Package ‘Bergm’

Type Package
Title Bayesian Exponential Random Graph Models
Version 4.2.0
Date 2018-09-25
Author Alberto Caimo [aut, cre],
Lampros Bouranis [aut],
Robert Krause [aut]
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 6.1.0
NeedsCompilation no
Repository CRAN
Date/Publication 2018-09-25 15:50:03 UTC

R topics documented:

Bergm-package ................................................. 2
adjustPL .................................................... 2
bergm ..................................................... 3
bergm.output ............................................. 4
bgof ....................................................... 5
calibrate.bergm .......................................... 6
evidence_CJ ............................................. 7
evidence_PP ............................................. 9
missBergm ............................................... 10

Index 13
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(formula, aux.iters = 3000, noisy.nsim = 50, noisy.thin = 50, ladder = 50, ...)

Arguments

- formula: formula; an ergm 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.nsim: 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

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, prior.sigma = NULL, nchains = NULL, 
       gamma = 0.5, sigma.epsilon = NULL, ...)

Arguments

formula formula; an ergm 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.
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). 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

Statistical Software, 61(2), 1-25. jstatsoft.org/v61/i02
Examples

# 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 = 500,
              gamma = 1)

# Posterior summaries:
bergm.output(p.flo)

# Bayesian goodness-of-fit test:

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

---

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, pseudo.bergm, or calibrate.bergm.

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

```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` 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


calibrate.bergm

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

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

Arguments

- `formula`: a formula; an `ergm` formula object, of the form `<network> ~ <model terms>` where `<network>` is a `network` object and `<model terms>` are `ergm`-terms.
- `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.nsims`: 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 = 2.5)

# 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**

*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**

```r
evidence_cj(formula, prior.mean, prior.sigma, nits, burnin, thin = 1,
num.samples = 5000, tunePL = 2, seed = NA, info.adjustPL = NULL)
```

**Arguments**

- `formula`: formula; an `ergm` 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.
evidence_CJ

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.

info.adjustPL Transformation parameters for adjusting the pseudolikelihood function adjustPL.

References


Examples

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

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

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

# Specify a prior distribution:
mean.priors <- rep(0, 2)
sigma.priors <- diag(5, 2)

# MCMC sampling and evidence estimation:
Chib.est.evidence <- evidence_CJ(formula = flo.formula,
                                  prior.mean = mean.priors,
                                  prior.sigma = sigma.priors,
                                  nits = 30000,
                                  burnin = 5000,
                                  thin = 1,
                                  num.samples = 25000,
)```
function to estimate the evidence (marginal likelihood) with Power posteriors, based on the adjusted pseudolikelihood function.

Usage

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

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.
- **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, \( t \in [0, 1] \).
- **info.adjustPL**: Transformation parameters for adjusting the pseudolikelihood function `adjPL`.

Description

The function `evidence_PP` is used to estimate the evidence (marginal likelihood) with Power posteriors, based on the adjusted pseudolikelihood function. It takes the formula, prior means, prior covariance matrix, number of MCMC iterations, burn-in iterations, thinning interval, tuning parameter for the Metropolis sampling, and the transformation parameters for adjusting the pseudolikelihood function as arguments.

Usage

- **formula**: An R formula object, specifying the network and model terms.
- **prior.mean**: A vector specifying the prior means.
- **prior.sigma**: A matrix specifying the prior covariance matrix.
- **nits**: An integer specifying the number of MCMC iterations after burn-in.
- **burnin**: An integer specifying the number of burn-in iterations.
- **thin**: An integer specifying the thinning interval.
- **tunePL**: An integer specifying the tuning parameter for the Metropolis sampling.
- **seed**: An integer specifying the seed for the random number generator.
- **temps**: A numeric vector specifying the inverse temperature ladder.
- **info.adjustPL**: A list specifying the transformation parameters for adjusting the pseudolikelihood function.

Example

```r
# MCMC diagnostics and posterior summaries:
bergm.output(Chib.est.evidence)

# Log-marginal likelihood estimate:
Chib.est.evidence$log.evidence

## End(Not run)
```
References


Examples

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

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

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

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

pp.est.evidence <- evidence_PP(formula = flo.formula,
                                prior.mean = prior.mean,
                                prior.sigma = prior.sigma,
                                nits = 10000,
                                burnin = 2000,
                                temps = seq(0, 1, length.out = 20)^5,
                                info.adjustPL = info.adjustPL)

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

# Log-marginal likelihood estimate:
pp.est.evidence$log.evidence

## End(Not run)
```

---

**missBergm**  
*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

missBergm(formula, burn.in = 100, main.iters = 1000, aux.iters = 1000, prior.mean = NULL, prior.sigma = NULL, nchains = NULL, gamma = 0.5, sigma.epsilon = NULL, seed = NULL, startVals = NULL, nImp = NULL, missingUpdate = NULL, ...)

Arguments

formula  formula; an ergm 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.
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). 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.
seed  count; random number seed for the Bergm estimation.
startVals  numeric matrix; Starting values for the parameter estimation. startVals requires a matrix with parameters by number of chains. If nchains == NULL, nchains is equal to 2 * the number of parameters in the model.
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


Examples

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

data(florentine)

# Create missing data
set.seed(22101992)

missNode <- sample(1:16,1)
flomarriage[missNode,] <- NA
flomarriage[,missNode] <- NA

# Posterior parameter estimation:
m.flo <- missBergm(flomarriage ~ edges + kstar(2),
    burn.in = 50,
    aux.iters = 500,
    main.iters = 500,
    gamma = 1,
    nImp = 5)

# Posterior summaries:
bergm.output(m.flo)

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

## End(Not run)
```
Index

adjustPL, 2, 8, 9

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

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

ergm, 2, 3, 6, 7, 11
evidence_CJ, 7
evidence_PP, 9

MCMCmetrop1R, 8, 9
missBergm, 10

network, 2, 3, 6, 7, 9, 11