

# Package ‘Bergm’

September 13, 2023

**Type** Package

**Title** Bayesian Exponential Random Graph Models

**Version** 5.0.6

**Date** 2023-09-12

**Author** Alberto Caimo [aut, cre], Lampros Bouranis [aut], Robert Krause [aut] Nial Friel [ctb]

**Maintainer** Alberto Caimo <alberto.caimo1@ucd.ie>

**Description** Bayesian analysis for exponential random graph models using advanced computational algorithms. More information can be found at: <<https://acaimo.github.io/Bergm/>>.

**License** GPL (>= 2)

**URL** <https://acaimo.github.io/Bergm/>

**Depends** ergm, R (>= 4.1)

**Imports** coda, graphics, grDevices, Matrix, matrixcalc, MCMCpack, mvtnorm, network, Rglpk, statnet.common, stats, utils

**Config/testthat/edition** 3

**Encoding** UTF-8

**Language** en-US

**LazyData** true

**RoxygenNote** 7.2.3

**Suggests** spelling

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2023-09-13 07:12:37 UTC

## R topics documented:

Bergm-package . . . . .	2
bergm . . . . .	2

bergmC . . . . .	4
bergmM . . . . .	6
bgof . . . . .	8
ergmAPL . . . . .	9
evidence . . . . .	10
evidenceCJ . . . . .	11
evidencePP . . . . .	13
lazega . . . . .	15
plot.bergm . . . . .	16
summary.bergm . . . . .	17

## Index 18

---

Bergm-package	<i>Bayesian exponential random graph models</i>
---------------	---

---

### Description

Bergm provides a range of tools to analyse Bayesian exponential random graph models using advanced computational methods.

---

bergm	<i>Parameter estimation for Bayesian ERGMs</i>
-------	--

---

### Description

Function to fit Bayesian exponential random graphs models using the approximate exchange algorithm.

### Usage

```
bergm(
  formula,
  prior.mean = NULL,
  prior.sigma = NULL,
  burn.in = 100,
  main.iters = 1000,
  aux.iters = 1000,
  nchains = NULL,
  gamma = 0.5,
  V.proposal = 0.0025,
  startVals = NULL,
  offset.coef = NULL,
  ...
)
```

## Arguments

formula	formula; an <code>ergm</code> formula object, of the form <code>&lt;network&gt; ~ &lt;model terms&gt;</code> where <code>&lt;network&gt;</code> is a <code>network</code> object and <code>&lt;model terms&gt;</code> are <code>ergm</code> -terms.
prior.mean	vector; mean vector of the multivariate Normal prior. By default set to a vector of 0's.
prior.sigma	square matrix; variance/covariance matrix for the multivariate Normal prior. By default set to a diagonal matrix with every diagonal entry equal to 100.
burn.in	count; number of burn-in iterations for every chain of the population.
main.iters	count; number of iterations for every chain of the population.
aux.iters	count; number of auxiliary iterations used for network simulation.
nchains	count; number of chains of the population MCMC. By default set to twice the model dimension (number of model terms).
gamma	scalar; parallel adaptive direction sampling move factor.
V.proposal	count; diagonal entry for the multivariate Normal proposal. By default set to 0.0025.
startVals	vector; optional starting values for the parameter estimation.
offset.coef	vector; A vector of coefficients for the offset terms.
...	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(1), 41-55. <https://arxiv.org/abs/1007.5192>

Caimo, A. and Friel, N. (2014), "Bergm: Bayesian Exponential Random Graphs in R," *Journal of Statistical Software*, 61(2), 1-25. <https://www.jstatsoft.org/article/view/v061i02>

## Examples

```
## Not run:
# Load the florentine marriage network
data(florentine)

# Posterior parameter estimation:
p.flo <- bergm(flo.marriage ~ edges + kstar(2),
              burn.in   = 50,
              aux.iters = 500,
              main.iters = 1000,
              gamma     = 1.2)

# Posterior summaries:
summary(p.flo)

## End(Not run)
```

---

 bergmC

*Calibrating misspecified Bayesian ERGMs*


---

### Description

Function to transform a sample from the pseudo-posterior to one that is approximately sampled from the intractable posterior distribution.

### Usage

```
bergmC(
  formula,
  prior.mean = NULL,
  prior.sigma = NULL,
  burn.in = 10000,
  main.iters = 40000,
  aux.iters = 3000,
  V.proposal = 1.5,
  thin = 1,
  rm.iters = 500,
  rm.a = 0.001,
  rm.alpha = 0,
  n.aux.draws = 400,
  aux.thin = 50,
  estimate = c("MLE", "CD"),
  seed = 1,
  ...
)
```

### Arguments

<code>formula</code>	formula; an <a href="#">ergm</a> formula object, of the form <code>&lt;network&gt; ~ &lt;model terms&gt;</code> where <code>&lt;network&gt;</code> is a <a href="#">network</a> object and <code>&lt;model terms&gt;</code> are <code>ergm</code> -terms.
<code>prior.mean</code>	vector; mean vector of the multivariate Normal prior. By default set to a vector of 0's.
<code>prior.sigma</code>	square matrix; variance/covariance matrix for the multivariate Normal prior. By default set to a diagonal matrix with every diagonal entry equal to 100.
<code>burn.in</code>	count; number of burn-in iterations at the beginning of an MCMC run for the pseudo-posterior estimation.
<code>main.iters</code>	count; number of MCMC iterations after burn-in for the pseudo-posterior estimation.
<code>aux.iters</code>	count; number of auxiliary iterations used for drawing the first network from the ERGM likelihood (Robbins-Monro). See <a href="#">control.simulate.formula</a> .
<code>V.proposal</code>	count; diagonal entry for the multivariate Normal proposal. By default set to 1.5.

<code>thin</code>	count; thinning interval used in the simulation for the pseudo-posterior estimation. The number of MCMC iterations must be divisible by this value.
<code>rm.iters</code>	count; number of iterations for the Robbins-Monro stochastic approximation algorithm.
<code>rm.a</code>	scalar; constant for sequence <code>alpha_n</code> (Robbins-Monro).
<code>rm.alpha</code>	scalar; noise added to gradient (Robbins-Monro).
<code>n.aux.draws</code>	count; number of auxiliary networks drawn from the ERGM likelihood (Robbins-Monro). See <a href="#">control.simulate.formula</a> .
<code>aux.thin</code>	count; number of auxiliary iterations between network draws after the first network is drawn (Robbins-Monro). See <a href="#">control.simulate.formula</a> .
<code>estimate</code>	If "MLE" (the default), then an approximate maximum likelihood estimator is used as a starting point in the Robbins-Monro algorithm. If "CD", the Monte-Carlo contrastive divergence estimate is returned. See <a href="#">ergm</a> .
<code>seed</code>	integer; seed for the random number generator. See <code>set.seed</code> .
<code>...</code>	Additional arguments, to be passed to the <code>ergm</code> function. See <a href="#">ergm</a> .

## References

Bouranis, L., Friel, N., & Maire, F. (2017). Efficient Bayesian inference for exponential random graph models by correcting the pseudo-posterior distribution. *Social Networks*, 50, 98-108. <https://arxiv.org/abs/1510.00934>

## Examples

```
## Not run:
# Load the florentine marriage network
data(florentine)

# Calibrated pseudo-posterior:
cpp.flo <- bergmC(flomarriage ~ edges + kstar(2),
  aux.iters = 500,
  burn.in = 500,
  main.iters = 10000,
  V.proposal = 2.5)

# Posterior summaries:
summary(cpp.flo)

## End(Not run)
```

---

 bergmM

---

*Parameter estimation for Bayesian ERGMs under missing data*


---

## Description

Function to fit Bayesian exponential random graphs models under missing data using the approximate exchange algorithm.

## Usage

```
bergmM(
  formula,
  burn.in = 100,
  main.iters = 1000,
  aux.iters = 1000,
  prior.mean = NULL,
  prior.sigma = NULL,
  nchains = NULL,
  gamma = 0.5,
  V.proposal = 0.0025,
  seed = NULL,
  startVals = NULL,
  offset.coef = NULL,
  nImp = NULL,
  missingUpdate = NULL,
  ...
)
```

## Arguments

formula	formula; an <code>ergm</code> formula object, of the form <code>&lt;network&gt; ~ &lt;model terms&gt;</code> where <code>&lt;network&gt;</code> is a <code>network</code> object and <code>&lt;model terms&gt;</code> are <code>ergm</code> -terms.
burn.in	count; number of burn-in iterations for every chain of the population.
main.iters	count; number of iterations for every chain of the population.
aux.iters	count; number of auxiliary iterations used for network simulation.
prior.mean	vector; mean vector of the multivariate Normal prior. By default set to a vector of 0's.
prior.sigma	square matrix; 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).
gamma	scalar; parallel adaptive direction sampling move factor.
V.proposal	count; diagonal entry for the multivariate Normal proposal. By default set to 0.0025.

seed	count; random number seed for the Bergm estimation.
startVals	vector; optional starting values for the parameter estimation.
offset.coef	vector; A vector of coefficients for the offset terms.
nImp	count; number of imputed networks to be returned. If null, no imputed network will be returned.
missingUpdate	count; number of tie updates in each imputation step. By default equal to number of missing ties. Smaller numbers increase speed. Larger numbers lead to better sampling.
...	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(1), 41-55. <https://arxiv.org/abs/1007.5192>
- Caimo, A. and Friel, N. (2014), "Bergm: Bayesian Exponential Random Graphs in R," *Journal of Statistical Software*, 61(2), 1-25. <https://www.jstatsoft.org/v61/i02>
- Koskinen, J.H., Robins, G.L., Pattison, P.E. (2010), "Analysing exponential random graph (p-star) models with missing data using Bayesian data augmentation," *Statistical Methodology* 7(3), 366-384.
- Krause, R.W., Huisman, M., Steglich, C., Snijders, T.A. (2020), "Missing data in cross-sectional networks-An extensive comparison of missing data treatment methods", *Social Networks* 62: 99-112.

## Examples

```
## Not run:
# Load the florentine marriage network
data(florentine)

# Create missing data
set.seed(14021994)
n <- dim(flomarriage[, ])[1]
missNode <- sample(1:n, 1)
flomarriage[missNode, ] <- NA
flomarriage[, missNode] <- NA

# Posterior parameter estimation:
m.flo <- bergmM(flomarriage ~ edges + kstar(2),
               burn.in = 50,
               aux.iters = 500,
               main.iters = 1000,
               gamma = 1.2,
               nImp = 5)

# Posterior summaries:
summary(m.flo)

## End(Not run)
```

---

 bgof

*Bayesian goodness-of-fit diagnostics for ERGMs*


---

## Description

Function to calculate 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,
  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>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.



## References

Caimo, A. and Friel, N. (2011), "Bayesian Inference for Exponential Random Graph Models," *Social Networks*, 33(1), 41-55. <https://arxiv.org/abs/1007.5192>

Caimo, A. and Friel, N. (2014), "Bergm: Bayesian Exponential Random Graphs in R," *Journal of Statistical Software*, 61(2), 1-25. <https://www.jstatsoft.org/v61/i02>

## Examples

```
## Not run:
# Load the florentine marriage network
data(florentine)

# Posterior parameter estimation:
p.flo <- bergm(flomarriage ~ edges + kstar(2),
              burn.in = 50,
              aux.iters = 500,
              main.iters = 1000,
              gamma = 1.2)

# Bayesian goodness-of-fit test:
bgof(p.flo,
      aux.iters = 500,
      sample.size = 30,
      n.deg = 10,
      n.dist = 9,
      n.esp = 6)

## End(Not run)
```

---

ergmAPL

*Adjustment of ERGM pseudolikelihood*

---

## Description

Function to estimate the transformation parameters for adjusting the pseudolikelihood function.

## Usage

```
ergmAPL(
  formula,
  aux.iters = NULL,
  n.aux.draws = NULL,
  aux.thin = NULL,
  ladder = NULL,
  estimate = c("MLE", "CD"),
  seed = 1,
  ...
)
```

**Arguments**

formula	formula; an <code>ergm</code> formula object, of the form <code>&lt;network&gt; ~ &lt;model terms&gt;</code> where <code>&lt;network&gt;</code> is a <code>network</code> object and <code>&lt;model terms&gt;</code> are <code>ergm</code> -terms.
aux.iter	count; number of auxiliary iterations used for drawing the first network from the ERGM likelihood. See <code>control.simulate.formula</code> .
n.aux.draws	count; Number of auxiliary networks drawn from the ERGM likelihood. See <code>control.simulate.formula</code> .
aux.thin	count; Number of auxiliary iterations between network draws after the first network is drawn. See <code>control.simulate.formula</code> .
ladder	count; Length of temperature ladder ( $\geq 3$ ).
estimate	If "MLE" (the default), then an approximate maximum likelihood estimator is returned. If "CD", the Monte-Carlo contrastive divergence estimate is returned. See <code>ergm</code> .
seed	integer; seed for the random number generator. See <code>set.seed</code> .
...	Additional arguments, to be passed to the <code>ergm</code> function. See <code>ergm</code> .

**References**

Bouranis, L., Friel, N., & Maire, F. (2018). Bayesian model selection for exponential random graph models via adjusted pseudolikelihoods. *Journal of Computational and Graphical Statistics*, 27(3), 516-528. <https://arxiv.org/abs/1706.06344>

---

evidence	<i>Wrapper function for evidence estimation</i>
----------	---

---

**Description**

Function to estimate the evidence (marginal likelihood) with Chib and Jeliazkov's method or Power posteriors, based on the adjusted pseudolikelihood function.

**Usage**

```
evidence(evidence.method = c("CJ", "PP"), ...)
```

**Arguments**

evidence.method	vector Method to estimate the marginal likelihood. Options are: "CJ", in which case the marginal likelihood is estimated with Chib and Jeliazkov's method; "PP", in which case the marginal likelihood is estimated with Power posteriors.
...	further arguments to be passed. See <code>evidenceCJ</code> and <code>evidencePP</code> .

## References

Bouranis, L., Friel, N., & Maire, F. (2018). Bayesian model selection for exponential random graph models via adjusted pseudolikelihoods. *Journal of Computational and Graphical Statistics*, 27(3), 516-528. <https://arxiv.org/abs/1706.06344>

## Examples

```
## Not run:
# Load the florentine marriage network:
data(florentine)

# MCMC sampling and evidence estimation:
CJE <- evidence(evidence.method = "CJ",
               formula      = flomarriage ~ edges + kstar(2),
               main.iters   = 30000,
               burn.in     = 2000,
               aux.iters    = 1000,
               num.samples  = 25000,
               V.proposal   = 2.5,
               ladder       = 100,
               seed         = 1)

# Posterior summaries:
summary(CJE)

# MCMC diagnostics plots:
plot(CJE)

# Log-evidence (marginal likelihood) estimate:
CJE$log.evidence

## End(Not run)
```

---

evidenceCJ

*Evidence estimation via Chib and Jeliazkov's method*

---

## Description

Function to estimate the evidence (marginal likelihood) with Chib and Jeliazkov's method, based on the adjusted pseudolikelihood function.

## Usage

```
evidenceCJ(
  formula,
  prior.mean = NULL,
  prior.sigma = NULL,
  aux.iters = 1000,
```

```

n.aux.draws = 5,
aux.thin = 50,
ladder = 30,
main.iters = 30000,
burn.in = 5000,
thin = 1,
V.proposal = 1.5,
num.samples = 25000,
seed = 1,
estimate = c("MLE", "CD"),
...
)

```

### Arguments

formula	formula; an <a href="#">ergm</a> formula object, of the form <network> ~ <model terms> where <network> is a <a href="#">network</a> object and <model terms> are <a href="#">ergm</a> -terms.
prior.mean	vector; mean vector of the multivariate Normal prior. By default set to a vector of 0's.
prior.sigma	square matrix; variance/covariance matrix for the multivariate Normal prior. By default set to a diagonal matrix with every diagonal entry equal to 100.
aux.iters	count; number of auxiliary iterations used for drawing the first network from the ERGM likelihood. See <a href="#">control.simulate.formula</a> and <a href="#">ergmAPL</a> .
n.aux.draws	count; number of auxiliary networks drawn from the ERGM likelihood. See <a href="#">control.simulate.formula</a> and <a href="#">ergmAPL</a> .
aux.thin	count; number of auxiliary iterations between network draws after the first network is drawn. See <a href="#">control.simulate.formula</a> and <a href="#">ergmAPL</a> .
ladder	count; length of temperature ladder ( $\geq 3$ ). See <a href="#">ergmAPL</a> .
main.iters	count; number of MCMC iterations after burn-in for the adjusted pseudo-posterior estimation.
burn.in	count; number of burn-in iterations at the beginning of an MCMC run for the adjusted pseudo-posterior estimation.
thin	count; thinning interval used in the simulation for the adjusted pseudo-posterior estimation. The number of MCMC iterations must be divisible by this value.
V.proposal	count; diagonal entry for the multivariate Normal proposal. By default set to 1.5.
num.samples	integer; number of samples used in the marginal likelihood estimate. Must be lower than <code>main.iters - burn.in</code> .
seed	integer; seed for the random number generator. See <code>set.seed</code> and <a href="#">MCMCmetrop1R</a> .
estimate	If "MLE" (the default), then an approximate maximum likelihood estimator is returned. If "CD", the Monte-Carlo contrastive divergence estimate is returned. See <a href="#">ergm</a> .
...	additional arguments, to be passed to the <a href="#">ergm</a> function. See <a href="#">ergm</a> and <a href="#">ergmAPL</a> .

## References

Caimo, A., & Friel, N. (2013). Bayesian model selection for exponential random graph models. *Social Networks*, 35(1), 11-24. <https://arxiv.org/abs/1201.2337>

Bouranis, L., Friel, N., & Maire, F. (2018). Bayesian model selection for exponential random graph models via adjusted pseudolikelihoods. *Journal of Computational and Graphical Statistics*, 27(3), 516-528. <https://arxiv.org/abs/1706.06344>

## Examples

```
## Not run:
# Load the florentine marriage network:
data(florentine)

# MCMC sampling and evidence estimation:
CJE <- evidenceCJ(flomarriage ~ edges + kstar(2),
                 main.iters = 2000,
                 burn.in   = 200,
                 aux.iters  = 500,
                 num.samples = 25000,
                 V.proposal = 2.5)

# Posterior summaries:
summary(CJE)

# MCMC diagnostics plots:
plot(CJE)

# Log-evidence (marginal likelihood) estimate:
CJE$log.evidence

## End(Not run)
```

---

evidencePP

*Evidence estimation via power posteriors*

---

## Description

Function to estimate the evidence (marginal likelihood) with Power posteriors, based on the adjusted pseudolikelihood function.

## Usage

```
evidencePP(
  formula,
  prior.mean = NULL,
  prior.sigma = NULL,
  aux.iters = 1000,
```

```

n.aux.draws = 50,
aux.thin = 50,
ladder = 30,
main.iters = 20000,
burn.in = 5000,
thin = 1,
V.proposal = 1.5,
seed = 1,
temps = NULL,
estimate = c("MLE", "CD"),
...
)

```

### Arguments

formula	formula; an <a href="#">ergm</a> formula object, of the form <network> ~ <model terms> where <network> is a <a href="#">network</a> object and <model terms> are ergm-terms.
prior.mean	vector; mean vector of the multivariate Normal prior. By default set to a vector of 0's.
prior.sigma	square matrix; variance/covariance matrix for the multivariate Normal prior. By default set to a diagonal matrix with every diagonal entry equal to 100.
aux.iters	count; number of auxiliary iterations used for drawing the first network from the ERGM likelihood. See <a href="#">control.simulate.formula</a> and <a href="#">ergmAPL</a> .
n.aux.draws	count; number of auxiliary networks drawn from the ERGM likelihood. See <a href="#">control.simulate.formula</a> and <a href="#">ergmAPL</a> .
aux.thin	count; number of auxiliary iterations between network draws after the first network is drawn. See <a href="#">control.simulate.formula</a> and <a href="#">ergmAPL</a> .
ladder	count; length of temperature ladder ( $\geq 3$ ). See <a href="#">ergmAPL</a> .
main.iters	count; number of MCMC iterations after burn-in for the adjusted pseudo-posterior estimation.
burn.in	count; number of burn-in iterations at the beginning of an MCMC run for the adjusted pseudo-posterior estimation.
thin	count; thinning interval used in the simulation for the adjusted pseudo-posterior estimation. The number of MCMC iterations must be divisible by this value.
V.proposal	count; diagonal entry for the multivariate Normal proposal. By default set to 1.5.
seed	integer; seed for the random number generator. See <a href="#">set.seed</a> and <a href="#">MCMCmetrop1R</a> .
temps	numeric vector; inverse temperature ladder, $t \in [0, 1]$ .
estimate	If "MLE" (the default), then an approximate maximum likelihood estimator is returned. If "CD", the Monte-Carlo contrastive divergence estimate is returned. See <a href="#">ergm</a> .
...	additional arguments, to be passed to the <a href="#">ergm</a> function. See <a href="#">ergm</a> and <a href="#">ergmAPL</a> .

## References

Bouranis, L., Friel, N., & Maire, F. (2018). Bayesian model selection for exponential random graph models via adjusted pseudolikelihoods. *Journal of Computational and Graphical Statistics*, 27(3), 516-528. <https://arxiv.org/abs/1706.06344>

## Examples

```
## Not run:
# Load the florentine marriage network:
data(florentine)

PPE <- evidencePP(flomarriage ~ edges + kstar(2),
                 aux.iters = 500,
                 noisy.nsim = 50,
                 aux.thin = 50,
                 main.iters = 2000,
                 burn.in = 100,
                 V.proposal = 2.5)

# Posterior summaries:
summary(PPE)

# MCMC diagnostics plots:
plot(PPE)

# Log-evidence (marginal likelihood) estimate:
PPE$log.evidence

## End(Not run)
```

---

lazega

*Lazega lawyers network data*

---

## Description

Lazega lawyers network data

## Usage

```
lazega
```

## Format

An object of class network.

**Source**

This network dataset comes from a network study of corporate law partnership that was carried out in a Northeastern US corporate law firm in New England from 1988 to 1991. It represents collaborative relations among the 36 attorneys (partners and associates) of this firm. Nodal attributes include: Age, Gender, Office, Practice, School, and Years.

**References**

Lazega, E. (2001), "The Collegial Phenomenon: The Social Mechanisms of Cooperation Among Peers in a Corporate Law Partnership," Oxford University Press.

**Examples**

```
## Not run:
par(mfrow = c(1, 2), oma = rep(0, 4))
CC <- hcl.colors(3, "Teal")
set.seed(22)
plot(lazega,
     vertex.col = CC[lazega %v% "Office"],
     vertex.cex = 2)
legend("topright",
     pch = 21,
     pt.bg = CC,
     legend = c("Boston", "Hartford", "Providence"),
     title = "OFFICE")

## End(Not run)
```

---

plot.bergm

*Plot BERGM posterior output*


---

**Description**

This function creates MCMC diagnostic plots for `bergm` objects.

**Usage**

```
## S3 method for class 'bergm'
plot(x, ...)
```

**Arguments**

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



**Examples**

```
## Not run:
# Load the florentine marriage network
data(florentine)

# Posterior parameter estimation:
p.flo <- bergm(flomarriage ~ edges + kstar(2),
              burn.in   = 50,
              aux.iters  = 500,
              main.iters = 1000,
              gamma      = 1.2)

# MCMC diagnostics plots:
plot(p.flo)

## End(Not run)
```

---

summary.bergm

*Summary of BERGM posterior output*

---

**Description**

This function summarises MCMC output for bergm objects.

**Usage**

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

**Arguments**

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

# Index

## \* datasets

lazega, [15](#)

Bergm (Bergm-package), [2](#)

bergm, [2](#)

Bergm-package, [2](#)

bergmC, [4](#)

bergmM, [6](#)

bgof, [8](#)

control.simulate.formula, [4](#), [5](#), [10](#), [12](#), [14](#)

ergm, [3–6](#), [10](#), [12](#), [14](#)

ergmAPL, [9](#), [12](#), [14](#)

evidence, [10](#)

evidenceCJ, [11](#)

evidencePP, [13](#)

lazega, [15](#)

MCMCmetrop1R, [12](#), [14](#)

network, [3](#), [4](#), [6](#), [10](#), [12](#), [14](#)

plot.bergm, [16](#)

summary.bergm, [17](#)