Package ‘Bergm’

November 13, 2017

Type Package
Title Bayesian Exponential Random Graph Models
Version 4.1.0
Date 2017-11-13
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)
Depends ergm
Imports network, coda, MCMCpack, Matrix, mvtnorm
URL http://acaimo.github.io/Bergm/
RoxygenNote 6.0.1
NeedsCompilation no
Repository CRAN
Date/Publication 2017-11-13 15:40:29 UTC

R topics documented:

Bergm-package ......................................................... 2
adjustPL ......................................................... 2
bergm .......................................................... 3
bergm.output ...................................................... 4
bgof ............................................................ 5
calibrate.bergm ..................................................... 6
evidence_CJ ......................................................... 8
evidence_powerP ...................................................... 9

Index 12

1
Bergm-package

Bayesian exponential random graph models

Description

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

adjustPL

Adjustment of pseudolikelihood function

Description

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

Usage

adjustPL(ergm.formula, aux.iters = 3000, noisy.nsims = 50, noisy.thin = 50, ladder = 50, ...)

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.
- `aux.iters` count; Number of proposals before any MCMC sampling is done. See control.simulate.formula.
- `noisy.nsims` count; Number of TNT draws. See control.simulate.formula.
- `noisy.thin` count; Number of proposals between sampled statistics. See control.simulate.formula.
- `ladder` count; Length of temperature ladder (>=3).
- `...` Additional arguments, to be passed to the ergm function. See ergm.

References

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: 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


---

**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**

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

**Arguments**

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

**Examples**

```r
# 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)
```
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

```r
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`, `pseudo.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.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.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.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.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.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


**Examples**

```r
# Load the florentine marriage network
data(florentine)

# Posterior parameter estimation:

p.flo <- bergm(fomarriage ~ 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

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

# Calibrated pseudo-posterior:

cpp.flo <- calibrate.bergm(fomarriage ~ 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)

## End(Not run)
```
evidence_CJ

Marginal likelihood estimation

Description

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

Usage

evidence_CJ(ergm.formula, prior.mean, prior.sigma, nits, burnin, thin = 1, num.samples = 5000, tunePL = 2, seed = NA, calibr.info = NULL)

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.
- `prior.mean`: vector; Prior means.
- `prior.sigma`: matrix; Prior covariance matrix.
- `nits`: count; Number of MCMC iterations after burn-in for the adjusted pseudo-posterior estimation.
- `burnin`: 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.
- `num.samples`: integer; number of samples used in the marginal likelihood estimate. Must be <=(nits-burnin).
- `tunePL`: count; Tuning parameter for the Metropolis sampling for the pseudo-posterior estimation.
- `seed`: The seed for the random number generator. See `MCMCmetrop1R`.
- `calibr.info`: list; Transformation parameters for adjusting the pseudolikelihood function. See `adjustPL`.

References


Examples

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

flo.formula <- flomarriage ~ edges + kstar(2)
```
Function to estimate the marginal likelihood with Power posteriors, based on the adjusted pseudo-likelihood function.
evidence_powerP

Usage

evidence_powerP(ergm.formula, prior.mean, prior.sigma, nits, burnin, thin = 1, tunePL = 2, seed = 1, temps = seq(0, 1, length.out = 50)^5, calibr.info)

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.
prior.mean vector; Prior means.
prior.sigma matrix; Prior covariance matrix.
nits count; Number of MCMC iterations after burn-in for the adjusted pseudo-posterior estimation.
burnin 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.
tunePL count; Tuning parameter for the Metropolis sampling for the pseudo-posterior estimation.
seed The seed for the random number generator. See MCMCmetrop1r.
temps numeric vector; Inverse temperature ladder, \( t \in [0, 1] \).
calibr.info list; Transformation parameters for adjusting the pseudolikelihood function. See adjustPL.

References


Examples

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

flo.formula <- flomarriage ~ edges + kstar(2)

info.adjustPL <- adjustPL(ergm.formula = flo.formula, 
aux.iters = 100, 
noisy.nsim = 50, 
noisy.thin = 50, 
ladder = 30, 
estimate = "MLE", 
control = control.ergm(MCMC.samplesize=2000))

# Add the output into a list:
calibration.list <- list(Theta_MLE= info.adjustPL$Theta_MLE, 
Theta_PL = info.adjustPL$theta_pl)
```
evidence_powerP

\[
W = \text{info.adjustPL}\$W, \\
C = \text{info.adjustPL}\$C
\]

\# Specify location and shape of prior distribution:
mean.priors <- rep(0,2)
sigma <- 5
sigma.priors <- diag(sigma,2)

\# MCMC sampling and evidence estimation:

tempvec.powerp <- seq(0,1,length.out=20)^5

pp.est.evidence <- evidence_powerP(ermg.formula = flo.formula,
prior.mean = mean.priors,
prior.sigma = sigma.priors,
nits = 10000,
burnin = 2000,
thin = 1,
tunePL = 2,
seed = 1,
temps = tempvec.powerp,
calibr.info = calibration.list)

\# MCMC diagnostics and posterior summaries:
bergm.output(pp.est.evidence)

\# Log-marginal likelihood estimate:
flo.model.logevidence <- pp.est.evidence\$log.evidence

## End(Not run)
Index

adjustPL, 2, 8, 10

Bergm (Bergm-package), 2
bergm, 3
Bergm-package, 2
bergm.output, 4
bgof, 5

calibrate.bergm, 6
control.simulate.formula, 2, 6, 7

ergm, 2
evidence_CJ, 8
evidence_powerP, 9

MCMCmetrop1R, 8, 10

network, 2, 3, 6, 8, 10