

Package ‘hergm’

February 14, 2012

Version 1.3-5

Date 2011-08-20

Title Hierarchical Exponential-Family Random Graph Models

Author Michael Schweinberger <michael.schweinberger@stat.psu.edu>

Maintainer Michael Schweinberger <michael.schweinberger@stat.psu.edu>

Depends R (>= 2.10.0), ergm, snow

Suggests

Description The R package ‘hergm’ implements Hierarchical Exponential-Family Random Graph Models (HERGMs), which can be used to model a wide range of relational data (networks). ‘hergm’ implements both simulation and Bayesian inference.

License GPL-3

URL

BugReports

Repository CRAN

Date/Publication 2011-08-21 05:08:03

R topics documented:

example	2
hergm	2
hergm-postprocess	5
hergm-terms	7

Index	10
--------------	-----------

example

Example data set

Description

Example data set: synthetic, undirected network with 15 nodes.

Usage

```
data(example)
```

Value

Undirected network.

See Also

network, hergm, ergm.terms, hergm.terms, hergm.postprocess

Examples

```
## Not run: # Load undirected network with 15 nodes
data(example)

# p_1 model for undirected network with Dirichlet process prior
hergm(d ~ edges_i)

# Hierarchical exponential-family model with stick-breaking prior
hergm(d ~ edges + triangle_ijk)

## End(Not run)
```

hergm

Hierarchical Exponential-Family Random Graph Models: Simulation and Bayesian inference

Description

The package `hergm` implements exponential-family random graph models with Dirichlet process / stick-breaking priors, including

- the `p_1` model for directed networks of Holland and Leinhardt (1981) and its extension to undirected random graph models with Dirichlet process priors (see `arcs_i`, `arcs_j`, `mutual_i`, and `edges_i`). While the `p_1` model for undirected and directed networks with parametric priors contains $O(n)$ parameters (n = number of nodes) and therefore is not parsimonious, the non-parametric Dirichlet

process prior encourages a small number of unique parameters and therefore represents an elegant alternative to parametric priors.

- the stochastic block model of Snijders and Nowicki (1997) and Nowicki and Snijders (2001) with natural parameterization (restricted between-block parameters) and Dirichlet process priors (see `edges_ij`).
- the hierarchical exponential-family models of Schweinberger and Handcock (2009) with stick-breaking priors (see `mutual_ij`, `twostar_ijk`, `triangle_ijk`, `ttriple_ijk`, `ctriple_ijk`). Hierarchical exponential-family models replace the strong dependence of simple exponential-family models by weak dependence with an eye to solving the near-degeneracy problem of simple exponential-family model.

The package `hergm` implements simulation and Bayesian inference for the mentioned models.

Usage

```
hergm(formula,
      alpha = NULL,
      alpha_shape = NULL,
      alpha_rate = NULL,
      eta = NULL,
      eta_mean = NULL,
      eta_sd = NULL,
      eta_mean_mean = NULL,
      eta_mean_sd = NULL,
      eta_precision_shape = NULL,
      eta_precision_rate = NULL,
      indicator = NULL,
      parallel = 1,
      simulate = FALSE,
      seeds = NULL,
      samplesize = 1e+5,
      burnin = 1e+4,
      interval = 1e+2,
      mh_scale = NULL,
      output = TRUE,
      verbose = -1,
      name = NULL,
      ...)
```

Arguments

- | | |
|--|---|
| <code>formula</code> | formula of the form <code>network ~ terms</code> . Networks can be created by calling the function <code>network</code> . Possible terms can be found in <code>ergm.terms</code> and <code>hergm.terms</code> . |
| <code>alpha</code> | scaling parameter of truncated Dirichlet process / stick-breaking prior of natural parameters of exponential-family model. |
| <code>alpha_shape</code> , <code>alpha_rate</code> | shape and rate (inverse scale) parameter of Gamma prior of scaling parameter. |

<code>eta</code>	natural parameters of exponential-family model.
<code>eta_mean, eta_sd</code>	means and standard deviations of Gaussian baseline distribution of Dirichlet process / stick-breaking prior of natural parameters.
<code>eta_mean_mean, eta_mean_sd</code>	means and standard deviations of Gaussian prior of mean of Gaussian baseline distribution of Dirichlet process / stick-breaking prior.
<code>eta_precision_shape, eta_precision_rate</code>	shape and rate (inverse scale) parameter of Gamma prior of precision parameter of Gaussian baseline distribution of Dirichlet process / stick-breaking prior.
<code>indicator</code>	if <code>simulate == TRUE</code> and indicator of block membership is specified, condition on specified indicator (of block membership).
<code>parallel</code>	number of processors; if more than one, computing is parallel.
<code>simulate</code>	if <code>TRUE</code> , simulation of networks, otherwise Bayesian inference.
<code>seeds</code>	seed of pseudo-random number generator; if computing is parallel, number of seeds must equal number of processors.
<code>samplesize</code>	if <code>simulate == TRUE</code> , number of networks to be sampled, otherwise number of draws from posterior; if computing is parallel, number of draws per processors.
<code>burnin</code>	if <code>simulate == TRUE</code> , number of burn-in iterations.
<code>interval</code>	if <code>simulate == TRUE</code> , number of proposals between sampled networks.
<code>mh_scale</code>	if <code>simulate == FALSE</code> , scale factor of candidate-generating distribution of Metropolis-Hastings algorithm.
<code>output</code>	if <code>TRUE</code> , full output, otherwise limited output.
<code>name</code>	name of project; if <code>output == TRUE</code> , name of project is used to name output files.
<code>verbose</code>	console output: -1: no output; 0: short output; +1: long output.
<code>...</code>	additional arguments, to be passed to lower-level functions in the future.

Value

If called with the option `simulate = TRUE`, the function `hergm` returns a sample of networks, otherwise a raw MCMC sample from the posterior. To postprocess the sample, call the function `hergm.postprocess`.

References

- Holland, P. W. and S. Leinhardt (1981). An exponential family of probability distributions for directed graphs. *Journal of the American Statistical Association* 76 (373), 33–65.
- Nowicki, K. and T. A. B. Snijders (2001). Estimation and prediction for stochastic blockstructures. *Journal of the American Statistical Association* 96 (455), 1077–1087.
- Schweinberger, M. and M. S. Handcock (2009). Hierarchical exponential-family random graph models. Technical report, Pennsylvania State University. Submitted.
- Snijders, T. A. B. and K. Nowicki (1997). Estimation and prediction for stochastic blockmodels for graphs with latent block structure. *Journal of Classification* 14, 75–100.

See Also

network, ergm.terms, hergm.terms, hergm.postprocess

Examples

```
## Not run: # Load undirected network with 15 nodes (see ?example)
data(example)

# p_1 model for undirected network with Dirichlet process prior
hergm(d ~ edges_i)

# Stochastic block model for undirected network
# with natural parameterization and Dirichlet process prior
hergm(d ~ edges_ij)

# Hierarchical exponential-family model with stick-breaking prior
hergm(d ~ edges + triangle_ijk)

## End(Not run)
```

hergm-postprocess	<i>Hierarchical Exponential-Family Random Graph Models: Postprocessing Samples</i>
-------------------	--

Description

If called with the option `simulate = TRUE`, the function `hergm` returns a sample of networks, otherwise a raw MCMC sample from the posterior. The function `hergm.postprocess` postprocesses samples: if called with the `relabel = FALSE`, `hergm.postprocess` extracts information of interest, otherwise it solves, in addition, the so-called label-switching problem. The label-switching problem is rooted in the invariance of the likelihood function to permutations of the labels of blocks, and implies that the raw MCMC sample cannot be used to infer to block-dependent entities. The label-switching problem can be solved in a Bayesian decision-theoretic framework: by defining a loss function and minimizing the posterior expected loss. Calling `hergm.process` minimizes the posterior expected loss using a simple and convenient loss function. The required computations can be time-consuming when the number of blocks `k` is large.

Usage

```
hergm.postprocess(sample = NULL,
                  burnin = 0,
                  thinning = 1,
                  relabel = FALSE,
                  name = NULL,
                  ...)
```

Arguments

sample	MCMC sample generated by function <code>hergm</code> .
burnin	number of burn-in iterations; if computing is parallel, number of burn-in iterations per processor.
thinning	if <code>thinning > 1</code> , every <code>thinning</code> -th sample point is used while all others discarded.
relabel	if TRUE, relabel MCMC sample.
name	name of project; if <code>output == TRUE</code> , name of project is used to name output files.
...	additional arguments, to be passed to lower-level functions in the future.

Value

<code>ergm_theta</code>	parameters of <code>ergm</code> -terms.
<code>alpha</code>	scaling parameter of truncated Dirichlet process / stick-breaking prior of parameters of <code>hergm</code> -terms.
<code>eta_mean</code>	mean parameters of Gaussian base distribution of parameters of <code>hergm</code> -terms.
<code>eta_precision</code>	precision parameters of Gaussian base distribution of parameters of <code>hergm</code> -terms.
<code>hergm_theta</code>	parameters of <code>hergm</code> -terms.
<code>p_k</code>	probabilities of membership to blocks.
<code>indicator</code>	indicators of memberships of nodes.
<code>p_i_k</code>	probabilities of membership of nodes to blocks.
<code>prediction</code>	posterior predictions of statistics.

See Also

`network`, `hergm.example`, `hergm`, `ergm.terms`, `hergm.terms`

Examples

```
## Not run: # Load undirected network with 15 nodes (see ?example)
data(example)

# Generate MCMC sample of size 1,000
mcmc <- hergm(d ~ edges_i, parallel = 1, samplesize = 1000)
# Postprocess MCMC sample of size 1,000
processed_mcmc <- hergm.postprocess(sample = mcmc, burnin = 200)

# Generate MCMC sample of size 20 * 100 = 2,000
mcmc <- hergm(d ~ edges + triangle_ijk, parallel = 20, samplesize = 100)
# Postprocess MCMC sample of size 20 * 100 = 2,000
processed_mcmc <- hergm.postprocess(sample = mcmc, burnin = 20)

## End(Not run)
```

Description

Hierarchical Exponential-Family Random Graph Models can be specified by calling the function `hergm(formula)`, where `formula` is a formula of the form `network ~ terms`.

By using suitable terms, it is possible to specify

- the `p_1` model for directed networks of Holland and Leinhardt (1981) and its extension to undirected random graph models with Dirichlet process priors (see `arcs_i`, `arcs_j`, `mutual_i`, and `edges_i`). While the `p_1` model for undirected and directed networks with parametric priors contains $O(n)$ parameters (n = number of nodes) and therefore is not parsimonious, the non-parametric Dirichlet process prior encourages a small number of unique parameters and therefore represents an elegant alternative to parametric priors.
- the stochastic block model of Snijders and Nowicki (1997) and Nowicki and Snijders (2001) with natural parameterization (restricted between-block parameters) and Dirichlet process priors (see `edges_ij`).
- the hierarchical exponential-family models of Schweinberger and Handcock (2009) with stick-breaking priors (see `mutual_ij`, `twostar_ijk`, `triangle_ijk`, `ttriple_ijk`, `ctruple_ijk`). Hierarchical exponential-family models replace the strong dependence of simple exponential-family models by weak dependence with an eye to solving the near-degeneracy problem of simple exponential-family model.

`hergm.terms` can be found here. Additional terms, e.g. covariate-dependent terms, can be found in `ergm.terms`.

Arguments

- `edges_i(k)` (undirected network)
adding the term `edges_i` to the model adds node-dependent edge terms to the model; the optional argument `k` is the maximum number of blocks (default: the number of nodes).
- `arcs_i(k)` (directed network)
adding the term `arcs_i` to the model adds node-dependent outdegree terms to the model; the optional argument `k` is the maximum number of blocks (default: the number of nodes).
- `arcs_j(k)` (directed network)
adding the term `arcs_j` to the model adds node-dependent indegree terms to the model; the optional argument `k` is the maximum number of blocks (default: the number of nodes).
- `edges_ij(k)` (undirected, directed network)
adding the term `edges_ij` to the model adds block-dependent edge terms to the model. the optional argument `k` is the maximum number of blocks (default: the number of nodes).
- `mutual_i(k)` (directed network)
adding the term `mutual_i` to the model adds additive, block-dependent mutual edge terms to the model. the optional argument `k` is the maximum number of blocks (default: the number of nodes).

- `mutual_ij(k)` (directed network)
 adding the term `mutual_ij` to the model adds block-dependent mutual edge terms to the model. the optional argument `k` is the maximum number of blocks (default: the number of nodes).
- `twostar_ijk(k)` (undirected network)
 adding the term `twostar_ijk` to the model adds block-dependent two-star terms to the model. the optional argument `k` is the maximum number of blocks (default: the number of nodes).
- `triangle_ijk(k)` (undirected, directed network)
 adding the term `triangle_ijk` to the model adds block-dependent triangle terms to the model. the optional argument `k` is the maximum number of blocks (default: the number of nodes).
- `ttriple_ijk(k)` (directed network)
 adding the term `ttriple_ijk` to the model adds block-dependent transitive triple terms to the model. the optional argument `k` is the maximum number of blocks (default: the number of nodes).
- `ctriple_ijk(k)` (directed network)
 adding the term `ctriple_ijk` to the model adds block-dependent cyclic triple terms to the model. the optional argument `k` is the maximum number of blocks (default: the number of nodes).

References

- Holland, P. W. and S. Leinhardt (1981). An exponential family of probability distributions for directed graphs. *Journal of the American Statistical Association* 76 (373), 33–65.
- Nowicki, K. and T. A. B. Snijders (2001). Estimation and prediction for stochastic blockstructures. *Journal of the American Statistical Association* 96 (455), 1077–1087.
- Schweinberger, M. and M. S. Handcock (2009). Hierarchical exponential-family random graph models. Technical report, Pennsylvania State University. Submitted.
- Snijders, T. A. B. and K. Nowicki (1997). Estimation and prediction for stochastic blockmodels for graphs with latent block structure. *Journal of Classification* 14, 75–100.

See Also

`network`, `hergm`, `ergm.terms`, `hergm.postprocess`

Examples

```
## Not run: # Load undirected network with 15 nodes (see ?example)
data(example)
# p_1 model for undirected network with Dirichlet process prior
hergm(d ~ edges_i)

# Load directed network with 18 nodes (see ?sampson)
data(sampson)
# p_1 model for directed network with Dirichlet process prior
hergm(samplike ~ arcs_i + arcs_j + mutual)

# Load undirected network with 15 nodes (see ?example)
```

```
data(example)
# Stochastic block model for undirected network
# with natural parameterization and Dirichlet process prior
hergm(d ~ edges_ij)

# Load directed network with 18 nodes (see ?sampson)
data(sampson)
# Stochastic block model for directed network
# with natural parameterization and Dirichlet process prior
hergm(samplike ~ edges_ij + mutual)

# Load undirected network with 15 nodes (see ?example)
data(example)
# Hierarchical exponential-family model with stick-breaking prior
hergm(d ~ edges + mutual + ttriple_ijk)

# Load directed network with 18 nodes (see ?sampson)
data(sampson)
# Hierarchical exponential-family model with stick-breaking prior
hergm(samplike ~ edges + mutual + ttriple_ijk)

## End(Not run)
```

Index

`arcs_i` (`hergm-terms`), [7](#)
`arcs_j` (`hergm-terms`), [7](#)

`ctriple_ijk` (`hergm-terms`), [7](#)

`d` (`example`), [2](#)

`edges_i` (`hergm-terms`), [7](#)
`edges_ij` (`hergm-terms`), [7](#)
`example`, [2](#)

`hergm`, [2](#)
`hergm-postprocess`, [5](#)
`hergm-terms`, [7](#)
`hergm.postprocess` (`hergm-postprocess`), [5](#)
`hergm.terms` (`hergm-terms`), [7](#)

`mutual_i` (`hergm-terms`), [7](#)
`mutual_ij` (`hergm-terms`), [7](#)

`postprocess-hergm` (`hergm-postprocess`), [5](#)
`postprocess.hergm` (`hergm-postprocess`), [5](#)

`terms-hergm` (`hergm-terms`), [7](#)
`terms.hergm` (`hergm-terms`), [7](#)
`triangle_ijk` (`hergm-terms`), [7](#)
`ttriple_ijk` (`hergm-terms`), [7](#)
`twostar_ijk` (`hergm-terms`), [7](#)