Package ‘Bergm’

February 19, 2015

Type Package
Title Bayesian analysis for exponential random graph models
Version 3.0.1
Date 2014-11-23
Author Alberto Caimo [aut, cre], 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://tiny.cc/Bergm-info
Depends ergm, network, coda, mvtnorm
LazyLoad yes
NeedsCompilation no
Repository CRAN
Date/Publication 2014-11-24 12:11:13

R topics documented:

Bergm-package ............................................................. 2
abergm ................................................................. 2
bbeta ................................................................. 4
bbeta.output .......................................................... 5
bergm ................................................................. 5
bergm.output ......................................................... 7
bergmS ............................................................... 7
bergmS.output ...................................................... 9
bgof ................................................................. 10

Index 11
**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 `<acaimo.stats@gmail.com>`, Nial Friel `<nial.friel@ucd.ie>`.

**References**


---

**abergm**

*Adaptive algorithms for Bayesian exponential random graph models*

**Description**

Function to fit Bayesian exponential random graphs models using adaptive exchange algorithms.

**Usage**

```r
abergm(formula, 
   burn.in = 100, 
   main.iters = 1000, 
   aux.iters = 1000, 
   m.prior = NULL, 
   sigma.prior = NULL, 
   nchains = NULL, 
   gamma = 0.5, 
   method = 'Adaptive.chains', 
   rectangular = TRUE, 
   sigma.epsilon = NULL, 
   updategap = 10, 
   ...)
```
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 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.

method  name of the adaptive strategy: ADS = adaptive direction sampling, Adaptive.past = adaptive strategy where past parameter particles are used, Adaptive.chains (default) = adaptive strategy all particles at the current time for all chains are used.

rectangular  logical; if Adaptive.past is used, it defines the type of adaptive strategy: if TRUE (default) = all parameter particles from all chains and all past simulations are used, if FALSE = all parameter particles along the same chain and all are used.

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.

updategap  scalar; iteration interval used to update the variance/covariance matrix.

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

See Also

bergm

Examples

data(molecule)

mol <- abergm(molecule ~ edges + kstar(R),
  burn.in = 50,
  aux.iters = 50,
bbeta

Bayesian beta model

Description

Function to fit the Bayesian beta model using an MCMC algorithm.

Usage

bbeta(formula, 
    burn.in = 100, 
    main.iters = 1000, 
    m.prior = NULL, 
    sigma.prior = NULL, 
    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.

main.iters count; number of iterations for the MCMC chain excluding burn-in.

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.

sigma.epsilon variance/covariance matrix for the multivariate Normal proposal. By default set to a diagonal matrix with every diagonal entry equal to 0.0025.

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

Examples

# Make sure the observed network does not have isolated nodes 
set.seed(27)

y <- network(7, directed = FALSE) 
post.est <- bbeta(y ~ sociality(base = 0), 
    main.iters = 3000, 
    sigma.epsilon = diag(1.3, dim(y[,1])[[1]]))

bbeta.output(post.est, plot = TRUE)
bbeta.output

MCMC output diagnostics for the Bayesian beta model estimation procedure

Description

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

Usage

bbeta.output(x, plot = FALSE, ...)

Arguments

x an R object of class bbeta.
plot logical; if TRUE diagnostic plots for each parameter are plotted.
... additional arguments, to be passed to lower-level functions.

See Also

bbeta.

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=10, main.iters=1000, aux.iters=1000, m.prior = NULL, sigma.prior = 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 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.
- **...**: additional arguments, to be passed to lower-level functions.

See Also

- `bergm.output`, `bgof`.

Examples

```r
data(florentine)

# Parameter estimation via approximate exchange algorithm with ADS:

flo <- bergm(flomarriage ~ edges + kstar(2),
             burn.in=10,
             aux.iters=500,
             main.iters=500,
             gamma=1)

# MCMC diagnostics:

bergm.output(flo)

# Bayesian goodness-of-fit test:
```

bergm.output

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

---

**bergm.output**

*MCMC output diagnostics for the Bayesian parameter estimation algorithm*

---

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

**See Also**

- `bergm`

---

**bergmS**

*Bayesian model selection for exponential random graph models*

---

**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 = NULL, 
  main.iters = NULL, 
  burn.ins=NULL, 
  ...) 

Arguments

  formulae  list; R formula objects (competing models) of the form <network> ~ <model terms>, see ergm-terms.

  iters     count; number of iterations for auto-RJ exchange algorithm (online step).

  m.priors  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.

  sigma.priors  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.

  gammas  vector; “parallel ADS move factors” for each competing model (offline step). By default set to a vector of 0.5's.

  nchains  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. epsilons  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.

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

See Also

bergm.
Examples

data(florentine)
y <- flomarriage

# Competing models:
formulae <- c(y ~ edges,
              y ~ edges + kstar(2))

# Model selection via auto-RJ approximate exchange algorithm
flo <- bergm(formulae, 
              iters=1500,
              aux.iters=500,
              main.iters=rep(100,2),
              burn.in=rep(50,2),
              gammas=c(1,1))

# MCMC diagnostics
flo.out <- bergm.output(flo)

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.
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
```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`.
- `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.

See Also
- `bergm`
Index

abergm, 2

bbeta, 4, 5
bbeta.output, 5
bergm, 3, 5, 7, 8, 10
Bergm-package, 2
bergm.output, 6, 7
bergmS, 7, 9
bergmS.output, 9
bgof, 6, 10

network, 3, 4, 6