

# Package ‘Bergm’

April 11, 2012

**Type** Package

**Title** Bayesian analysis for exponential random graph models

**Version** 2.3

**Date** 2012-3-30

**Author** Alberto Caimo and Nial Friel

**Maintainer** Alberto Caimo <acaimo.stats@gmail.com>

**Description** Set of tools to analyse Bayesian exponential random graph models

**License** GPL (>= 2)

**URL** <https://sites.google.com/site/albertocaimo/Bergm/>

**Depends** network, ergm, coda, mvtnorm

**LazyLoad** yes

**Repository** CRAN

**Date/Publication** 2012-04-11 13:30:24

## R topics documented:

Bergm-package . . . . .	2
bergm . . . . .	2
bergm.output . . . . .	4
bergmS . . . . .	5
bergmS.output . . . . .	7
bgof . . . . .	7

<b>Index</b>	<b>9</b>
--------------	----------

---

Bergm-package

*Bayesian analysis for exponential random graph models*

---

### Description

Bergm is a collection of functions for Bayesian exponential random graph models.

### Author(s)

Alberto Caimo <alberto.caimo@ucd.ie> and Nial Friel <nial.friel@ucd.ie>.

### References

- Caimo, A. and Friel, N. (2012b), “Bergm: Bayesian Exponential Random Graphs in R,” Tech. rep., University College Dublin. Available in e-print format at <http://arxiv.org/abs/1201.2770>.
- Caimo, A. and Friel, N. (2012a), “Bayesian model selection for exponential random graph models,” Tech. rep., University College Dublin. Available in e-print format at <http://arxiv.org/abs/1201.2337>.
- Caimo, A. and Friel, N. (2011), “Bayesian inference for exponential random graph models,” *Social Networks*, 33, 41-55.
- Hunter, D. R., Handcock, M. S., Butts, C. T., Goodreau, S. M., and Morris M. (2008), “ergm: A Package to Fit, Simulate and Diagnose Exponential-Family Models for Networks”, *Journal of Statistical Software*, 24, 1-29.
- Handcock, M. S., Hunter, D. R., Butts, C. T., Goodreau, S. M., and Morris, M. (2007), “statnet: Software tools for the representation, visualization, analysis and simulation of network data,” *Journal of Statistical Software*, 24,1-11.

---

bergm

*Bayesian exponential random graph models*

---

### Description

Function to fit Bayesian exponential random graphs models using the exchange algorithm. Two are the sampling approaches available: block update and population MCMC with parallel Adaptive Direction Sampling (ADS).

### Usage

```
bergm(formula,  
      burn.in=100,  
      main.iters=1000,  
      aux.iters=1000,  
      m.prior = NULL,  
      sigma.prior = NULL,  
      nchains = NULL,
```

```

gamma = 0.5,
sigma.epsilon = NULL,
save = FALSE,
...)
```

### Arguments

formula	formula; an R formula object, of the form <network> ~ <model terms> where <network> is a <a href="#">network</a> object and <model terms> are <a href="#">ergm-terms</a> .
burn.in	count; number of burn-in iterations at the beginning of an MCMC run. If population MCMC is performed, it refers to the number of burn-in iterations for every chain of the population.
main.iters	count; number of iterations for the MCMC chain(s) excluding burn-in. If population MCMC is performed, it refers to the number of iterations for every chain of the population.
aux.iters	count; number of auxiliary iterations used for network simulation.
m.prior	vector; mean of the multivariate Normal prior. By default set to a vector of 0's.
sigma.prior	variance/covariance matrix for the multivariate Normal prior. By default set to a diagonal matrix with every diagonal entry equal to 100.
nchains	count; number of chains of the population MCMC. By default set to twice the model dimension (number of model terms). If the model is one-dimensional, nchains is set to 1.
gamma	scalar; “parallel ADS move factor.” In case of one-dimensional models, the population MCMC procedure is disabled and gamma is used as variance of the Normal proposal distribution.
sigma.epsilon	variance/covariance matrix for the multivariate Normal proposal or “parallel ADS move parameter”. By default set to a diagonal matrix with every diagonal entry equal to 0.0025. If the model is one-dimensional, sigma.epsilon is set equal to gamma.
save	logical; if TRUE a file called "bergm.out" is saved in the working directory.
...	additional arguments, to be passed to lower-level functions.

### References

Caimo, A. and Friel, N., (2011) “Bayesian inference for exponential random graph models,” *Social Networks*, 33, 41-55.

### See Also

[bergm.output](#), [bgof](#).

### Examples

```

# load the Florentine marriage network
# included with the ergm package

data(florentine)
```

```
# Estimation of a 3-dimensional model
# measuring the propensity to form 2- and 3- stars.
# Population MCMC with ADS approach is used
# (this will take about 1 minute)

flo <- bergm(flomarriage ~ edges + kstar(2:3),
             burn.in=500,
             aux.iters=3000,
             main.iters=1500)

# MCMC diagnostics

bergm.output(flo)

# Bayesian goodness-of-fit test

bgof(flo,
      n.sim=100,
      n.deg=10,
      n.dist=9,
      n.esp=6)
```

---

bergm.output	<i>MCMC output diagnostics for the Bayesian parameter estimation algorithm</i>
--------------	--

---

## Description

This function returns the posterior parameter density estimate and creates simple diagnostic plots for the MCMC produced from a fit.

## Usage

```
bergm.output(x,
             ...)
```

## Arguments

x	an R object of class bergm.
...	additional arguments, to be passed to lower-level functions.

## See Also

[bergm.](#)

**Description**

Bayesian model selection for exponential random graphs models using the auto-RJ exchange algorithm. The algorithm consists of two steps: the first step (offline) is used to sample from the posterior of each competing model using the `bergm` function and then approximated by normal distributions determined by the moments of each sample. The second step (online step) of the algorithm makes use of the normal posterior proposal estimated in the offline step as within-model proposals for the RJ-MCMC computation.

**Usage**

```
bergmS(formulae,
       iters = 10000,
       m.priors = NULL,
       sigma.priors = NULL,
       gammas = NULL,
       nchains = NULL,
       sigma.epsilons = NULL,
       aux.iters = 1000,
       main.iters = NULL,
       burn.ins=NULL,
       save=FALSE,
       ...)
```

**Arguments**

<code>formulae</code>	list; R formula objects (competing models) of the form <code>&lt;network&gt; ~ &lt;model terms&gt;</code> , see <a href="#">ergm-terms</a> .
<code>iters</code>	count; number of iterations for auto-RJ exchange algorithm (online step).
<code>m.priors</code>	list; vectors of means of the multivariate Normal priors for each competing model. By default set to a list of vectors (one for each competing model) of 0's.
<code>sigma.priors</code>	list; variance /covariance matrices of the multivariate Normal priors for each competing model. By default set to a list of diagonal matrices (one for each competing model) with every diagonal entry equal to 100.
<code>gammas</code>	vector; "parallel ADS move factors" for each competing model (offline step). By default set to a vector of 0.5's.
<code>nchains</code>	vector; number of MCMC chains for each competing model (offline step). By default set to a vector with entries (one for each competing model) equal to twice the dimensions of the competing models.

`sigma.epsilon` list; containing variance/covariance matrices for the multivariate Normal proposals or “parallel ADS move parameters” for each competing model (offline step). By default set to a list of diagonal matrices (one for each competing model) with every diagonal entry equal to 0.0025.

`aux.iters` count; number of auxiliary iterations for network simulation.

`main.iters` vector; number of iterations for the MCMC chain(s) for each competing model (offline step). By default set to a vector of 1000’s.

`burn.ins` vector; number of burn-in iterations at the beginning of an MCMC run for each competing model (offline step). By default set to a vector of 100’s.

`save` logical; if TRUE a file called "bergmS.out" is saved in the working directory.

`...` additional arguments, to be passed to lower-level functions.

## References

Caimo, A. and Friel, N., (submitted) “Bayesian model selection for exponential random graph models.”

## See Also

[bergm](#).

## Examples

```

# load the Florentine marriage network
# included with the ergm package

data(florentine)
y <- flomarriage

# Competing models:

formulae <- c(y ~ edges,
              y ~ edges + gwesp(log(2),fixed=TRUE),
              y ~ edges + gwdegree(log(2),fixed=TRUE))

# Model selection via auto-RJ exchange algorithm
# (this will take about 3.5 minutes)

flo <- bergmS(formulae,
              iters=10000,
              aux.iters=3000,
              main.iters=rep(1000,3),
              burn.in=rep(100,3),
              gammas=c(0.8,1,1))

# MCMC diagnostics

flo.out <- bergmS.output(flo)

# Bayesian goodness-of-fit test (for the best model)

```

```
bgof(flo.out,  
     lags=200,  
     n.sim=100,  
     n.deg=10,  
     n.dist=10,  
     n.esp=5)
```

---

bergmS.output

*MCMC output diagnostics for the Bayesian model selection algorithm*

---

### Description

This function returns posterior model and parameter density estimates and creates simple diagnostic plots for the MCMC produced by the function `bergmS`.

### Usage

```
bergmS.output(x,  
             ...)
```

### Arguments

`x` an R object of class `bergmS`.  
`...` additional arguments, to be passed to lower-level functions.

### See Also

[bergmS](#).

---

bgof

*Bayesian goodness-of-fit diagnostics*

---

### Description

Calculates summaries for degree, minimum geodesic distances, and edge-wise shared partner distributions to diagnose the Bayesian goodness-of-fit of exponential random graph models.

**Usage**

```
bgof(x,
     directed = FALSE,
     sample.size=100,
     aux.iters = 10000,
     n.deg = NULL,
     n.dist = NULL,
     n.esp = NULL,
     n.ideg = NULL,
     n.odeg = NULL,
     ...)
```

**Arguments**

<code>x</code>	an R object of class <code>bergm</code> .
<code>directed</code>	logical; TRUE if the observed graph is directed.
<code>sample.size</code>	count; number of networks to be simulated and compared to the observed network.
<code>aux.iters</code>	count; number of iterations used for network simulation.
<code>n.deg</code>	count; used to plot only the first <code>n.deg-1</code> degree distributions. By default no restrictions on the number of degree distributions is applied.
<code>n.dist</code>	count; used to plot only the first <code>n.dist-1</code> geodesic distances distributions. By default no restrictions on the number of geodesic distances distributions is applied.
<code>n.esp</code>	count; used to plot only the first <code>n.esp-1</code> edge-wise shared partner distributions. By default no restrictions on the number of edge-wise shared partner distributions is applied.
<code>n.ideg</code>	count; used to plot only the first <code>n.ideg-1</code> in-degree distributions. By default no restrictions on the number of in-degree distributions is applied.
<code>n.odeg</code>	count; used to plot only the first <code>n.odeg-1</code> out-degree distributions. By default no restrictions on the number of out-degree distributions is applied.
<code>...</code>	additional arguments, to be passed to lower-level functions.

**See Also**

[bergm](#).

# Index

bergm, [2](#), [4–6](#), [8](#)  
Bergm-package, [2](#)  
bergm.output, [3](#), [4](#)  
bergmS, [5](#), [7](#)  
bergmS.output, [7](#)  
bgof, [3](#), [7](#)  
  
ergm-terms, [3](#), [5](#)  
  
network, [3](#)