<code>probs</code>	A vector of probabilities/weights of each sector; they do not have to sum to 1.
<code>n</code>	Number of points to use to approximate the "circle".
<code>cols</code>	A vector of colors to use for the sectors.
<code>...</code>	Additional arguments, currently unused.

### Author(s)

See COPYRIGHT.

### See Also

`plot.ergmm`

### Examples

```
plot(c(0,sum(1:11))*2,c(-10,10),type="n",asp=1)
for(i in 1:10) ergmm.drawpie(c(sum(1:i)*2,0), radius=i, probs=1:(i+1))
```

---

ergmm.par.list                    *A List of ERGMM Parameter Configuration*

---

## Description

A class `ergmm.par.list` to represent a series of parameter configurations for the same exponential random graph mixed model.

## Usage

```
as.ergmm.par.list(x, ...)
```

```
## S3 method for class 'ergmm.par.list'
```

```
x[i]
```

```
## S3 method for class 'ergmm.par.list'
```

```
length(x)
```

```
## S3 method for class 'ergmm.par.list'
```

```
x[[i]]
```

```
## S3 method for class 'ergmm.par.list'
```

```
unstack(x, ...)
```

## Arguments

<code>x</code>	an <code>ergm.par.list</code> object.
<code>...</code>	extra arguments, currently unused.
<code>i</code>	index of the iteration to extract.

## Details

`[[` operator with a numeric or integer index returns a list with the the configuration with that index. `[` operator given a numeric vector returns a `ergmm.par.list` object with the subset of configurations with the indices given.

The structure of `ergmm.par.list` is derived from named lists, with each entry having an additional dimension (always the first one), indexed by configuration. That is, scalars become vectors, vectors become matrixes with the original vectors in rows, and matrices become 3-dimensional arrays, with the original matrices indexed by their first dimension. See term documentation for comon elements of these configurations. In some cases, such as when representing MCMC or optimization output, the object may also have some of the following elements:

`m1p`  $\log p(Y, Z, \beta, \mu, \sigma, \delta, \gamma, \sigma_\delta, \sigma_\gamma, |K)$ , the joint probability/density of network, the covariate coefficients, the latent space positions and parameters, and the random effects and their variances, conditional on cluster assignments.

`1pY`  $\log p(Y | \dots)$ , depending on the model, the log-probability or log-density of the network conditional on all the parameters.

`lpZ`  $\log p(Z|\mu, \sigma, K)$ , the log-density of latent space positions conditional on latent space or cluster parameters and cluster assignments.

`lpbeta`  $\log p(\beta)$ , the prior log-density of the covariate coefficients.

`lpRE`  $\log p(\delta, \gamma|\sigma_\delta, \sigma_\gamma)$ , the log-density of all random effects, conditional on their respective variances.

`lpLV`  $\log p(\mu, \sigma)$ , the prior log-density of latent space or cluster parameters (but not that of the cluster assignments).

`lpREV`  $\log p(\sigma_\delta, \sigma_\gamma)$ , the prior log-density of all random effect variances.

`Z.rate` Proportion of single-vertex proposals accepted over the preceding interval.

`beta.rate` Proportion of group proposals accepted over the preceding interval.

### See Also

[ergmm](#)

---

ergmm.prior

*Auxiliary for Setting the ERGMM Prior*

---

### Description

Auxiliary function as user interface for [ergmm](#) prior specification. Typically only used when calling [ergmm](#). It is used to supply the parameters of the prior distribution of the model, to overwrite those specified in the model formula, and to supply miscellaneous prior parameters.

### Usage

```
ergmm.prior(..., adjust.beta.var = TRUE)
```

### Arguments

`...` Prior distribution parameters. See term documentation and [ergmm\(\)](#) for more information.

`adjust.beta.var`

A shortcut: whether the prior variance for each covariate coefficient should be divided by the mean square of that covariate. This adjustment affects those variances specified in the formula or by default, but not those specified through the `prior=` argument.

### Value

A list with the arguments as elements.

### See Also

[ergmm\(\)](#), [ergmTerm](#)

---

euclidean-ergmTerm      *Euclidean distance latent space, with optional clustering*

---

### Description

Adds a term to the model equal to the negative Euclidean distance  $-||Z_i - Z_j||$ , where  $Z_i$  and  $Z_j$  are the positions of their respective actors in an unobserved social space. These positions may optionally have a finite spherical Gaussian mixture clustering structure. This term was previously called latent.

**Important:** This term works in **latentnet**'s `ergmm()` only. Using it in `ergm()` will result in an error.

### Usage

```
# binary: euclidean(d, G=0, var.mul=1/8, var=NULL, var.df.mul=1, var.df=NULL,
#               mean.var.mul=1, mean.var=NULL, pK.mul=1, pK=NULL)

# valued: euclidean(d, G=0, var.mul=1/8, var=NULL, var.df.mul=1, var.df=NULL,
#               mean.var.mul=1, mean.var=NULL, pK.mul=1, pK=NULL)
```

### Arguments

d	The dimension of the latent space.
G	The number of groups (0 for no clustering).
var.mul	In the absence of var, this argument will be used as a scaling factor for a function of average cluster size and latent space dimension to set var. To set it in the prior argument to <code>ergmm</code> , use <code>Z.var.mul</code> .
var	If given, the scale parameter for the scale-inverse-chi-squared prior distribution of the within-cluster variance. To set it in the prior argument to <code>ergmm</code> , use <code>Z.var</code> .
var.df.mul	In the absence of var.df, this argument is the multiplier for the square root of average cluster size, which serves in place of var.df. To set it in the prior argument to <code>ergmm</code> , use <code>Z.var.df.mul</code> .
var.df	The degrees of freedom parameter for the scale-inverse-chi-squared prior distribution of the within-cluster variance. To set it in the prior argument to <code>ergmm</code> , use <code>Z.var.df</code> .
mean.var.mul	In the absence of mean.var, the multiplier for a function of number of vertices and latent space dimension to set mean.var. To set it in the prior argument to <code>ergmm</code> , use <code>Z.mean.var.mul</code> .
mean.var	The variance of the spherical Gaussian prior distribution of the cluster means. To set it in the prior argument to <code>ergmm</code> , use <code>Z.mean.var</code> .
pK.mul	In the absence of pK, this argument is the multiplier for the square root of the average cluster size, which is used as pK. To set it in the prior argument to <code>ergmm</code> , use <code>Z.pK</code> .
pK	The parameter of the Dirichlet prior distribution of cluster assignment probabilities. To set it in the prior argument to <code>ergmm</code> , use <code>Z.pK</code> .

**Details**

The following parameters are associated with this term:

Z Numeric matrix with rows being latent space positions.

Z.K (**when**  $G > 0$ ) Integer vector of cluster assignments.

Z.mean (**when**  $G > 0$ ) Numeric matrix with rows being cluster means.

Z.var (**when**  $G > 0$ ) Depending on the model, either a numeric vector with within-cluster variances or a numeric scalar with the overall latent space variance.

Z.pK (**when**  $G > 0$ ) Numeric vector of probabilities of a vertex being in a particular cluster.

**See Also**

[ergmTerm](#) for index of model terms currently visible to the package.

**Keywords:** None

---

euclidean2-ergmTerm     *Squared euclidean distance latent space, with optional clustering*

---

**Description**

Adds a term to the model equal to the negative Euclidean distance  $-||Z_i - Z_j||^2$ , where  $Z_i$  and  $Z_j$  are the positions of their respective actors in an unobserved social space. These positions may optionally have a finite spherical Gaussian mixture clustering structure. This term was previously called latent.

**Important:** This term works in **latentnet**'s `ergmm()` only. Using it in `ergm()` will result in an error.

**Usage**

```
# binary: euclidean(d, G=0, var.mul=1/8, var=NULL, var.df.mul=1, var.df=NULL,
#               mean.var.mul=1, mean.var=NULL, pK.mul=1, pK=NULL)
```

```
# valued: euclidean(d, G=0, var.mul=1/8, var=NULL, var.df.mul=1, var.df=NULL,
#               mean.var.mul=1, mean.var=NULL, pK.mul=1, pK=NULL)
```

**Arguments**

d	The dimension of the latent space.
G	The number of groups (0 for no clustering).
var.mul	In the absence of var, this argument will be used as a scaling factor for a function of average cluster size and latent space dimension to set var. To set it in the prior argument to <code>ergmm</code> , use <code>Z.var.mul</code> .
var	If given, the scale parameter for the scale-inverse-chi-squared prior distribution of the within-cluster variance. To set it in the prior argument to <code>ergmm</code> , use <code>Z.var</code> .

var.df.mul	In the absence of var.df, this argument is the multiplier for the square root of average cluster size, which serves in place of var.df. To set it in the prior argument to <a href="#">ergmm</a> , use Z.var.df.mul.
var.df	The degrees of freedom parameter for the scale-inverse-chi-squared prior distribution of the within-cluster variance. To set it in the prior argument to <a href="#">ergmm</a> , use Z.var.df.
mean.var.mul	In the absence of mean.var, the multiplier for a function of number of vertices and latent space dimension to set mean.var. To set it in the prior argument to <a href="#">ergmm</a> , use Z.mean.var.mul.
mean.var	The variance of the spherical Gaussian prior distribution of the cluster means. To set it in the prior argument to <a href="#">ergmm</a> , use Z.mean.var.
pK.mul	In the absence of pK, this argument is the multiplier for the square root of the average cluster size, which is used as pK. To set it in the prior argument to <a href="#">ergmm</a> , use Z.pK.
pK	The parameter of the Dirichlet prior distribution of cluster assignment probabilities. To set it in the prior argument to <a href="#">ergmm</a> , use Z.pK.

### Details

The following parameters are associated with this term:

Z Numeric matrix with rows being latent space positions.

Z.K (**when**  $G > 0$ ) Integer vector of cluster assignments.

Z.mean (**when**  $G > 0$ ) Numeric matrix with rows being cluster means.

Z.var (**when**  $G > 0$ ) Depending on the model, either a numeric vector with within-cluster variances or a numeric scalar with the overall latent space variance.

Z.pK (**when**  $G > 0$ ) Numeric vector of probabilities of a vertex being in a particular cluster.

### See Also

[ergmTerm](#) for index of model terms currently visible to the package.

**Keywords:** None

---

gof.ergmm

*Conduct Goodness-of-Fit Diagnostics on a Exponential Family Random Graph Mixed Model Fit*

---

### Description

[gof](#) calculates  $p$ -values for geodesic distance, degree, and reachability summaries to diagnose the goodness-of-fit of exponential family random graph mixed models. See [ergmm](#) for more information on these models.

**Usage**

```
## S3 method for class 'ergmm'
gof(
  object,
  ...,
  nsim = 100,
  GOF = ~idegree + odegree + distance,
  verbose = FALSE
)
```

**Arguments**

object	an <a href="#">ergmm</a> object (returned by <a href="#">ergmm</a> ).
...	Additional arguments, to be passed to lower-level functions in the future.
nsim	The number of simulations to use for the MCMC $p$ -values. This is the size of the sample of graphs to be randomly drawn from the distribution specified by the object on the set of all graphs.
GOF	formula; an formula object, of the form $\sim$ <model terms> specifying the statistics to use to diagnosis the goodness-of-fit of the model. They do not need to be in the model formula specified in formula, and typically are not. Examples are the degree distribution ("degree"), minimum geodesic distances ("dist"), and shared partner distributions ("espartners" and "dpartners"). For the details on the possible <model terms>, see <a href="#">ergm-terms</a> .
verbose	Provide verbose information on the progress of the simulation.

**Details**

A sample of graphs is randomly drawn from the posterior of the [ergmm](#).

A plot of the summary measures is plotted. More information can be found by looking at the documentation of [ergm](#).

**Value**

[gof](#) and [gof.ergmm](#) return an object of class `gof`. This is a list of the tables of statistics and  $p$ -values. This is typically plotted using [plot.gof](#).

**See Also**

[ergmm](#), [ergmm\(object\)](#), [ergm](#), [network](#), [simulate.ergmm](#), [plot.gof](#)

**Examples**

```
#
data(sampson)
#
# test the gof.ergm function
#
```

```

samplike.fit <- ergmm(samplike ~ euclidean(d=2,G=3),
                    control=ergmm.control(burnin=1000,interval=5))
samplike.fit
summary(samplike.fit)

#
# Plot the probabilities first
#
monks.gof <- gof(samplike.fit)
monks.gof
#
# Place all three on the same page
# with nice margins
#
par(mfrow=c(1,3))
par(oma=c(0.5,2,1,0.5))
#
plot(monks.gof)
#
# And now the odds
#
plot(monks.gof, plotlogodds=TRUE)

```

---

Intercept-ergmTerm      *Intercept*

---

## Description

This term serves as an intercept term, is included by default (though, as in `lm`, it can be excluded by adding `+0` or `-1` into the model formula). It adds one covariate to the model, for which  $x[i, j]=1$  for all  $i$  and  $j$ .

It can be used explicitly to set prior mean and variance for the intercept term.

**Important:** This term works in `latentnet`'s `ergmm()` only. Using it in `ergm()` will result in an error.

## Usage

```

# binary: 1(mean=0, var=9)

# binary: Intercept(mean=0, var=9)

# binary: intercept(mean=0, var=9)

# valued: 1(mean=0, var=9)

# valued: Intercept(mean=0, var=9)

# valued: intercept(mean=0, var=9)

```

**Arguments**

mean, var            prior mean and variance.

**See Also**

[ergmTerm](#) for index of model terms currently visible to the package.

**Keywords:** None

latentcov-ergmTerm      *Edge covariates for the latent model*

**Description**

This term adds one statistic to the model, providing a covariate for each dyad.

latentcov can be called more than once, to model the effects of multiple covariates. Note that some covariates can be more conveniently specified using other terms.

**Important:** This term works in **latentnet**'s `ergmm()` only. Using it in `ergm()` will result in an error.

It is recommended to only use this term if your network contains self-loops. Otherwise, use the **ergm** term(s) `edgecov()`.

**Usage**

```
# binary: latentcov(x, attrname=NULL, mean=0, var=9)
```

```
# valued: latentcov(x, attrname=NULL, mean=0, var=9)
```

**Arguments**

x, attrname            a specification for the dyadic covariate: either one of the following, or the name of a network attribute containing one of the following:

**a covariate matrix** with dimensions  $n \times n$  for unipartite networks and  $b \times (n - b)$  for bipartite networks; attrname, if given, is used to construct the term name.

**a network object** with the same size and bipartitedness as LHS; attrname, if given, provides the name of the quantitative edge attribute to use for covariate values (in this case, missing edges in x are assigned a covariate value of zero).

mean, var            prior mean and variance.

**See Also**

[ergmTerm](#) for index of model terms currently visible to the package.

**Keywords:** None

---

loopcov-ergmTerm	<i>Covariate effect on self-loops</i>
------------------	---------------------------------------

---

### Description

This term adds one covariate to the model, for which  $x[i, i] = \text{attrname}(i)$  and  $x[i, j] = 0$  for  $i \neq j$ . This term only makes sense if the network has self-loops.

**Important:** This term works in **latentnet**'s `ergmm()` only. Using it in `ergm()` will result in an error.

### Usage

```
# binary: loopcov(attrname, mean=0, var=9)
```

```
# valued: loopcov(attrname, mean=0, var=9)
```

### Arguments

attrname	a character string giving the name of a numeric (not categorical) attribute in the network's vertex attribute list.
mean, var	prior mean and variance.

### See Also

[ergmTerm](#) for index of model terms currently visible to the package.

**Keywords:** None

---

loopfactor-ergmTerm	<i>Factor attribute effect on self-loops</i>
---------------------	--

---

### Description

This term adds multiple covariates to the model, one for each of (a subset of) the unique values of the `attrname` attribute (or each combination of the attributes given). Each of these covariates has  $x[i, i] = 1$  if  $\text{attrname}(i) = l$ , where  $l$  is that covariate's level, and  $x[i, j] = 0$  otherwise. To include all attribute values set `base=0` – because the sum of all such statistics equals twice the number of self-loops and hence a linear dependency would arise in any model also including loops. Thus, the `base` argument tells which value(s) (numbered in order according to the `sort` function) should be omitted. The default value, `base=1`, means that the smallest (i.e., first in sorted order) attribute value is omitted. For example, if the “fruit” factor has levels “orange”, “apple”, “banana”, and “pear”, then to add just two terms, one for “apple” and one for “pear”, then set “banana” and “orange” to the base (remember to sort the values first) by using `nodefactor("fruit", base=2:3)`. For an analogous term for quantitative vertex attributes, see `nodecov.attrname` is a character string giving the name of a numeric (not categorical) attribute in the network's vertex attribute list. This term adds one covariate to the model, for which  $x[i, i] = \text{attrname}(i)$  and  $x[i, j] = 0$  for  $i \neq j$ . This term only makes sense if the network has self-loops.

**Important:** This term works in **latentnet**'s `ergmm()` only. Using it in `ergm()` will result in an error.

**Usage**

```
# binary: loopfactor(attrname, mean=0, var=9)
# valued: loopfactor(attrname, mean=0, var=9)
```

**Arguments**

attrname            a character vector giving one or more names of categorical attributes in the network's vertex attribute list.

mean, var            prior mean and variance.

**See Also**

[ergmTerm](#) for index of model terms currently visible to the package.

**Keywords:** None

---

loops-ergmTerm	<i>Self-loops</i>
----------------	-------------------

---

**Description**

Effect of the dyad being a self-loop (i.e.,  $(i, i)$ ).

**Important:** This term works in **latentnet**'s [ergmm\(\)](#) only. Using it in [ergm\(\)](#) will result in an error.

**Usage**

```
# binary: loops(mean=0, var=9)
# valued: loops(mean=0, var=9)
```

**Arguments**

mean, var            prior mean and variance.

**See Also**

[ergmTerm](#) for index of model terms currently visible to the package.

**Keywords:** None

---

mcmc.diagnostics.ergmm

*Conduct MCMC diagnostics on an ERGMM fit*


---

## Description

This function creates simple diagnostic plots for the MCMC sampled statistics produced from a fit. It also prints the Raftery-Lewis diagnostics, indicates if they are sufficient, and suggests the run length required.

## Usage

```
## S3 method for class 'ergmm'
mcmc.diagnostics(
  object,
  which.diags = c("cor", "acf", "trace", "raftery"),
  burnin = FALSE,
  which.vars = NULL,
  vertex.i = c(1),
  ...
)
```

## Arguments

object	An object of class <code>ergmm</code> .
which.diags	A list of diagnostics to produce. "cor" is the correlation matrix of the statistics, "acf" plots the autocorrelation functions, "trace" produces trace plots and density estimates, and "raftery" produces the Raftery-Lewis statistics.
burnin	If not FALSE, rather than perform diagnostics on the sampling run, performs them on the pilot run whose index is given.
which.vars	A named list mapping variable names to the indices to include. If given, overrides the defaults and all arguments that follow.
vertex.i	A numeric vector of vertices whose latent space coordinates and random effects to include.
...	Additional arguments. None are supported at the moment.

## Details

Produces the plots per `which.diags`. Autocorrelation function that is printed if "acf" is requested is for lags 0 and interval.

## Value

`mcmc.diagnostics.ergmm` returns a table of Raftery-Lewis diagnostics.

**See Also**

[ergmm](#), [ergmm.object](#), [raftery.diag](#), [autocorr](#), [plot.mcmc.list](#)

**Examples**

```
#
data(sampson)
#
# test the mcmc.diagnostics function
#
gest <- ergmm(samplike ~ euclidean(d=2),
              control=ergmm.control(burnin=1000,interval=5))
summary(gest)
#
# Plot the traces and densities
#
mcmc.diagnostics(gest)
```

---

merge.ergmm

---

*Merge two or more replications of ERGMM fits*


---

**Description**

A [merge](#) method for [ergmm](#) objects, constructing an [ergmm](#) object containing the combined MCMC output (and derived estimates) of several [ergmm](#) objects produced with the same input parameters but different starting values, random seeds, etc..

**Usage**

```
## S3 method for class 'ergmm'
merge(x, y, ..., verbose = FALSE)
```

**Arguments**

x	The first <a href="#">ergmm</a> object to be merged.
y	The second <a href="#">ergmm</a> object to be merged.
...	Additional <a href="#">ergmm</a> objects to be merged.
verbose	If TRUE, marks the progress of merging.

**Value**

An object of class [ergmm](#).

**See Also**[ergmm.object](#), [ergmm](#)**Examples**

```

data(sampson)
# Run two short MCMC-based fits.
samp.fit1 <- ergmm(samplike ~ euclidean(d=2, G=3),
  control=ergmm.control(burnin=1000,interval=10,sample.size=2000))
samp.fit2 <- ergmm(samplike ~ euclidean(d=2, G=3),
  control=ergmm.control(burnin=1000,interval=10,sample.size=2000))

# Combine them, and summarize the result.
samp.fit <- merge(samp.fit1,samp.fit2)
summary(samp.fit)

```

plot.ergmm

*Plotting Method for class ERGMM***Description**

`plot.ergmm` is the plotting method for `ergmm` objects. For latent models, this plots the minimum Kullback-Leibler positions by default. The maximum likelihood, posterior mean, posterior mode, or a particular iteration's or configuration's positions can be used instead, or pie charts of the posterior probabilities of cluster membership can be shown. See `ergmm` for more information on how to fit these models.

**Usage**

```

## S3 method for class 'ergmm'
plot(
  x,
  ...,
  vertex.cex = 1,
  vertex.sides = 16 * ceiling(sqrt(vertex.cex)),
  what = "mkl",
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  zlab = NULL,
  xlim = NULL,
  ylim = NULL,
  zlim = NULL,
  object.scale = formals(plot.network.default)[["object.scale"]],
  pad = formals(plot.network.default)[["pad"]],

```

```

cluster.col = c("red", "green", "blue", "cyan", "magenta", "orange", "yellow",
  "purple"),
vertex.col = NULL,
print.formula = TRUE,
edge.col = 8,
Z.ref = NULL,
Z.K.ref = NULL,
zoom.on = NULL,
pie = FALSE,
labels = FALSE,
rand.eff = NULL,
rand.eff.cap = NULL,
plot.means = TRUE,
plot.vars = TRUE,
suppress.axes = FALSE,
jitter1D = 1,
curve1D = TRUE,
use.rgl = FALSE,
vertex.3d.cex = 1/20,
edge.plot3d = TRUE,
suppress.center = FALSE,
density.par = list()
)

```

### Arguments

x	an R object of class <code>ergmm</code> . See documentation for <code>ergmm</code> .
...	Other optional arguments passed to the <code>plot.network</code> function.
what	Character vector, integer, or a list that specifies the point estimates to be used. Can be one of the following: <ul style="list-style-type: none"> <li>"mkl" This is the default. Plots the Minimum Kulblack-Leibler divergence values.</li> <li>"start","burnin.start" Plots the starting configuration.</li> <li>"sampling.start" Plots the starting configuration of the sampling phase (the last burnin configuration).</li> <li>"mle" Plots the maximum likelihood estimates. Random effects are treated as fixed.</li> <li>"pmean" Plots the posterior means.</li> <li>"pmode" Plots the conditional posterior mode.</li> <li>"cloud" Plots the "cloud" of latent space position draws, with their cluster colors.</li> <li>"density" Plots density and contours of the posterior latent positions, and, in cluster models, each cluster.</li> <li>list Plots the configuration contained in the list.</li> <li><b>integer</b> Plots the configuration of whatth MCMC draw stored in x.</li> </ul>
main, vertex.cex, vertex.col, xlim, ylim, vertex.sides	Arguments passed to <code>plot.network</code> , whose defaults differ from those of <code>plot.network</code> .

zlim, zlab	Limits and labels for the third latent space dimension or principal component, if use.rgl=TRUE.
object.scale, pad, edge.col, xlab, ylab	Arguments passed to <a href="#">plot.network</a> , whose defaults differ from those of <a href="#">plot.network</a> .
cluster.col	A vector of colors used to distinguish clusters in a latent cluster model.
print.formula	Whether the formula based on which the x was fitted should be printed under the main title. Defaults to TRUE.
Z.ref	If given, rotates the the latent positions to the nearest configuration to this one before plotting.
Z.K.ref	If given, relabels the clusters to the nearest configuration to this one before plotting.
zoom.on	If given a list of vertex indices, sets the plotting region to the smallest that can fit those vertices.
pie	For latent clustering models, each node is drawn as a pie chart representing the probabilities of cluster membership.
labels	Whether vertex labels should be displayed. Defaults to FALSE.
rand.eff	A character vector selecting "sender", "receiver", "sociality", or "total" random effects. Each vertex is scaled such that its area is proportional to the odds ratio due to its selected random effect.
rand.eff.cap	If not NULL and rand.eff is given, limits the scaling of the plotting symbol due to random effect to the given value.
plot.means	Whether cluster means are plotted for latent cluster models. The "+" character is used. Defaults to TRUE.
plot.vars	Whether circles with radius equal to the square root of posterior latent or intra-cluster variance estimates are plotted. Defaults to TRUE.
suppress.axes	Whether axes should <i>not</i> be drawn. Defaults to FALSE. (Axes are drawn.)
jitter1D	For 1D latent space fits, it often helps to jitter the positions for visualization. This option controls the amount of jitter.
curve1D	Controls whether the edges in 1D latent space fits are plotted as curves. Defaults to TRUE.
use.rgl	Whether the package rgl should be used to plot fits for latent space dimension 3 or higher in 3D. Defaults to FALSE. If set to TRUE, argument pie has no effect.
vertex.3d.cex	Controls the size of the plotting symbol when use.rgl=TRUE.
edge.plot3d	If TRUE (the default) edges or arcs in a 3D plot will be drawn. Otherwise, only vertices and clusters.
suppress.center	Suppresses the plotting of "+" at the origin. Defaults to FALSE.
density.par	A list of optional parameters for density plots: <ul style="list-style-type: none"> <li>totaldens Whether the overall density of latent space positions should be plotted. Defaults to TRUE.</li> <li>subdens Whether the densities of latent space positions broken down by cluster should be plotted. Defaults to TRUE.</li> <li>mfrom When plotting multiple clusters' densities, passed to <a href="#">par</a></li> </ul>

**Details**

At this time, no plotting non-latent-space model fits is not supported.

Plots the results of an ergmm fit.

More information can be found by looking at the documentation of [ergmm](#).

For bipartite networks, the events are marked with a bullet (small black circle) inside the plotting symbol.

**Value**

If applicable, invisibly returns the vertex positions plotted.

**See Also**

[ergmm](#), [ergmm.object](#), [network](#), [plot.network](#), [plot](#)

**Examples**

```
#
# Using Sampson's Monk data, let's fit a
# simple latent position model
#
data(sampson)
#
# Using Sampson's Monk data, let's fit a
# latent clustering random effects model
# Store the burn-in.
samp.fit <- ergmm(samplike ~ euclidean(d=2, G=3)+rreceiver,
                 control=ergmm.control(store.burnin=TRUE))
#
# See if we have convergence in the MCMC
mcmc.diagnostics(samp.fit)
# We can also plot the burn-in:
for(i in samp.fit$control$pilot.runs) mcmc.diagnostics(samp.fit,burnin=i)
#
# Plot the resulting fit.
#
plot(samp.fit,labels=TRUE,rand.eff="receiver")
plot(samp.fit,pie=TRUE,rand.eff="receiver")
plot(samp.fit,what="pmean",rand.eff="receiver")
plot(samp.fit,what="cloud",rand.eff="receiver")
plot(samp.fit,what="density",rand.eff="receiver")
plot(samp.fit,what=5,rand.eff="receiver")

## Not run:
# Fit a 3D latent space model to Sampson's Monks
samp.fit3 <- ergmm(samplike ~ euclidean(d=3))

# Plot the first two principal components of the
# latent space positions
```

```

plot(samp.fit,use.rgl=FALSE)
# Plot the resulting fit in 3D
plot(samp.fit,use.rgl=TRUE)

## End(Not run)

```

---

predict.ergmm	<i>Predicted Dyad Values for an ERGMM.</i>
---------------	--

---

### Description

Returns a matrix of expected dyad values based on an ERGMM fit.

### Usage

```

## S3 method for class 'ergmm'
predict(object, ..., type = "post")

```

### Arguments

object	An object of class <code>ergmm</code> .
...	Additional arguments. Currently unused.
type	One of "mkl", "start", "mle", "pmean", "mkl", "pmode", "post", an index of the iteration to use, or a list, for the configuration of parameters based on which the prediction is made. An exception is "post", which computes the expected dyad values integrated over the posterior.

### Value

A sociomatrix of predicted values. Note that predictions are made for unobserved values (whether structural zeros or unobserved dyads).

### See Also

[ergmm](#)

### Examples

```

data(sampson)
monks.fit<-ergmm(samplike~euclidean(d=2,G=3),tofit="mcmc")
heatmap(predict(monks.fit),Rowv=NA,Colv=NA)

```

---

receivercov-ergmTerm *Receiver covariate effect*

---

### Description

If the attribute is numeric, this term adds one covariate to the model equaling `attrname(i)`. If the attribute is not numeric or `force.factor==TRUE`, this term adds  $p-1$  covariates to the model, where  $p$  is the number of unique values of `attrname`. The  $k$ th such covariate has the value `attrname(i) == value(k+1)`, where `value(k)` is the  $k$ th smallest unique value of the `attrname` attribute. This term only makes sense if the network is directed.

**Important:** This term works in **latentnet**'s `ergmm()` only. Using it in `ergm()` will result in an error.

It is recommended to only use this term if your network contains self-loops. Otherwise, use the **ergm** term(s) `nodeicov()` or `nodeifactor()`.

### Usage

```
# binary: receivercov(attrname, force.factor=FALSE, mean=0, var=9)
```

```
# valued: receivercov(attrname, force.factor=FALSE, mean=0, var=9)
```

### Arguments

<code>attrname</code>	a character string giving the name of an attribute in the network's vertex attribute list.
<code>force.factor</code>	logical, indicating if <code>attrname</code> 's value should be interpreted as categorical even if numeric.
<code>mean, var</code>	prior mean and variance.

### Note

This term can only be used with directed networks.

### See Also

[ergmTerm](#) for index of model terms currently visible to the package.

**Keywords:** None

---

rreceiver-ergmTerm      *Random receiver effect*

---

### Description

Adds a random receiver effect to the model, with normal prior centered around 0 and a variance that is estimated. Can only be used on directed networks.

**Important:** This term works in **latentnet**'s `ergmm()` only. Using it in `ergm()` will result in an error.

### Usage

```
# binary: rreceiver(var=1, var.df=3)
```

```
# valued: rreceiver(var=1, var.df=3)
```

### Arguments

<code>var</code>	The scale parameter for the scale-inverse-chi-squared prior distribution of the receiver effect variance. To set it in the prior argument to <code>ergmm</code> , use <code>receiver.var</code> .
<code>var.df</code>	The degrees of freedom parameter for the scale-inverse-chi-squared prior distribution of the receiver effect variance. To set it in the prior argument to <code>ergmm</code> , use <code>receiver.var.df</code> .

### Details

The following parameters are associated with this term:

`receiver` Numeric vector of values of each vertex's random receiver effect.

`receiver.var` Random receiver effect's variance.

### Note

This term can only be used with directed networks.

### See Also

[ergmTerm](#) for index of model terms currently visible to the package.

**Keywords:** None

---

rsender-ergmTerm      *Random sender effect*

---

### Description

Adds a random sender effect to the model, with normal prior centered around 0 and a variance that is estimated. Can only be used on directed networks.

**Important:** This term works in **latentnet**'s `ergmm()` only. Using it in `ergm()` will result in an error.

### Usage

```
# binary: rsender(var=1, var.df=3)
```

```
# valued: rsender(var=1, var.df=3)
```

### Arguments

<code>var</code>	The scale parameter for the scale-inverse-chi-squared prior distribution of the sender effect variance. To set it in the <code>prior</code> argument to <code>ergmm</code> , use <code>sender.var</code> .
<code>var.df</code>	The degrees of freedom parameter for the scale-inverse-chi-squared prior distribution of the sender effect variance. To set it in the <code>prior</code> argument to <code>ergmm</code> , use <code>sender.var.df</code> .

### Details

The following parameters are associated with this term:

`sender` Numeric vector of values of each vertex's random sender effect.

`sender.var` Random sender effect's variance.

### Note

This term can only be used with directed networks.

### See Also

[ergmTerm](#) for index of model terms currently visible to the package.

**Keywords:** None

---

rsociality-ergmTerm    *Random sociality effect*

---

### Description

Adds a random sociality effect to the model, with normal prior centered around 0 and a variance that is estimated.

**Important:** This term works in **latentnet**'s `ergmm()` only. Using it in `ergm()` will result in an error.

### Usage

```
# binary: rsociality(var=1, var.df=3)
# valued: rsociality(var=1, var.df=3)
```

### Arguments

<code>var</code>	The scale parameter for the scale-inverse-chi-squared prior distribution of the sociality effect variance. To set it in the prior argument to <code>ergmm</code> , use <code>sociality.var</code> .
<code>var.df</code>	The degrees of freedom parameter for the scale-inverse-chi-squared prior distribution of the sociality effect variance. To set it in the prior argument to <code>ergmm</code> , use <code>sociality.var.df</code> .

### Details

The following parameters are associated with this term:

`sociality` Numeric vector of values of each vertex's random sociality effect.

`sociality.var` Random sociality effect's variance.

### See Also

[ergmTerm](#) for index of model terms currently visible to the package.

**Keywords:** None

---

sendercov-ergmTerm      *Sender covariate effect*

---

### Description

If the attribute is numeric, this term adds one covariate to the model equaling `attrname(i)`. If the attribute is not numeric or `force.factor==TRUE`, this term adds  $p-1$  covariates to the model, where  $p$  is the number of unique values of `attrname`. The  $k$ th such covariate has the value `attrname(i) == value(k+1)`, where `value(k)` is the  $k$ th smallest unique value of the `attrname` attribute. This term only makes sense if the network is directed.

**Important:** This term works in **latentnet**'s `ergmm()` only. Using it in `ergm()` will result in an error.

It is recommended to only use this term if your network contains self-loops. Otherwise, use the **ergm** term(s) `nodeocov()` or `nodeofactor()`.

### Usage

```
# binary: sendercov(attrname, force.factor=FALSE, mean=0, var=9)
```

```
# valued: sendercov(attrname, force.factor=FALSE, mean=0, var=9)
```

### Arguments

<code>attrname</code>	a character string giving the name of an attribute in the network's vertex attribute list.
<code>force.factor</code>	logical, indicating if <code>attrname</code> 's value should be interpreted as categorical even if numeric.
<code>mean, var</code>	prior mean and variance.

### Note

This term can only be used with directed networks.

### See Also

[ergmTerm](#) for index of model terms currently visible to the package.

**Keywords:** None

---

simulate.ergmm	<i>Draw from the distribution of an Exponential Random Graph Mixed Model</i>
----------------	--

---

### Description

If passed a [ergmm](#) fit object, `simulate` is used to simulate networks from the posterior of an exponential random graph mixed model fit. Alternatively, a `ergmm.model` can be passed to simulate based on a particular parameter configuration.

### Usage

```
## S3 method for class 'ergmm'
simulate(object, nsim = 1, seed = NULL, ...)

## S3 method for class 'ergmm.model'
simulate(object, nsim = 1, seed = NULL, par, prior = list(), ...)
```

### Arguments

<code>object</code>	either an object of class <a href="#">ergmm</a> for posterior simulation, or an object of class <code>ergmm.model</code> for a specific model.
<code>nsim</code>	number of networks to draw (independently)
<code>seed</code>	random seed to use; defaults to using the current state of the random number generator
<code>...</code>	Additional arguments. Currently unused.
<code>par</code>	a list with the parameter configuration based on which to simulate
<code>prior</code>	a list with the prior distribution parameters that deviate from their defaults

### Details

A sample of networks is randomly drawn from the specified model. If a needed value of `par` is missing, it is generated from its prior distribution.

### Value

If `nsim = 1`, `simulate` returns an object of class [network](#). Otherwise, an object of class `network.series` that is a list consisting of the following elements:

<code>\$formula</code>	The formula used to generate the sample.
<code>\$networks</code>	A list of the generated networks.

### See Also

[ergmm](#), [network](#), [print.network](#)

## Examples

```
#
# Fit a short MCMC run: just the MCMC.
#
data(sampson)
gest <- ergmm(samplike ~ euclidean(d=2,G=3),
              control=ergmm.control(burnin=100,interval=5,sample.size=100),tofit="mcmc")
#
# Draw from the posterior
#
g.sim <- simulate(gest)
plot(g.sim)
#
# Draw from the first draw from the posterior
#
g.sim <- with(gest,simulate(model,par=sample[[1]],prior=prior))
plot(g.sim)
```

---

socialitycov-ergmTerm *Sociality covariate effect*

---

## Description

If the attribute is numeric, this term adds one covariate to the model equaling `attrname(i)`. If the attribute is not numeric or `force.factor==TRUE`, this term adds  $p-1$  covariates to the model, where  $p$  is the number of unique values of `attrname`. The  $k$ th such covariate has the value `attrname(i) == value(k+1)`, where `value(k)` is the  $k$ th smallest unique value of the `attrname` attribute. This term only makes sense if the network is directed.

**Important:** This term works in **latentnet**'s `ergmm()` only. Using it in `ergm()` will result in an error. It is recommended to only use this term if your network contains self-loops. Otherwise, use the **ergm** term(s) `nodecov()` or `nodefactor()`.

## Usage

```
# binary: socialitycov(attrname, force.factor=FALSE, mean=0, var=9)
# valued: socialitycov(attrname, force.factor=FALSE, mean=0, var=9)
```

## Arguments

<code>attrname</code>	a character string giving the name of an attribute in the network's vertex attribute list.
<code>force.factor</code>	logical, indicating if <code>attrname</code> 's value should be interpreted as categorical even if numeric.
<code>mean, var</code>	prior mean and variance.

**See Also**

[ergmTerm](#) for index of model terms currently visible to the package.

**Keywords:** None

---

summary.ergmm

*ERGMM Fit Summaries*


---

**Description**

summary.ergmm produces a summary of an [ergmm](#) object, including point estimates, standard errors, and BIC calculation.

**Usage**

```
## S3 method for class 'ergmm'
summary(
  object,
  point.est = c(if (!is.null(object[["mle"]])) "mle", if (!is.null(object[["sample"]]))
    c("pmean", "mkl")),
  quantiles = c(0.025, 0.975),
  se = "mle" %in% point.est,
  bic.eff.obs = c("ties", "dyads", "actors"),
  ...
)
```

**Arguments**

object	An <a href="#">ergmm</a> object to be summarized.
point.est	Point estimates to compute: a character vector with some subset of "mle", "pmean", "mkl", and "pmode". Defaults to a concatenation of "mle" (if fit), "pmean", and "mkl" (if MCMC was run).
quantiles	Posterior quantiles (credible intervals) to compute.
se	Whether to compute standard errors. Defaults to TRUE if MLE was fit.
...	Additional arguments.
eff.obs, bic.eff.obs	What effective sample size to use for BIC calculation? "ties" the number of non-missing ties in the network. This is the approach recommended by Handcock et al. (2007) and the default. Not well-defined for valued networks. "dyads" the number of non-missing dyads (potential ties) in the network. "actors" the number of actors in the network. The default prior to 2.7.0. <b>a number</b> to specify a specific sample size. NULL Don't compute the BIC at all. Mostly for internal use.

## Details

Note that BIC computed for the random effects models uses the same formulation as Handcock et al., so it is likely correct, but has not been peer-reviewed.

This BIC can be (reasonably) safely used to select the number of clusters or which fixed effects to include in the model. It is not clear whether it is appropriate to use this BIC to select the dimension of latent space and whether or not to include random actor effects. These considerations are independent of the bug described below.

Prior to version 2.7.0, there was a bug in BIC calculation that used  $p + n(d + r + s)$  as the number of parameters in the likelihood (where  $p$  is the number of fixed effects,  $n$  the number of actors,  $d$ , the latent space dimension, and  $r$  and  $s$  indicators of presence of sender and receiver (or sociality) effects). This value should have been just  $p$ .

The following applications could have produced different results:

- Using the BIC to select latent space dimension.
- Using the BIC to decide whether or not to include random effects.

The following applications could not (i.e., would be off by a constant):

- Using the BIC to select the number of clusters.
- Using the BIC to select the fixed effects to be used.

## Value

For summary, an object of class `summary.ergmm`. A print method is available.

The BICs are available as the element "bic" of the object returned.

`bic.ergmm` returns the BIC for the model directly.

## References

Chris Fraley and Adrian E. Raftery (2002). *Model-based clustering, discriminant analysis, and density estimation*. Journal of the American Statistical Association, 97(458), 611-631.

Mark S. Handcock, Adrian E. Raftery and Jeremy Tantrum (2007). *Model-Based Clustering for Social Networks*. Journal of the Royal Statistical Society: Series A (Statistics in Society), 170(2), 301-354.

## See Also

[ergmm.object](#), [ergmm](#)

## Examples

```
data(sampson)
# Fit the model for cluster sizes 1 through 4:
fits<-list(
  ergmm(samplike~euclidean(d=2,G=1)),
  ergmm(samplike~euclidean(d=2,G=2)),
  ergmm(samplike~euclidean(d=2,G=3)),
```

```

    ergmm(samplike~euclidean(d=2,G=4))
  )

## Not run:
# Optionally, plot all fits.
lapply(fits,plot)

## End(Not run)

# Compute the BICs for the fits and plot them:
(bics<-reshape(
  as.data.frame(t(sapply(fits,
    function(x)c(G=x$model$G,unlist(bic.ergmm(x))[c("Y","Z","overall")])))),
  list(c("Y","Z","overall"),idvar="G",v.names="BIC",timevar="Component",
  times=c("likelihood","clustering","overall"),direction="long"
  ))

with(bics,interaction.plot(G,Component,BIC,type="b",xlab="Clusters", ylab="BIC"))

# Summarize and plot whichever fit has the lowest overall BIC:
bestG<-with(bics[bics$Component=="overall",],G[which.min(BIC)])
summary(fits[[bestG]])
plot(fits[[bestG]])

```

---

tribes

*Read Highland Tribes*


---

### Description

A network of political alliances and enmities among the 16 Gahuku-Gama sub-tribes of Eastern Central Highlands of New Guinea, documented by Read (1954).

### Format

An undirected [network](#) object with no loops, having the following attributes:

%v% "vertex.names" Character attribute with names of tribes.

%e% "pos" Logical attribute indicating an alliance relationship.

%e% "neg" Logical attribute indicating a hostile relationship ("rova").

%e% "sign" Numeric attribute coding -1 for enmity, 0 for no relationship, and 1 for alliance.

%e% "sign.012" Numeric attribute coding 0 for enmity, 1 for no relationship, and 2 for alliance.

Because of limitations of [network](#) objects, the object itself is a complete graph, and is thus meaningless if used directly or plotted.

### Details

This network shows 3 clusters.

## Source

<https://github.com/bavla/Nets/blob/master/data/Pajek/ucinet/README.md>, with corrections from Read (1954).

## References

Taken from UCINET IV, which cites the following: Hage P. and Harary F. (1983). Structural models in anthropology. Cambridge: Cambridge University Press. (See p 56-60). Read K. (1954). Cultures of the central highlands, New Guinea. Southwestern Journal of Anthropology, 10, 1-43.

## Examples

```
data(tribes)
# Only model positive ties:
tribes.fit<-ergmm(tribes~euclidean(d=2,G=3),response="pos")
# Edge color must be set manually, for green ties to represent alliance
# and for red ties to represent enmity.
plot(tribes.fit,edge.col=as.matrix(tribes,"pos",m="a")*3+as.matrix(tribes,"neg",m="a")*2, pie=TRUE)
# Model both positive and negative ties:
tribes.fit3<-ergmm(tribes~euclidean(d=2,G=3),response="sign.012",
                  family="binomial.logit",fam.par=list(trials=2))
# Edge color must be set manually, for green ties to represent alliance
# and for red ties to represent enmity.
plot(tribes.fit3,edge.col=as.matrix(tribes,"pos",m="a")*3+as.matrix(tribes,"neg",m="a")*2, pie=TRUE)
```

# Index

- \* **bipartite**
  - socialitycov-ergmTerm, 40
- \* **categorical nodal attribute**
  - loopfactor-ergmTerm, 25
- \* **cluster**
  - simulate.ergmm, 39
  - tribes, 43
- \* **datagen**
  - simulate.ergmm, 39
- \* **data**
  - davis, 9
- \* **debugging**
  - as.mcmc.list.ergmm, 4
  - mcmc.diagnostics.ergmm, 27
- \* **directed**
  - bilinear-ergmTerm, 5
  - euclidean-ergmTerm, 19
  - euclidean2-ergmTerm, 20
  - Intercept-ergmTerm, 23
  - latentcov-ergmTerm, 24
  - loopcov-ergmTerm, 25
  - loopfactor-ergmTerm, 25
  - loops-ergmTerm, 26
  - receivercov-ergmTerm, 34
  - rreceiver-ergmTerm, 35
  - rsender-ergmTerm, 36
  - rsociality-ergmTerm, 37
  - sendercov-ergmTerm, 38
- \* **distribution**
  - predict.ergmm, 33
- \* **dyad-independent**
  - bilinear-ergmTerm, 5
  - euclidean-ergmTerm, 19
  - euclidean2-ergmTerm, 20
  - Intercept-ergmTerm, 23
  - latentcov-ergmTerm, 24
  - loopcov-ergmTerm, 25
  - loopfactor-ergmTerm, 25
  - loops-ergmTerm, 26
- receivercov-ergmTerm, 34
- rreceiver-ergmTerm, 35
- rsender-ergmTerm, 36
- rsociality-ergmTerm, 37
- sendercov-ergmTerm, 38
- socialitycov-ergmTerm, 40
- \* **graphs**
  - as.mcmc.list.ergmm, 4
  - control.ergmm, 7
  - ergmm, 11
  - ergmm-class, 14
  - ergmm-families, 15
  - ergmm.drawpie, 16
  - ergmm.par.list, 17
  - ergmm.prior, 18
  - latentnet-package, 2
  - mcmc.diagnostics.ergmm, 27
  - merge.ergmm, 28
  - plot.ergmm, 29
  - predict.ergmm, 33
  - simulate.ergmm, 39
  - summary.ergmm, 41
  - tribes, 43
- \* **hplot**
  - mcmc.diagnostics.ergmm, 27
  - plot.ergmm, 29
- \* **latent**
  - bilinear-ergmTerm, 5
  - euclidean-ergmTerm, 19
  - euclidean2-ergmTerm, 20
  - Intercept-ergmTerm, 23
  - latentcov-ergmTerm, 24
  - loopcov-ergmTerm, 25
  - loopfactor-ergmTerm, 25
  - loops-ergmTerm, 26
  - receivercov-ergmTerm, 34
  - rreceiver-ergmTerm, 35
  - rsender-ergmTerm, 36
  - rsociality-ergmTerm, 37

- sendercov-ergmTerm, 38
- socialitycov-ergmTerm, 40
- \* **list**
  - ergmm.par.list, 17
- \* **loops**
  - receivercov-ergmTerm, 34
  - sendercov-ergmTerm, 38
  - socialitycov-ergmTerm, 40
- \* **manip**
  - as.mcmc.list.ergmm, 4
  - ergmm.par.list, 17
- \* **methods**
  - ergmm.par.list, 17
- \* **misc**
  - control.ergmm, 7
- \* **models**
  - ergmm-class, 14
  - ergmm-families, 15
  - ergmm.prior, 18
  - gof.ergmm, 21
  - latentnet-package, 2
  - merge.ergmm, 28
  - predict.ergmm, 33
  - simulate.ergmm, 39
  - summary.ergmm, 41
- \* **multivariate**
  - tribes, 43
- \* **nonlinear**
  - latentnet-package, 2
  - simulate.ergmm, 39
- \* **nonparametric**
  - latentnet-package, 2
  - simulate.ergmm, 39
- \* **package**
  - latentnet-package, 2
- \* **print**
  - summary.ergmm, 41
- \* **quantitative nodal attribute**
  - loopcov-ergmTerm, 25
- \* **regression**
  - ergmm-class, 14
  - ergmm-families, 15
  - latentnet-package, 2
- \* **undirected**
  - bilinear-ergmTerm, 5
  - euclidean-ergmTerm, 19
  - euclidean2-ergmTerm, 20
  - Intercept-ergmTerm, 23
  - latentcov-ergmTerm, 24
  - loopcov-ergmTerm, 25
  - loopfactor-ergmTerm, 25
  - loops-ergmTerm, 26
  - rsociality-ergmTerm, 37
  - socialitycov-ergmTerm, 40
- \* **utilities**
  - ergmm.par.list, 17
  - [.ergmm.par.list (ergmm.par.list), 17
  - [[.ergmm.par.list (ergmm.par.list), 17
  - \$.ergmm.par.list (ergmm.par.list), 17
  - 1-ergmTerm (Intercept-ergmTerm), 23
  - as.ergmm.par.list (ergmm.par.list), 17
  - as.mcmc.ergmm (as.mcmc.list.ergmm), 4
  - as.mcmc.list.ergmm, 4
  - as.mcmc.list.ergmm(), 14, 15
  - as.mcmc.list.ergmm.par.list (ergmm.par.list), 17
  - autocorr, 28
  - bic.ergmm (summary.ergmm), 41
  - bilinear-ergmTerm, 5
  - control.ergmm, 7, 12
  - davis, 9
  - del.iteration (ergmm.par.list), 17
  - dlpY.ddispersion.fs (ergmm-families), 15
  - dlpY.ddispersion.normal.identity (ergmm-families), 15
  - dlpY.deta.Bernoulli.logit (ergmm-families), 15
  - dlpY.deta.binomial.logit (ergmm-families), 15
  - dlpY.deta.fs (ergmm-families), 15
  - dlpY.deta.normal.identity (ergmm-families), 15
  - dlpY.deta.Poisson.log (ergmm-families), 15
  - edgecov(), 24
  - ergm, 22
  - ergm(), 5, 19, 20, 23–26, 34–38, 40
  - ergmm, 4–6, 9, 11, 12, 14, 18–22, 27–30, 32, 33, 35–37, 39, 41, 42
  - ERGMM fit object, 12
  - ergmm(), 3, 5, 11, 14, 15, 18–20, 23–26, 34–38, 40

- ergmm-class, 14
- ergmm-families, 15
- ergmm.control(control.ergmm), 7
- ergmm.control(), 3
- ergmm.drawpie, 16
- ergmm.families(ergmm-families), 15
- ergmm.object, 28, 29, 32, 42
- ergmm.object(ergmm-class), 14
- ergmm.par.list, 14, 17, 17
- ergmm.prior, 12, 18
- ergmTerm, 3, 6, 11–13, 18, 20, 21, 24–26, 34–38, 41
- euclidean-ergmTerm, 19
- euclidean2-ergmTerm, 20
- EY.Bernoulli.logit(ergmm-families), 15
- EY.binomial.logit(ergmm-families), 15
- EY.fs(ergmm-families), 15
- EY.normal.identity(ergmm-families), 15
- EY.Poisson.log(ergmm-families), 15
  
- fam.par.check(ergmm-families), 15
- families.ergmm, 3, 12, 13
- families.ergmm(ergmm-families), 15
- family, 13
- family(ergmm-families), 15
  
- gof, 21, 22
- gof(gof.ergmm), 21
- gof.ergmm, 21, 22
  
- InitErgmTerm.bilinear  
(bilinear-ergmTerm), 5
- InitErgmTerm.euclidean  
(euclidean-ergmTerm), 19
- InitErgmTerm.euclidean2  
(euclidean2-ergmTerm), 20
- InitErgmTerm.Intercept  
(Intercept-ergmTerm), 23
- InitErgmTerm.latentcov  
(latentcov-ergmTerm), 24
- InitErgmTerm.loopcov  
(loopcov-ergmTerm), 25
- InitErgmTerm.loopfactor  
(loopfactor-ergmTerm), 25
- InitErgmTerm.loops(loops-ergmTerm), 26
- InitErgmTerm.receivercov  
(receivercov-ergmTerm), 34
- InitErgmTerm.rreceiver  
(rreceiver-ergmTerm), 35
  
- InitErgmTerm.rsender  
(rsender-ergmTerm), 36
- InitErgmTerm.rsociality  
(rsociality-ergmTerm), 37
- InitErgmTerm.sendercov  
(sendercov-ergmTerm), 38
- InitErgmTerm.socialitycov  
(socialitycov-ergmTerm), 40
- intercept, 12
- Intercept-ergmTerm, 23
- intercept-ergmTerm  
(Intercept-ergmTerm), 23
  
- latent(ergmm), 11
- latentcluster(ergmm), 11
- latentcov-ergmTerm, 24
- latentnet(latentnet-package), 2
- latentnet-package, 2
- length.ergmm.par.list(ergmm.par.list), 17
- lm, 23
- lm(), 12
- loopcov-ergmTerm, 25
- loopfactor-ergmTerm, 25
- loops-ergmTerm, 26
- lpY.Bernoulli.logit(ergmm-families), 15
- lpY.binomial.logit(ergmm-families), 15
- lpY.fs(ergmm-families), 15
- lpY.normal.identity(ergmm-families), 15
- lpY.Poisson.log(ergmm-families), 15
- lpYc.Bernoulli.logit(ergmm-families), 15
- lpYc.binomial.logit(ergmm-families), 15
- lpYc.fs(ergmm-families), 15
- lpYc.normal.identity(ergmm-families), 15
- lpYc.Poisson.log(ergmm-families), 15
  
- mcmc.diagnostics  
(mcmc.diagnostics.ergmm), 27
- mcmc.diagnostics.ergmm, 5, 27
- mcmc.list, 4, 5
- merge, 28
- merge.ergmm, 28
  
- network, 9, 13, 39, 43
- nodecov(), 40
- nodefactor(), 40
- nodeicov(), 34

nodeifactor(), [34](#)  
nodeocov(), [38](#)  
nodeofactor(), [38](#)

par, [31](#)  
plot, [32](#)  
plot(), [3](#)  
plot.ergmm, [16](#), [29](#), [29](#)  
plot.ergmm(), [3](#), [14](#), [15](#)  
plot.gof, [22](#)  
plot.mcmc.list, [28](#)  
plot.network, [30–32](#)  
plot3d.ergmm(plot.ergmm), [29](#)  
predict.ergmm, [33](#)  
predict.ergmm(), [14](#), [15](#)  
print(), [3](#)  
print.ergmm(ergmm-class), [14](#)  
print.ergmm(), [14](#)  
print.network, [39](#)  
print.summary.ergmm(summary.ergmm), [41](#)  
pY.Bernoulli.logit(ergmm-families), [15](#)  
pY.binomial.logit(ergmm-families), [15](#)  
pY.fs(ergmm-families), [15](#)  
pY.normal.identity(ergmm-families), [15](#)  
pY.Poisson.log(ergmm-families), [15](#)

raftery.diag, [28](#)  
receivercov-ergmTerm, [34](#)  
rreceiver-ergmTerm, [35](#)  
rsender-ergmTerm, [36](#)  
rsm.Bernoulli.logit(ergmm-families), [15](#)  
rsm.binomial.logit(ergmm-families), [15](#)  
rsm.fs(ergmm-families), [15](#)  
rsm.normal.identity(ergmm-families), [15](#)  
rsm.Poisson.log(ergmm-families), [15](#)  
rsociality-ergmTerm, [37](#)

sendercov-ergmTerm, [38](#)  
show.ergmm(ergmm-class), [14](#)  
simulate(simulate.ergmm), [39](#)  
simulate.ergmm, [22](#), [39](#)  
snowFT, [8](#)  
socialitycov-ergmTerm, [40](#)  
summary(), [3](#)  
summary.ergmm, [13](#), [41](#), [42](#)  
summary.ergmm(), [14](#), [15](#)

tribes, [43](#)

unstack.ergmm.par.list  
(ergmm.par.list), [17](#)