Package ‘Bergm’

March 14, 2017

Type Package
Title Bayesian Exponential Random Graph Models
Version 4.0.0
Date 2017-03-14
Author Alberto Caimo [aut, cre],
      Lampros Bouranis [aut],
      Florian Maire [ctb],
      Nial Friel [ctb]
Maintainer Alberto Caimo <acaimo.stats@gmail.com>
Description Set of tools to analyse Bayesian exponential random graph models.
License GPL (>= 2)
URL http://acaimo.github.io/Bergm/
Depends ergm, network, coda, mvtnorm, MCMCpack
LazyLoad yes
RoxygenNote 6.0.1
NeedsCompilation no
Repository CRAN
Date/Publication 2017-03-14 14:45:00

R topics documented:

bergm ....................................................... 2
bergm.output ........................................... 3
bgof ....................................................... 4
calibrate.bergm ........................................... 5

Index 7
bergm

Bayesian parameter inference for ERGMs

Description

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

Usage

```
bergm(formula, burn.in = 100, main.iters = 1000, aux.iters = 1000,
      prior.mean = NULL, sigma.mean = NULL, nchains = NULL, gamma = 0.5,
      sigma.epsilon = NULL, ...)  
```

Arguments

- **formula**: an R formula object, of the form `<network> ~ <model terms>` where `<network>` is a network object and `<model terms>` are ergm-terms.
- **burn.in**: count; number of burn-in iterations at the beginning of an MCMC run. If `nchains > 2`, 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 `nchains > 2`, it refers to the 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.
- **sigma.mean**: 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). If the model is one-dimensional, `nchains = 1`.
- **gamma**: scalar; parallel ADS move factor. If the model is one-dimensional, `nchains = 1` and `gamma = sigma.epsilon` and is used as the variance of the Normal proposal distribution.
- **sigma.epsilon**: square matrix; variance/covariance matrix for the multivariate Normal proposal when `nchains > 2`. By default set to a diagonal matrix with every diagonal entry equal to 0.0025. If the model is one-dimensional, `sigma.epsilon = gamma` and is used as the variance of the Normal proposal distribution.
- **...**: additional arguments, to be passed to lower-level functions.

References

Examples

# Load the florentine marriage network
data(florentine)

# Posterior parameter estimation:

p.flo <- bergm(formula = flomarriage ~ edges + kstar(2),
               burn.in = 50,
               aux.iters = 500,
               main.iters = 500,
               gamma = 1)

bergm.output

bergm.output  Summarising posterior BERGM output

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 or calibrate.bergm
...
additional arguments, to be passed to lower-level functions.

Examples

# 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 = 500,
               gamma = 1)

# MCMC diagnostics and posterior summaries:

bergm.output(p.flo)
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, directed = FALSE, sample.size = 100, aux.iters = 10000,
    n.deg = NULL, n.dist = NULL, n.esp = NULL, n.ideg = NULL,
    n.odeg = NULL, ...)

Arguments

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

References


See Also

bergm, calibrate.bergm.
Examples

# 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 = 500,
               gamma = 1)

# Bayesian goodness-of-fit test:

bgof(p.flo,
     aux.iters = 500,
     sample.size = 50,
     n.deg = 10,
     n.dist = 9,
     n.esp = 6)

---

**calibrate.bergm**

*Calibrating misspecified ERGMs for Bayesian parameter inference*

Description

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

Usage

```r
calibrate.bergm(ergm.formula, iters = 500, a = 0.001, alpha = 0,
                aux.iters = 5000, noisy.nsim = 400, noisy.thin = 50,
                prior.mean = NULL, prior.sigma = NULL, thin = 1, mcmc = 40000,
                burnin = 10000, tunePL = 1)
```

Arguments

- **ergm.formula**: formula; an R formula object, of the form `<network> ~ <model terms>` where `<network>` is a `network` object and `<model terms>` are `ergm-terms`.
- **iters**: count; Iterations for the Robbins-Monro stochastic approximation algorithm.
- **a**: scalar; Constant for sequence alpha_n (Robbins-Monro).
- **alpha**: scalar; Noise added to gradient (Robbins-Monro).
- **aux.iters**: count; Number of proposals before any MCMC sampling is done (Robbins-Monro). See `control.simulate.formula`. 
noisy.nsim count; Number of TNT draws (Robbins-Monro). See control.simulate.formula.

noisy.thin count; Number of proposals between sampled statistics (Robbins-Monro). See control.simulate.formula.

prior.mean vector; Prior means.

prior.sigma matrix; Prior covariance matrix.

thin count; Thinning interval used in the simulation for the pseudo-posterior estimation. The number of MCMC iterations must be divisible by this value.

mcmc count; Number of MCMC iterations after burn-in for the pseudo-posterior estimation.

burnin count; Number of burn-in iterations at the beginning of an MCMC run for the pseudo-posterior estimation.

tunePL count; Tuning parameter for the Metropolis sampling for the pseudo-posterior estimation.

References


Examples

# Load the florentine marriage network
data(florentine)

# Calibrated pseudo-posterior:
cpp.flo <- calibrate.bergm(flomarriage ~ edges + kstar(2),
 aux.iters = 3000,
 mcmc = 10000,
 burnin = 500,
 tunePL = 2)

# MCMC diagnostics and posterior summaries:
bergm.output(cpp.flo)

# Bayesian goodness-of-fit test:
bgof(cpp.flo,
aux.iters = 500,
sample.size = 50,
n.deg = 10,
n.dist = 9,
n.esp = 6)
Index

bergm, 2, 4
bergm.output, 3
bgof, 4

calibrate.bergm, 4, 5
control.simulate.formula, 5, 6

network, 2, 5