Package 'Bergm'

October 27, 2026

Type Package

Title Bayesian Exponential Random Graph Models

Version 6.0.0

Date 2026-10-18

Description Bayesian analysis for exponential random graph models using advanced computational algorithms: the approximate exchange algorithm with adaptive direction sampling (optionally adaptive, delayed-acceptance and parallel), calibrated pseudo-posteriors, Bayesian data augmentation for missing network data, and Bayesian goodness-of-fit diagnostics. More information can be found at: <https://acaimo.github.io/Bergm/>.

License GPL (>= 2)

URL https://acaimo.github.io/Bergm/

Depends ergm (>= 4.2.3), R (>= 4.1)

Imports coda, graphics, grDevices, mice, network, parallel, Rcpp, stats, utils

LinkingTo Rcpp

Suggests spelling, testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.3.1

NeedsCompilation yes

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

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

Repository CRAN

Date/Publication 2026-10-27 15:00:18 UTC

Contents

bergm 2
bergm-methods 6
bergmC 6
bergmM 9
bgof 12
ergmAPL 14
evidence 16
evidenceCJ 17
evidencePP 19
lazega 22
plot.bergm 23
plot.bgof 23
simulate.bergm 24
summary.bergm 25

Index **26**

`bergm` *Parameter estimation for Bayesian ERGMs*

Description

Fits Bayesian exponential random graph models with the approximate exchange algorithm and a population of interacting chains updated by adaptive direction sampling (ADS) (Caimo and Friel, 2011).

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,
  constraints = NULL,
  thin = 1,
```

```
  cut.reject = FALSE,
  saveEveryX = NULL,
  saveEveryXName = "partialBergmEstimate.rda",
  seed = NULL,
  adapt = TRUE,
  target.accept = 0.25,
  delayed.acceptance = TRUE,
  da.inflate = 1.5,
  cores = 1,
  log.prior = NULL,
  prior.decay = c(meanlog = log(0.5), sdlog = 0.75),
  verbose = TRUE,
  ...
)
```

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. Curved terms (e.g. gwesp(fixed = FALSE)) are supported.
prior.mean	vector; mean vector of the multivariate Normal prior. By default set to a vector of 0's.
prior.sigma	square matrix, vector or scalar; variance/covariance matrix for the multivariate Normal prior. A scalar or vector is interpreted as the diagonal. By default 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 twice the number of free model parameters (at least 4), rounded up to a multiple of 2 * cores when cores > 1.
gamma	scalar; parallel adaptive direction sampling move factor. When adapt = TRUE this is the starting value, tuned during burn-in.
V.proposal	scalar, vector or matrix; covariance of the small Normal perturbation added to every ADS move. A scalar is the diagonal entry (default 0.0025, as in previous versions).
startVals	vector; optional starting values for the parameter estimation. Alternatively a previous bergm fit of the same model, in which case every chain continues from where that run ended (and its adapted gamma is reused unless gamma is given), so a run can be extended in stages.
offset.coef	vector; a vector of coefficients for the offset terms.
constraints	formula; constraints on the support of the network distribution, as in ergm.
thin	count; thinning interval applied to every chain.
cut.reject	logical; if TRUE, repeated (rejected) draws are removed from the pooled sample Theta. This changes the posterior summaries and is only kept for backward compatibility; default FALSE.

`saveEveryX` count; if not `NULL`, the partial posterior sample is saved to `saveEveryXName` every `saveEveryX` iterations.

`saveEveryXName` character; file name of the partial estimation object.

`seed` integer; optional seed for reproducibility (also seeds the parallel workers when `cores` > 1).

`adapt` logical; tune `gamma` during burn-in towards `target.accept`. Adaptation stops at the end of burn-in.

`target.accept` scalar; target acceptance rate for `adapt`.

`delayed.acceptance`
logical; if `TRUE` (default), proposals are first screened with a Gaussian surrogate of the posterior fitted to the second half of the burn-in population, and an auxiliary network is simulated only for proposals that pass. The second stage corrects for the surrogate so the target distribution is unchanged (Christen and Fox, 2005). It needs a burn-in of at least 10 iterations; with a shorter burn-in it is switched off (silently when left at its default). `FALSE` gives the Bergm 5 kernel.

`da.inflate` scalar; variance inflation factor for the delayed-acceptance surrogate (values > 1 make the screen more permissive).

`cores` count; number of parallel workers. With `cores` > 1 the population is updated in two halves (each half conditionally on the other) and the auxiliary simulations of a half run in parallel. Worthwhile when a single auxiliary simulation takes more than a few milliseconds.

`log.prior` function; optional user-defined log prior density taking the full (named) parameter vector, with decay parameters on their natural (positive) scale, and returning a scalar. Overrides the Normal and log-normal priors; return `-Inf` outside the support.

`prior.decay` named vector or list with elements `meanlog` and `sdlog`; log-normal prior for the decay parameters of curved terms (e.g. `gwesp.decay`, `gwdegree.decay`, `altkstar.lambda`), recycled over all of them or given one value per decay parameter. Default: median 0.5, `sdlog` 0.75. Decay parameters are sampled on the log scale, so they are always positive; `prior.mean` and `prior.sigma` apply to the other parameters and may be given either for all parameters (decay entries are ignored) or for the non-decay ones only.

`verbose` logical; print progress information.

`...` additional arguments passed to `control.simulate.formula` to configure the network proposal (e.g. `MCMC.prop`, `MCMC.prop.weights`).

Details

Compared with Bergm 5, the sampler draws every auxiliary network through a single call to **ergm**'s compiled sampler on a pre-built network state, supports curved ERGM terms, models with a single parameter, custom priors, burn-in adaptation of the ADS move factor, delayed acceptance and parallel chains. The posterior sample is returned both pooled (`Theta`, as in previous versions) and per chain (`chains`), which enables standard convergence diagnostics such as the potential scale reduction factor.

Value

An object of class `bergm` with elements

`Theta`	pooled posterior sample (`mcmc`), chains stacked;
`chains`	the per-chain posterior samples (`mcmc.list`);
`AR`, `AR.chains`	overall and per-chain acceptance rates;
`ess`, `Rhat`	effective sample sizes and potential scale reduction factors;
`fixed`	logical vector flagging fixed (offset) parameters;
`gamma`	the (adapted) ADS move factor;
`n.sims`	number of auxiliary network simulations performed;

plus the model specification and run settings.

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. doi:10.18637/jss.v061.i02

Christen, J. A. and Fox, C. (2005), "Markov chain Monte Carlo using an approximation," Journal of Computational and Graphical Statistics, 14(4), 795-810.

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 = 3000,
               gamma      = 1.2,
               seed       = 1)

# Posterior summaries:
summary(p.flo)

# Curved model, delayed acceptance and 4 parallel workers:
p.laz <- bergm(lazega ~ edges + nodematch("Office") + gwesp(0.5, fixed = FALSE),
               burn.in = 200, main.iters = 2000, aux.iters = 3000,
               delayed.acceptance = TRUE, cores = 4, seed = 1)

## End(Not run)
```

bergm-methods *Posterior point estimates and covariance for* bergm *objects*

Description

Posterior point estimates and covariance for bergm objects

Usage

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

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

## S3 method for class 'bergm'
as.mcmc(x, ...)

## S3 method for class 'bergm'
as.mcmc.list(x, ...)
```

Arguments

object	an object of class bergm.
...	unused.
x	an object of class bergm.

Value

coef: posterior means; vcov: posterior covariance matrix.

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 (Bouranis, Friel and Maire, 2017).

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,
  map.method = c("newton", "rm"),
  map.iters = 20,
  init = NULL,
  verbose = TRUE,
  ...
)
```

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; mean vector of the multivariate Normal prior. By default set to a vector of 0's.
prior.sigma	square matrix (or scalar/vector diagonal); 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 at the beginning of an MCMC run for the pseudo-posterior estimation.
main.iters	count; number of MCMC iterations after burn-in for the pseudo-posterior estimation.
aux.iters	count; number of auxiliary iterations used for drawing the first network from the ERGM likelihood.
V.proposal	count; diagonal entry for the multivariate Normal proposal. By default set to 1.5.
thin	count; thinning interval used in the simulation for the pseudo-posterior estimation.
rm.iters	count; number of iterations for the Robbins-Monro algorithm (map.method = "rm").
rm.a	scalar; constant for sequence alpha_n (Robbins-Monro).

rm.alpha	scalar; noise added to gradient (Robbins-Monro).
n.aux.draws	count; number of auxiliary networks drawn from the ERGM likelihood at every stochastic approximation iteration and for the curvature adjustment.
aux.thin	count; number of auxiliary iterations between network draws after the first network is drawn.
estimate	If "MLE" (the default), then an approximate maximum likelihood estimator is used as a starting point for the posterior mode search. If "CD", the Monte-Carlo contrastive divergence estimate is used. See ergm.
seed	integer; seed for the random number generator. See set.seed.
map.method	character; "newton" (default) or "rm" (Robbins-Monro, as in Bergm 5).
map.iters	count; number of Newton stochastic approximation iterations.
init	vector; optional starting value for the posterior mode search. If given, the ergm fit is skipped.
verbose	logical; print progress information.
...	Additional arguments, to be passed to the ergm function. See ergm.

Details

The pseudo-posterior is sampled by a compiled random-walk Metropolis kernel. The mode of the true posterior, which anchors the calibration, is found by default with a Newton-type stochastic approximation that uses the simulated covariance of the sufficient statistics and a persistent auxiliary network chain (map.method = "newton"); this typically converges in about ten iterations, whereas the Robbins-Monro recursion of previous versions (map.method = "rm") needs hundreds. The pseudo-posterior mode is found exactly by Newton-Raphson.

Value

An object of class bergm, with additional elements Theta_star (estimated posterior mode), Theta_PL (pseudo-posterior mode), W (curvature adjustment matrix) and map.trajectory.

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 with Bayesian data augmentation (Koskinen, Robins and Pattison, 2010).

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,
  constraints = NULL,
  thin = 1,
  saveEveryX = NULL,
  cut.reject = FALSE,
  saveEveryXName = "partialBergmEstimate.rda",
  imputeAllItr = TRUE,
  imputeLast = TRUE,
  nImp = NULL,
  missingUpdate = NULL,
  imputeData = NULL,
  attributeNames = NULL,
  miceIt = 5,
  onlyKeepImputation = FALSE,
  adapt = TRUE,
  target.accept = 0.25,
  log.prior = NULL,
```

```
  prior.decay = c(meanlog = log(0.5), sdlog = 0.75),
  verbose = TRUE,
  ...
)
```

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 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 (or scalar/vector diagonal); 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 twice the number of free model parameters (at least 4).
gamma	scalar; parallel adaptive direction sampling move factor (starting value when adapt = TRUE).
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.
constraints	formula; constraints on the support of the network distribution, as in ergm.
thin	count; the thinning interval between consecutive observations.
saveEveryX	count; if not NULL, the partial posterior sample and imputations are saved to saveEveryXName every saveEveryX iterations.
cut.reject	logical; if TRUE, repeated (rejected) draws are removed from the pooled sample (backward compatibility only; default FALSE).
saveEveryXName	character; the name of the partial estimation object.
imputeAllItr	logical; default TRUE: missing data are imputed for every chain at every iteration (exact data augmentation). If FALSE, a chain's imputation is only refreshed after an accepted move, as in Bergm 5.
imputeLast	logical; kept for backward compatibility. Imputations always use the chain's current parameter value, which is what makes the sampler valid.
nImp	count; number of imputed networks to be returned. If NULL, no imputed network is returned. Imputations are taken at evenly spaced post burn-in iterations, cycling through the chains.
missingUpdate	count; number of tie updates in each imputation step. By default equal to the number of missing ties.

`imputeData`	data.frame; a data.frame with all attribute variables that should be imputed and additional attributes that should be used for the imputation. All non-numeric variables need to be specified as factors. Names of vertex attributes and variable names need to be identical.
`attributeNames`	character vector; the names of all variables that need to be imputed.
`miceIt`	count; number of iterations in the MICE imputation. Default is 5.
`onlyKeepImputation`	logical; should only imputations be returned, and no bergm estimate.
`adapt`	logical; tune gamma during burn-in (see `bergm`).
`target.accept`	scalar; target acceptance rate for `adapt`.
`log.prior`	function; optional user-defined log prior (see `bergm`).
`prior.decay`	named vector or list with elements `meanlog` and `sdlog`; log-normal prior for the decay parameters of curved terms (e.g. `gwesp.decay`, `gwdegree.decay`, `altkstar.lambda`), recycled over all of them or given one value per decay parameter. Default: median 0.5, `sdlog` 0.75. Decay parameters are sampled on the log scale, so they are always positive; `prior.mean` and `prior.sigma` apply to the other parameters and may be given either for all parameters (decay entries are ignored) or for the non-decay ones only.
`verbose`	logical; print progress information.
`...`	additional arguments passed to `control.simulate.formula`.

Details

Every chain of the population carries its own imputed network, and missing tie variables are updated by a Gibbs-type step from the ERGM restricted to the missing dyads given the chain's current parameter. This is a valid data-augmentation sampler for each chain (in Bergm 5 a single imputed network was shared by all chains). Imputation steps act directly on **ergm**'s compiled network states, which makes imputing at every iteration cheap; network objects are only built for the returned imputations.

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

Koskinen, J.H., Robins, G.L., and 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., and 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,
  plot = TRUE,
  cores = 1,
  seed = NULL,
  ...
)
```

Arguments

x	an R object of class bergm.
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.
plot	logical; draw the diagnostic plots (default TRUE).
cores	count; number of parallel workers.
seed	integer; optional seed.
...	additional arguments, passed to plot.bgof.

Details

For every sampled posterior parameter one network is simulated from the model, starting from the observed network. The goodness-of-fit statistics are monitored inside **ergm**'s compiled sampler (as additional zero-coefficient model terms), so no network objects are created and the posterior predictive distribution of the model statistics is obtained at no extra cost. Simulations can run in parallel with cores > 1.

Value

An object of class bgof (returned invisibly when plotted): simulated (sim.*) and observed (obs.*) proportions for each goodness-of-fit statistic, plus sim.model/obs.model and posterior predictive p-values (pval.model) for the model statistics.

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. doi:10.18637/jss.v061.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 (Bouranis, Friel and Maire, 2018).

Usage

```
ergmAPL(
  formula,
  aux.iters = NULL,
  n.aux.draws = NULL,
  aux.thin = NULL,
  ladder = NULL,
  estimate = c("MLE", "CD"),
  seed = 1,
  mle = NULL,
  cores = 1,
  llk.method = c("bridge", "ladder"),
  bridge.control = 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.
aux.iters	count; number of auxiliary iterations used for drawing the first network from the ERGM likelihood (for the curvature matching and, with llk.method = "ladder", at every rung of the ladder). Default 3000.
n.aux.draws	count; number of auxiliary networks drawn from the ERGM likelihood. Default 50.
aux.thin	count; number of auxiliary iterations between network draws after the first network is drawn. Default 50.
ladder	count; length of temperature ladder (>=3) for llk.method = "ladder". Default 50.
estimate	If "MLE" (the default), then an approximate maximum likelihood estimator is returned. If "CD", the Monte-Carlo contrastive divergence estimate is returned. See ergm.
seed	integer; seed for the random number generator. See set.seed.
mle	optional; a fitted ergm object or a numeric vector of coefficients to use as the MLE, skipping the ergm() fit.
cores	count; number of parallel workers for the ladder (at most 2 are used, one per sweep).
llk.method	character; "bridge" (default) or "ladder", the method used to estimate the log-likelihood at the MLE (see Details).
bridge.control	optional control.ergm.bridge list for llk.method = "bridge".
...	Additional arguments, to be passed to the ergm function. See ergm.

Details

The log-likelihood at the MLE, and hence the adjustment constant, is estimated by default (llk.method = "bridge") by bridge sampling from the dyad-independent submodel of the formula, using ergm.bridge.dindstart.llk. The reference model has an exact likelihood and is close to the target, which keeps the estimate accurate on large networks. llk.method = "ladder" uses the stepping-stone ladder $\theta_t = t\,\hat{\theta}_{MLE}$, $t \in [0, 1]$, of previous versions, traversed in both directions with warm-started chains and averaging forward and reverse estimates (which removes most of the bias of a single sweep); it needs many rungs when the network is large, because the path starts at the uniform random graph. With cores > 1 the two sweeps run in parallel. The returned object can be passed to evidenceCJ and evidencePP through their apl argument to avoid recomputing it.

Value

An object of class ergmAPL: a list with formula, Theta_MLE, Theta_PL, W, logC, ll_true, logztheta and ll_adjpseudo.

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:
data(florentine)
apl <- ergmAPL(flomarriage ~ edges + kstar(2))
apl

## End(Not run)
```

`evidence` *Wrapper function for evidence estimation*

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 `evidenceCJ` and `evidencePP`.

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

# Compare two models, reusing one pseudolikelihood adjustment per model:
a1 <- ergmAPL(flomarriage ~ edges + kstar(2))
a2 <- ergmAPL(flomarriage ~ edges + triangle)
e1 <- evidence("PP", formula = flomarriage ~ edges + kstar(2), apl = a1)
e2 <- evidence("PP", formula = flomarriage ~ edges + triangle, apl = a2)
e1$log.evidence - e2$log.evidence  # log Bayes factor

## 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 = 50,
  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"),
  apl = NULL,
  cores = 1,
```

```
  verbose = TRUE,
  ...
)
```

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; mean vector of the multivariate Normal prior. By default set to a vector of 0's.
prior.sigma	square matrix (or scalar/vector diagonal); 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 ergmAPL.
n.aux.draws	count; number of auxiliary networks drawn from the ERGM likelihood. See ergmAPL.
aux.thin	count; number of auxiliary iterations between network draws after the first network is drawn. See ergmAPL.
ladder	count; length of temperature ladder (>=3). See ergmAPL.
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.
V.proposal	count; diagonal entry for the multivariate Normal proposal. By default set to 1.5.
num.samples	integer; number of proposal draws used in the marginal likelihood estimate.
seed	integer; seed for the random number generator.
estimate	If "MLE" (the default), then an approximate maximum likelihood estimator is returned. If "CD" , the Monte-Carlo contrastive divergence estimate is returned. See ergm.
apl	optional; an ergmAPL object for the same model, to reuse an existing pseudo-likelihood adjustment.
cores	count; number of parallel workers used by ergmAPL.
verbose	logical; print progress information.
...	additional arguments, to be passed to the ergm function. See ergm and ergmAPL.

Details

The adjusted pseudo-posterior is sampled with a compiled random-walk Metropolis kernel and all pseudolikelihood evaluations needed by the estimator are vectorised in compiled code. The Metropolis-Hastings acceptance probabilities in the Chib-Jeliazkov identity are computed with the full (pseudo-)posterior, i.e. including the prior.

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"),
  apl = NULL,
  cores = 1,
  verbose = TRUE,
  ...
)
```

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; mean vector of the multivariate Normal prior. By default set to a vector of 0's.
prior.sigma	square matrix (or scalar/vector diagonal); 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 ergmAPL.
n.aux.draws	count; number of auxiliary networks drawn from the ERGM likelihood. See ergmAPL.
aux.thin	count; number of auxiliary iterations between network draws after the first network is drawn. See ergmAPL.
ladder	count; length of temperature ladder (>=3). See ergmAPL.
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.
V.proposal	count; diagonal entry for the multivariate Normal proposal. By default set to 1.5.
seed	integer; seed for the random number generator.
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 ergm.

apl	optional; an ergmAPL object for the same model, to reuse an existing pseudo-likelihood adjustment.
cores	count; number of parallel workers used by ergmAPL.
verbose	logical; print progress information.
...	additional arguments, to be passed to the ergm function. See ergm and ergmAPL.

Details

Each tempered adjusted pseudo-posterior is sampled with a compiled random-walk Metropolis kernel, and the zero-variance control variates and the corrected trapezoidal rule (Friel, Hurn and Wyse, 2014) are computed with vectorised compiled 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

Friel, N., Hurn, M., & Wyse, J. (2014). Improving power posterior estimation of statistical evidence. Statistics and Computing, 24(5), 709-723.

Examples

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

PPE <- evidencePP(flomarriage ~ edges + kstar(2),
                  aux.iters   = 500,
                  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: posterior density, trace plot (one line per chain when available) and autocorrelation for every free parameter.

Usage

```
## S3 method for class 'bergm'
plot(x, lag.max = 50, ...)
```

Arguments

`x`	an R object of class `bergm`.
`lag.max`	integer; maximum lag for the autocorrelation plot.
`...`	additional arguments, to be passed to lower-level functions.

Examples

```
## Not run:
data(florentine)
p.flo <- bergm(flomarriage ~ edges + kstar(2), main.iters = 1000, seed = 1)
plot(p.flo)

## End(Not run)
```

plot.bgof *Plot Bayesian goodness-of-fit diagnostics*

Description

Plot Bayesian goodness-of-fit diagnostics

Usage

```
## S3 method for class 'bgof'
plot(
  x,
  n.deg = NULL,
  n.dist = NULL,
  n.esp = NULL,
  n.ideg = NULL,
  n.odeg = NULL,
  ...
)
```

Arguments

x	an object of class bgof.
n.deg, n.dist, n.esp, n.ideg, n.odeg	counts; see bgof.
...	unused.

---

simulate.bergm *Posterior predictive simulation from a* bergm *fit*

Description

Draws parameters from the posterior sample and simulates one network for each, starting from the observed network.

Usage

```
## S3 method for class 'bergm'
simulate(
  object,
  nsim = 1,
  seed = NULL,
  aux.iters = 10000,
  output = c("network", "stats"),
  cores = 1,
  ...
)
```

Arguments

object	an object of class bergm.
nsim	count; number of networks.
seed	integer; optional seed.
aux.iters	count; number of MCMC iterations per network.
output	"network" (a list of networks) or "stats" (matrix of model statistics).
cores	count; parallel workers (for output = "stats").
...	unused.

summary.bergm *Summary of BERGM posterior output*

Description

This function summarises MCMC output for bergm objects.

Usage

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

Arguments

object	an R object of class bergm.
returnTable	logical; by default FALSE. Should the results table be returned.
...	additional arguments, to be passed to lower-level functions.

Value

Invisibly (or, with returnTable = TRUE, visibly) the table of posterior summaries. When per-chain samples are available, the table also reports the potential scale reduction factor (Rhat).

Index

∗ **datasets**
lazega, 22

as.mcmc.bergm *(*bergm-methods*)*, 6
as.mcmc.list.bergm *(*bergm-methods*)*, 6

bergm, 2, *11*
bergm-methods, 6
bergmC, 6
bergmM, 9
bgof, 12, *24*

coef.bergm *(*bergm-methods*)*, 6
control.ergm.bridge, *15*
control.simulate.formula, *4*, *11*

ergm, *3*, *7, 8*, *10*, *15*, *18*, *20, 21*
ergm.bridge.dindstart.llk, *15*
ergmAPL, 14, *18*, *20, 21*
evidence, 16
evidenceCJ, *15*, 17
evidencePP, *15*, 19

lazega, 22

network, *3*, *7*, *10*, *15*, *18*, *20*

plot.bergm, 23
plot.bgof, *13*, 23

simulate.bergm, 24
summary.bergm, 25

vcov.bergm *(*bergm-methods*)*, 6