Package 'Bstergm'

November 25, 2026

Type Package

Title Bayesian Separable Temporal Exponential Random Graph Models

Version 0.1.0

Description Bayesian inference for separable temporal exponential random graph models (STERGMs) of dynamic binary networks and for the separable two-layer STERGMs of dynamic signed networks (Caimo and Gollini, 2026). Each transition is decomposed into the formation and persistence of ties (and, for signed networks, of their signs); the resulting doubly-intractable posteriors are sampled with approximate exchange algorithms based on adaptive Metropolis-Hastings or adaptive direction sampling. Posterior predictive goodness of fit, diagnostics of the auxiliary simulations, simulation and forecasting tools are provided.

License GPL (>= 3)

Encoding UTF-8

LazyData true

Depends R (>= 4.2.0), ergm (>= 4.6.0), network

Imports ergm.multi, statnet.common, Matrix, coda, parallel, stats, graphics, grDevices, utils, methods

LinkingTo ergm

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

URL https://github.com/acaimo/Bstergm

BugReports https://github.com/acaimo/Bstergm/issues

NeedsCompilation yes

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

Author Alberto Caimo [aut, cre], Isabella Gollini [aut]

Config/roxygen2/version 8.1.1

Repository CRAN

Date/Publication 2026-11-25 20:40:08 UTC

Contents

active_signs-ergmConstraint . . . 2
aea_amh . . . 3
as.mcmc.list.bstergm . . . 4
as_signed_network . . . 4
aux_diagnostics . . . 5
bgof . . . 7
bstergm . . . 8
coef.bstergm . . . 12
control.bstergm . . . 13
degree_neg-ergmTerm . . . 14
esp_signed-ergmTerm . . . 15
gwdegree_neg-ergmTerm . . . 15
gwese_neg-ergmTerm . . . 16
gwese_pos-ergmTerm . . . 16
gwesf_neg-ergmTerm . . . 17
gwesf_pos-ergmTerm . . . 17
gwesp_signed-ergmTerm . . . 18
network_series . . . 19
plot.bgof_bstergm . . . 20
plot.bstergm . . . 20
predict.bstergm . . . 21
senators70_am . . . 23
signed_adjacency . . . 24
signed_series . . . 25
signed_triad_census . . . 26
simulate.bstergm . . . 27
simulate_stats . . . 28
summary.bstergm . . . 29
transition_networks . . . 30
triangle_signed-ergmTerm . . . 30

Index **32**

`active_signs-ergmConstraint`

Only the signs of active dyads may change

Description

Sample space constraint for signed networks built with `as_signed_network()`: only the active dyads (network attribute `.active`) may change, and only between positive and negative. This is the sample space of the sign process of a separable signed ERGM, where the interaction layer is held fixed.

Usage

```
# active_signs
```

See Also

as_signed_network()

---

aea_amh *Approximate exchange algorithm with adaptive Metropolis-Hastings*

Description

Shortcuts for bstergm_sign() with method = "amh" and method = "ads". They also accept the arguments of version 0.1 of the package (then called **B2Lstergm**), i.e. y_t, ergm_sim_iters, mu.prior_F, Sigma.prior_F, mu.prior_P, Sigma.prior_P, init_s_F, init_s_P, adapt_start, adapt_interval, gamma_F, gamma_P, sigma.prop_F and sigma.prop_P; the networks L.x_F, L.x_P, L.z_F, L.z_P and the statistics sy are no longer needed, as they are computed from y_t.

Usage

```
aea_amh(formula, data = NULL, ...)

aea_ads(formula, data = NULL, ...)
```

Arguments

formula, data	see bstergm_sign().
...	further arguments passed to bstergm_sign().

Value

A bstergm_sign() fit. As in version 0.1, its elements theta_F and theta_P are arrays of posterior draws (iterations x parameters x chains).

See Also

bstergm_sign()

Examples

```
data(senators70_am)
senate <- signed_series(senators70_am[1:3])
fit <- aea_amh(~ edges + gwesf_pos(0.6) + gwese_pos(0.6), senate,
               iters = 100, burn.in = 50, aux.iters = 1000, seed = 1, verbose = FALSE)
dim(fit$theta_F)
```

as.mcmc.list.bstergm *Convert bstergm draws to coda objects*

Description

Convert bstergm draws to coda objects

Usage

```
## S3 method for class 'bstergm'
as.mcmc.list(
  x,
  components = x$components,
  prefix = length(components) > 1,
  ...
)

## S3 method for class 'bstergm'
as.mcmc(x, components = x$components, prefix = length(components) > 1, ...)
```

Arguments

x	a bstergm() or bstergm_sign() fit.
components	components to include; with both, the parameter names are prefixed by F: and P:.
prefix	whether to prefix the parameter names with the component.
...	unused.

Value

An coda::mcmc.list() with one chain per sampler chain, or, for as.mcmc(), an coda::mcmc() object with the chains stacked.

as_signed_network *Signed network representation used by the native signed terms*

Description

Converts a signed network to the representation used by the signed-network ergm terms of this package: a network whose ties are the *positive* ties, with the active (interacting) dyads stored as a two-column edge list in the network attribute .active. Negative ties are the active dyads that are not ties.

Usage

```
as_signed_network(y, vertex.attr = NULL, sign.attr = "sign", ...)
```

Arguments

y	a signed network: a square symmetric matrix with entries in {-1, 0, 1}; an undirected network whose edges carry the edge attribute `sign` (or a different attribute given by `sign.attr`); or a two-layer `ergm.multi::Layer()` network with layers `positive` and `active` as used by earlier versions of this package.
vertex.attr	optional data frame (or named list) of vertex attributes with one row per node.
sign.attr	name of the edge attribute holding the signs when `y` is a network.
...	unused.

Value

An undirected `network` of positive ties with network attribute `.active`.

See Also

`signed_adjacency()`, `signed_triad_census()`, gwese_pos

Examples

```
data(senators70_am)
y <- senators70_am$S99
nw <- as_signed_network(y, vertex.attr = attr(y, "senators"))
nw
summary(nw ~ edges + gwesf_pos(0.6) + gwese_pos(0.6) + gwese_neg(0.6) +
          triangle_signed())
```

---

aux_diagnostics *Diagnose the length of the auxiliary simulations*

Description

The approximate exchange algorithm assumes that every auxiliary network, simulated by Metropolis-Hastings starting at the observed network, is a draw from the model. If the simulations are too short, the sampled posterior is closer to a contrastive-divergence approximation than to the posterior of the model. This function runs much longer simulations from the observed networks, at the posterior mean or at posterior draws, and records the trajectories of the model statistics. The trajectories should settle within the number of steps used for fitting (the dotted vertical line in the plot). Large drifts, or chains that move towards and freeze at extreme configurations, indicate that `aux.iters` (or `aux.visits`) should be increased, or that the model is near-degenerate at these parameter values.

Usage

```
aux_diagnostics(
  object,
  components = object$components,
  transitions = 1,
  visits = 50,
  nsamples = 100,
  ndraws = 1,
  seed = NULL
)

## S3 method for class 'aux_diagnostics_bstergm'
print(x, ...)

## S3 method for class 'aux_diagnostics_bstergm'
plot(x, ...)
```

Arguments

object	a bstergm() or bstergm_sign() fit.
components	components to check.
transitions	indices of the transitions to check.
visits	length of the simulations, in proposals per free dyad.
nsamples	number of recorded states along each simulation.
ndraws	number of parameter values: 1 uses the posterior mean, larger values use draws spread over the posterior sample.
seed	random seed.
x	an aux_diagnostics_bstergm object.
...	unused.

Value

An object of class aux_diagnostics_bstergm: for each component a list over transitions and parameter values of trajectories of the statistics, with the observed statistics and the number of visits per free dyad used for fitting.

See Also

bgof(), control.bstergm()

Examples

```
data(senators70_am)
senate <- signed_series(senators70_am[1:3])
fit <- bstergm_sign(~ edges + gwesf_pos(0.6) + gwese_neg(0.6), senate,
               iters = 100, burn.in = 50, aux.iters = 1000, seed = 1, verbose = FALSE)
dg <- aux_diagnostics(fit, visits = 10, nsamples = 20, seed = 1)
```

```
dg
plot(dg)
```

---

bgof *Posterior predictive goodness of fit*

Description

Simulates, for posterior draws of the parameters, the formation and persistence networks of every transition (starting from the observed networks, as in the auxiliary step of the sampler) and compares their statistics with the observed ones. Following Caimo and Gollini, the centred cumulative differences $\sum_t \{s(z_t^{U\prime}) - s(z_t^U)\}$ of the model statistics should be centred around zero for a well-fitting model. Additional statistics not in the model (e.g. degree distributions or the signed triangle census) can be checked with `monitor`.

Usage

```
bgof(
  object,
  nsim = 100,
  monitor = NULL,
  aux.iters = NULL,
  components = object$components,
  cores = getOption("Bstergm.cores", 1L),
  seed = NULL,
  verbose = TRUE
)

## S3 method for class 'bgof_bstergm'
print(x, ...)

## S3 method for class 'bgof_bstergm'
summary(object, ...)
```

Arguments

`object`	a `bstergm()` or `bstergm_sign()` fit.
`nsim`	number of posterior predictive simulations.
`monitor`	optional one-sided formula of additional statistics, e.g. `~ degree(0:10) + degree_neg(0:10) + triangle_signed()` for the sign process.
`aux.iters`	number of steps of each simulation; by default five times the number used for fitting.
`components`	components to check.
`cores`	number of processes. Each simulation has its own random number stream, so results do not depend on the number of cores.

seed	random seed.
verbose	print progress messages.
x	a bgof_bstergm object.
...	unused.

Value

An object of class bgof_bstergm with, for each component, an array sim (simulations x statistics x transitions) of simulated statistics, the matrix obs of observed statistics and the matrix diff of centred cumulative differences.

summary() returns, for each component, a table with the observed statistics summed over transitions, the mean and 95% interval of the simulated ones and a two-sided posterior predictive p-value.

See Also

plot.bgof_bstergm(), bstergm()

Examples

```
data(senators70_am)
senate <- signed_series(senators70_am[1:3],
                        vertex.attr = attr(senators70_am$S99, "senators"))
fit <- bstergm_sign(~ edges + nodematch("Party") + gwesf_pos(0.6) + gwese_pos(0.6),
               data = senate, iters = 100, burn.in = 50, aux.iters = 1000, seed = 1,
               verbose = FALSE)
gf <- bgof(fit, nsim = 10, monitor = ~ triangle_signed(), seed = 2, verbose = FALSE)
gf
plot(gf)
```

---

bstergm *Bayesian separable temporal ERGMs*

Description

bstergm() samples the posterior distribution of the formation and persistence parameters of a separable temporal ERGM (STERGM) for a sequence of binary networks. bstergm_sign() does the same for the sign process of the separable two-layer STERGM of a sequence of signed networks. Both use the approximate exchange algorithm with adaptive Metropolis-Hastings (method = "amh") or adaptive direction sampling (method = "ads").

Usage

```
bstergm(
  formula,
  data,
  method = c("amh", "ads"),
  components = c("F", "P"),
```

```
  iters = 30000,
  burn.in = 10000,
  thin = 1,
  n_chains = NULL,
  aux.iters = NULL,
  prior.mean = 0,
  prior.sigma = 10,
  init = "mple",
  offset.coef = NULL,
  cores = getOption("Bstergm.cores", 1L),
  seed = NULL,
  control = control.bstergm(),
  verbose = TRUE
)

bstergm_sign(
  formula,
  data,
  method = c("amh", "ads"),
  components = c("F", "P"),
  iters = 30000,
  burn.in = 10000,
  thin = 1,
  n_chains = NULL,
  aux.iters = NULL,
  prior.mean = 0,
  prior.sigma = 10,
  init = "mple",
  offset.coef = NULL,
  cores = getOption("Bstergm.cores", 1L),
  seed = NULL,
  control = control.bstergm(),
  verbose = TRUE
)
```

Arguments

formula	a one-sided ergm formula, or a list of such formulas (one per transition, for time-varying covariates) with identical terms. For `bstergm()` it is an ordinary ergm formula evaluated on the formation and persistence networks. For `bstergm_sign()` it is evaluated on the positive ties of each formation or persistence network: ordinary ergm terms (e.g. `edges`, `nodematch("Party")`, `gwdegree(0.2, fixed = TRUE)`, `edgecov(x)`) refer to positive ties, and the signed-network terms of this package (gwesf_pos, gwese_pos, gwese_neg, gwdegree_neg, ...) refer to positive and negative ties. The multilayer terms of **ergm.multi** on the layers `positive` and `active`, as in earlier versions of the package, are also accepted (slower).
data	for `bstergm()`, a list of at least two undirected binary networks (adjacency ma-

trices or network objects), a network_series() object, or a signed_series() object, whose interactions (ties of either sign) are then modelled. For bstergm_sign(), a signed_series() object, or a list of signed networks accepted by it.

method — "amh" (adaptive Metropolis-Hastings, default) or "ads" (adaptive direction sampling).

components — which components to fit: "F" (formation) and/or "P" (persistence).

iters — total number of iterations of each chain, including burn-in.

burn.in — number of initial iterations discarded.

thin — thinning interval of the stored draws.

n_chains — number of chains: independent chains for "amh" (default 1), the population size for "ads" (default twice the number of parameters, at least 3).

aux.iters — number of steps of each auxiliary network simulation; see control.bstergm() for the default.

prior.mean, prior.sigma — mean and covariance of the multivariate normal prior of each component: scalars, vectors (diagonal covariance), matrices, or lists with elements F and P for component-specific priors. The defaults are $N(0, 10I)$.

init — starting values: "mple" (default) starts from the penalised maximum pseudo-likelihood estimate pooled over transitions; "ads" (with method = "amh") first runs adaptive direction sampling for control.bstergm(ads.init.iters =) iterations, as in Caimo and Gollini, and starts the adaptive Metropolis-Hastings chains from its final population, with an initial proposal covariance estimated from it; "prior" starts from the prior mean plus a small uniform perturbation (the behaviour of version 0.1); a numeric vector, matrix (one column per chain), or list with elements F and P; or a previous fit of the same model, to continue its chains from their last stored states with the adapted proposal covariance.

offset.coef — values of the parameters of offset() terms.

cores — number of processes (default getOption("Bstergm.cores", 1)). The formation and persistence components and, for "amh", the chains are independent and are run in parallel (forking on Unix-alikes, a socket cluster on Windows). Results do not depend on the number of cores.

seed — random seed, for reproducible results.

control — a list of tuning parameters, see control.bstergm().

verbose — print progress messages.

Details

Binary networks. For each transition $y_{t-1} \to y_t$ the formation network $y_t^F = y_{t-1} \cup y_t$ and the persistence network $y_t^P = y_{t-1} \cap y_t$ are modelled by ERGMs with parameters θ^F on $\{y^F \supseteq y_{t-1}\}$ and θ^P on $\{y^P \subseteq y_{t-1}\}$, as in the separable temporal ERGM of Krivitsky and Handcock (2014).

Signed networks. A signed network is decomposed into an interaction layer x and a sign layer z (see signed_series()), and the formation and persistence networks are formed for both layers. bstergm_sign() models $z_t^U \mid x_t^U$ ($U = F, P$) by an ERGM with parameter ζ^U on the sample space in which the sign of every interacting dyad of x_t^U may vary and the interactions are fixed. The interaction layer follows a binary STERGM and is fitted by bstergm() on the same signed_series().

Under independent priors the posterior factorises over the components, which are sampled independently (and in parallel).

Algorithm. The likelihood is intractable, so the posterior is sampled with the approximate exchange algorithm (Caimo and Friel, 2011): at each iteration a parameter θ' is proposed, an auxiliary network is simulated from the model at θ' for every transition, starting at the observed network, and θ' is accepted with probability

$$\min\left\{1, \exp\left[(\eta(\theta)-\eta(\theta'))^{\top}\sum_{t}\{s(y_t')-s(y_t)\}\right]\frac{\pi(\theta')}{\pi(\theta)}\right\}$$

With `method = "amh"` proposals are Gaussian random walks whose covariance is adapted to the chain history (Haario et al., 2001); with `method = "ads"` each chain of a population moves along the difference of two other chains (Caimo and Friel, 2011). Caimo and Gollini use adaptive direction sampling only to find starting values during burn-in and then adaptive Metropolis-Hastings; this scheme is obtained with `method = "amh"` and `init = "ads"`.

Computation. The ergm model, proposal and constraints of every transition are initialised once, and only the C sampler of **ergm** is called during the iterations; with the signed-network terms of this package the model statistics are also computed in C. Statistics that are constant on the sample space (such as `L(~edges, ~active)` of the multilayer formulation) are detected and dropped.

Length of the auxiliary simulations. The approximate exchange algorithm treats the end of each auxiliary simulation as an exact draw from the model. Simulations that are too short return networks close to the observed ones, and the sampled distribution then resembles a contrastive-divergence approximation more than the posterior. This matters most for near-degenerate models. Check the auxiliary simulations with `aux_diagnostics()` and the fit with `bgof()`. If needed, increase `aux.iters` or `control.bstergm(aux.visits =)`; the cost per iteration grows linearly.

Lagged covariates. The networks of each transition carry the previous time point as network attributes, so that lagged dyadic covariates can be used as `edgecov("lag.active")` (any tie at time $t-1$) and, for signed data, `edgecov("lag.pos")` and `edgecov("lag.neg")`.

Value

An object of class `bstergm`, a list including

`theta`	a list with, for each fitted component, an array of posterior draws (iterations x parameters x chains);
`theta_F`, `theta_P`	the same arrays (for compatibility with version 0.1);
`acceptance`	acceptance rates of each chain;
`mple`	the pseudo-likelihood estimates used for initialisation;
`init.ads`	with `init = "ads"`, a summary of the initial adaptive direction sampling run of each component;
`obs.stats`	the observed statistics of each transition;
`process`	`"sign"` for `bstergm_sign()`, `"interaction"` for `bstergm()`;
`prior`, `control`, `formula`, `data`, `time`	inputs and timings.

Use `summary.bstergm()`, `plot.bstergm()`, `bgof()` and `as.mcmc.list.bstergm()` to analyse it.

References

Caimo, A. and Gollini, I. (2025). Separable models for dynamic signed networks. arXiv:2505.07669.

Caimo, A. and Friel, N. (2011). Bayesian inference for exponential random graph models. *Social Networks*, 33(1), 41-55.

Haario, H., Saksman, E. and Tamminen, J. (2001). An adaptive Metropolis algorithm. *Bernoulli*, 7(2), 223-242.

Krivitsky, P. N. and Handcock, M. S. (2014). A separable model for dynamic networks. *Journal of the Royal Statistical Society B*, 76(1), 29-46.

See Also

network_series(), signed_series(), summary.bstergm(), bgof(), aux_diagnostics(), control.bstergm(), predict.bstergm()

Examples

```
data(senators70_am)
senate <- signed_series(senators70_am[1:3],
                        vertex.attr = attr(senators70_am$S99, "senators"))

# Very short runs for illustration; in practice use many more iterations
# (e.g. iters = 30000, burn.in = 10000) and the default auxiliary length.

# Binary STERGM of the interactions (ties of either sign)
fit_x <- bstergm(~ edges + nodematch("Party"), data = senate,
                 iters = 100, burn.in = 50, aux.iters = 1000, seed = 1, verbose = FALSE)
summary(fit_x)

# Sign process given the interactions
fit_z <- bstergm_sign(~ edges + nodematch("Party") + gwesf_pos(0.6) + gwese_pos(0.6),
                      data = senate, iters = 100, burn.in = 50, aux.iters = 1000,
                      seed = 1, verbose = FALSE)
summary(fit_z)
```

---

coef.bstergm *Posterior means of a bstergm fit*

Description

Posterior means of a bstergm fit

Usage

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

Arguments

object	a bstergm() or bstergm_sign() fit.
...	unused.

Value

A list of named vectors of posterior means, one per component.

---

control.bstergm *Tuning parameters for bstergm*

Description

Tuning parameters for bstergm

Usage

```
control.bstergm(
  aux.iters = NULL,
  aux.visits = 3,
  aux.min = 500,
  MCMC.prop = ~.,
  MCMC.prop.weights = "default",
  MCMC.prop.args = list(),
  translate = TRUE,
  adapt.start = 100,
  adapt.interval = 50,
  adapt.stop = Inf,
  adapt.scale = NULL,
  adapt.eps = 1e-06,
  proposal.sd = NULL,
  ads.gamma = 0.5,
  ads.var = 0.025,
  init.jitter = 0.5,
  ads.init.iters = 1000,
  ads.init.chains = NULL,
  print.every = 1000
)
```

Arguments

aux.iters	number of Metropolis-Hastings steps of each auxiliary network simulation. If NULL (default) it is set separately for each transition to aux.visits times its number of free dyads (but at least aux.min), so that the quality of the auxiliary draws does not depend on network size. See aux_diagnostics() to check that it is large enough.

`aux.visits`, `aux.min`
: see `aux.iters`.

`MCMC.prop`, `MCMC.prop.weights`, `MCMC.prop.args`
: proposal of the auxiliary simulations, passed to `ergm::control.simulate.formula()`. The default `~.` toggles a uniformly chosen free dyad; `~sparse + .triadic` gives ergm's default triad-focused proposal.

`translate`
: if `TRUE` (default), sign-process formulas written with the multilayer terms of **ergm.multi** (`L()`, `gwespL()`), as in version 0.1, are translated to the equivalent native signed terms, which are much faster. The translation is used only if it reproduces the observed statistics exactly.

`adapt.start`, `adapt.interval`, `adapt.stop`
: adaptive Metropolis-Hastings: the proposal covariance is updated every `adapt.interval` iterations between iterations `adapt.start` and `adapt.stop`.

`adapt.scale`
: scaling of the empirical covariance; `NULL` uses $2.38^2/d$, where d is the number of parameters.

`adapt.eps`
: small constant added to the diagonal of the empirical covariance.

`proposal.sd`
: initial proposal standard deviation(s) of adaptive Metropolis-Hastings. If `NULL` (default) the initial proposal covariance is the scaled pseudo-likelihood covariance, or `0.025^2` times the identity when it is not available.

`ads.gamma`
: move factor of adaptive direction sampling.

`ads.var`
: variance of the Gaussian noise added to adaptive direction sampling proposals.

`init.jitter`
: scale of the random perturbation of the starting values: chains start at the pseudo-likelihood estimate plus `init.jitter` times a draw from its approximate covariance.

`ads.init.iters`, `ads.init.chains`
: length and population size (default `max(3, 2d)`) of the initial adaptive direction sampling run used with `init = "ads"` in `bstergm()` and `bstergm_sign()`.

`print.every`
: report progress every so many iterations when `verbose`.

Value

A list of class `control.bstergm`.

See Also

`bstergm()`, `bstergm_sign()`

`degree_neg-ergmTerm` *Degree distribution of negative ties*

Description

For each value in `d`, the number of nodes with exactly that many negative ties.

Usage

```
# binary: degree_neg(d)
```

Arguments

d	a vector of non-negative integers.

See Also

gwdegree_neg

---

esp_signed-ergmTerm *Edgewise shared partner distribution by sign*

Description

For each value in d, the number of dyads of sign base that have exactly that many shared partners whose legs match path. Useful mainly for goodness-of-fit checks.

Usage

```
# binary: esp_signed(d, base = "+", path = "++")
```

Arguments

d	a vector of non-negative integers.
base, path	as in gwesp_signed.

See Also

gwesp_signed

---

gwdegree_neg-ergmTerm *Geometrically weighted degree of negative ties*

Description

The geometrically weighted degree statistic computed on the negative ties (active dyads that are not positive), $e^{\alpha}\sum_i\{1-(1-e^{-\alpha})^{d_i^-}\}$, where d_i^- is the number of negative ties of node i. For the positive ties use the ordinary gwdegree(decay, fixed = TRUE).

Usage

```
# binary: gwdegree_neg(decay)
```

Arguments

decay	the decay parameter (required).

See Also

degree_neg, as_signed_network()

---

gwese_neg-ergmTerm *Geometrically weighted negative edgewise shared enemies*

Description

gwese^{-}: negative ties weighted by their number of shared enemies (both legs negative), i.e. the $(-,-,-)$ triad, which is balanced under weak balance theory. Equivalent to gwesp_signed(decay, base = "-", path = "--

Usage

```
# binary: gwese_neg(decay)
```

Arguments

decay	the decay parameter (required).

See Also

gwesp_signed, as_signed_network()

---

gwese_pos-ergmTerm *Geometrically weighted positive edgewise shared enemies*

Description

gwese^{+}: positive ties weighted by their number of shared enemies (both legs negative), i.e. the balanced $(+,-,-)$ triad viewed from its positive tie. Equivalent to gwesp_signed(decay, base = "+", path = "--") and to gwespL(decay, fixed = TRUE, L.base = ~positive, Ls.path = list(~active & !positive, ~active & !positive)) in the multilayer representation.

Usage

```
# binary: gwese_pos(decay)
```

Arguments

decay	the decay parameter (required).

See Also

gwesp_signed, as_signed_network()

gwesf_neg-ergmTerm *Geometrically weighted negative edgewise shared friends*

Description

gwesf^{-}: negative ties weighted by their number of shared friends (both legs positive), i.e. the unbalanced $(+,+,-)$ triad viewed from its negative tie. Equivalent to gwesp_signed(decay, base = "-", path = "++").

Usage

```
# binary: gwesf_neg(decay)
```

Arguments

decay	the decay parameter (required).

See Also

gwesp_signed, as_signed_network()

gwesf_pos-ergmTerm *Geometrically weighted positive edgewise shared friends*

Description

gwesf^{+}: positive ties weighted by their number of shared friends (both legs positive). This is the balanced $(+,+,+)$ triad viewed from a positive tie, and equals gwesp(decay, fixed = TRUE) on the positive ties. Equivalent to gwesp_signed(decay, base = "+", path = "++").

Usage

```
# binary: gwesf_pos(decay)
```

Arguments

decay	the decay parameter (required).

See Also

gwesp_signed, as_signed_network()

`gwesp_signed-ergmTerm` *Geometrically weighted edgewise shared partners by sign*

Description

Geometrically weighted edgewise shared partner statistic of a signed network, in which the base dyads have sign `base` and only shared partners whose two legs have the signs given by `path` are counted.

For a base dyad $\{i, j\}$ let sp_{ij} be the number of nodes k such that the signs of $\{i, k\}$ and $\{k, j\}$ match `path`. The statistic is

$$e^{\alpha} \sum_{\{i,j\}:\, y_{ij}=base} \{1 - (1 - e^{-\alpha})^{sp_{ij}}\}$$

It is evaluated on a signed network built by `as_signed_network()`: the ties of the network are the positive ties and the network attribute `.active` holds the interacting dyads, so that negative ties are active dyads that are not ties. The terms are implemented in C and are much faster than the equivalent multilayer gwespL terms of **ergm.multi**.

Usage

```
# binary: gwesp_signed(decay, base = "+", path = "++")
```

Arguments

`decay`	the decay parameter α (required; the curved version is not available).
`base`	sign of the base dyads, `"+"` or `"-"`.
`path`	signs of the two legs of a shared partner path: `"++"` (shared friends), `"--"` (shared enemies) or `"+-"` (one positive and one negative leg).

Details

The special cases used by Caimo and Gollini are available as gwesf_pos (gwesf^{+}), gwesf_neg (gwesf^{-}), gwese_pos (gwese^{+}) and gwese_neg (gwese^{-}). The statistic `gwesp_signed(decay, "+", "++")` is identical to the ordinary `gwesp(decay, fixed = TRUE)` on the positive ties.

See Also

`as_signed_network()`, esp_signed, triangle_signed

Examples

```
data(senators70_am)
nw <- as_signed_network(senators70_am$S99)
# The four balance-related statistics of Caimo and Gollini, and a mixed one
summary(nw ~ gwesf_pos(0.6) + gwese_pos(0.6) + gwesf_neg(0.6) + gwese_neg(0.6) +
          gwesp_signed(0.6, base = "+", path = "+-"))
```

```
# Simulate signs on the observed interactions
sim <- simulate(nw ~ edges + gwesf_pos(0.6) + gwese_neg(0.6), coef = c(-1, 0.3, -0.1),
                constraints = ~active_signs, seed = 1)
table(signed_adjacency(sim)[upper.tri(diag(70))])
```

`network_series` *Sequence of binary networks*

Description

Creates the data object used by `bstergm()`: a sequence of undirected binary networks $y_0, \ldots, y_T$ observed on the same set of nodes.

Usage

```
network_series(nets, vertex.attr = NULL)
```

Arguments

`nets`	a list of at least two undirected binary networks, given as adjacency matrices or `network` objects. A `signed_series()` object is returned unchanged: its interactions (ties of either sign) are then the binary networks.
`vertex.attr`	optional data frame (or named list) of vertex attributes, one row per node; by default taken from the first network if it is a `network` object.

Value

An object of class `c("network_series", "signed_series")` with all ties coded as 1.

See Also

`bstergm()`, `signed_series()`

Examples

```
data(senators70_am)
# Interactions (ties of either sign) of the Senate data as binary networks
nets <- lapply(senators70_am[1:3], function(m) abs(m))
network_series(nets, vertex.attr = attr(senators70_am$S99, "senators"))
```

plot.bgof_bstergm *Plot posterior predictive goodness of fit*

Description

`type = "model"` shows, for each component, boxplots of the centred cumulative differences $\sum_t \{s(z'_t) - s(z_t)\}$ of the model statistics (standardised by their simulated standard deviation when `standardize = TRUE`), which should be centred on the dashed zero line. `type = "monitor"` shows the simulated monitored statistics summed over transitions, with the observed values as black points.

Usage

```
## S3 method for class 'bgof_bstergm'
plot(
  x,
  type = c("model", "monitor"),
  components = x$components,
  standardize = TRUE,
  ...
)
```

Arguments

`x`	a `bgof()` result.
`type`	`"model"` and/or `"monitor"`.
`components`	components to plot.
`standardize`	standardise the centred differences of the model statistics, so that they share one axis.
`...`	further graphical parameters passed to `graphics::boxplot()`.

Value

`x`, invisibly.

plot.bstergm *Plot the posterior draws of a bstergm fit*

Description

For each parameter, shows the posterior density, the trace of every chain and the autocorrelation function.

Usage

```
## S3 method for class 'bstergm'
plot(
  x,
  components = x$components,
  which = c("density", "trace", "acf"),
  lag.max = 50,
  ask = length(components) > 1 && interactive(),
  ...
)
```

Arguments

x	a bstergm() or bstergm_sign() fit.
components	components to plot.
which	any of "density", "trace" and "acf".
lag.max	maximum lag of the autocorrelation plots.
ask	whether to ask before each new page (one page per component).
...	further graphical parameters passed to graphics::plot().

Value

x, invisibly.

---

predict.bstergm *Forecast networks*

Description

Simulates future networks $y_{T+1}, \ldots, y_{T+h}$ from the posterior predictive distribution of a fitted separable model. For binary networks (a bstergm() fit) each step simulates the formation and persistence networks and assembles $y_t = y_t^P \cup (y_t^F \setminus y_{t-1})$. For signed networks, a bstergm() fit of the interactions and a bstergm_sign() fit of the signs are both needed: the interactions are simulated first, then the signs of their formation and persistence networks, and the next network is assembled as

$$x_t = x_t^P \cup (x_t^F \setminus x_{t-1}), \qquad z_t = \{z_t^P \cup (z_t^F \setminus z_{t-1})\} \cap x_t,$$

where z are the positive ties; all other interactions are negative.

Usage

```
## S3 method for class 'bstergm'
predict(
  object,
  interaction = NULL,
  horizon = 1,
```

```
  nsim = 100,
  start = NULL,
  visits = 10,
  posterior = TRUE,
  seed = NULL,
  cores = getOption("Bstergm.cores", 1L),
  verbose = TRUE,
  ...
)

## S3 method for class 'bstergm_forecast'
summary(object, probs = c(0.025, 0.975), ...)
```

Arguments

object	a bstergm() or bstergm_sign() fit with both components ("F" and "P").
interaction	for signed networks, the other fit: the two fits may be given in either order. NULL (default) forecasts binary networks from a bstergm() fit.
horizon	number of future time points.
nsim	number of simulated trajectories.
start	the network to start from (a signed or binary adjacency matrix, or anything accepted by signed_adjacency()); by default the last observed network.
visits	length of each network simulation, in proposals per free dyad.
posterior	if TRUE (default), each trajectory uses a different draw from the posterior distributions; if FALSE, all use the posterior means.
seed	random seed.
cores	number of processes. Each simulation has its own random number stream, so results do not depend on the number of cores.
verbose	print progress messages.
...	unused.
probs	quantiles reported by summary().

Details

Unlike the auxiliary simulations of the sampler, which start at observed networks, these simulations start at the previous network (new interactions starting as negative), so they need to be longer; see visits.

If the fits use a list of formulas (time-varying covariates), the formula of the last transition is used for all forecasts. Lagged covariates (edgecov("lag.active"), ...) refer to the previous simulated network.

Value

An object of class bstergm_forecast, a list with

`sims` a list of `nsim` trajectories, each a list of `horizon` adjacency matrices (signed or binary);

`start` the starting network;

`signed`, `horizon`, `nsim`, `times`
the type of network, the settings and the labels of the forecast time points.

`summary()` gives, for each forecast time point, the mean and 95% interval of the numbers of ties (by sign) and of triangles (by sign composition).

See Also

`simulate.bstergm()` for posterior predictive replicates of the observed transitions.

Examples

```
data(senators70_am)
senate <- signed_series(senators70_am[1:3],
                        vertex.attr = attr(senators70_am$S99, "senators"))
# Very short runs for illustration only
fx <- bstergm(~ edges + nodematch("Party"), senate,
              iters = 30, burn.in = 15, aux.iters = 500, seed = 1, verbose = FALSE)
summary(predict(fx, horizon = 1, nsim = 2, visits = 3, seed = 2, verbose = FALSE))
fz <- bstergm_sign(~ edges + gwesf_pos(0.6) + gwese_neg(0.6), senate,
                   iters = 30, burn.in = 15, aux.iters = 500, seed = 1, verbose = FALSE)
fc <- predict(fz, fx, horizon = 1, nsim = 2, visits = 3, seed = 2, verbose = FALSE)
summary(fc)
```

`senators70_am` *Signed networks of 70 U.S. Senators, 99th to 102nd Congress*

Description

Signed networks among the 70 senators who served continuously in the 99th (1985-1987), 100th (1987-1989), 101st (1989-1991) and 102nd (1991-1993) U.S. Congresses. Positive and negative ties were extracted from bill co-sponsorship with the stochastic degree sequence model backbone of Neal (2014, 2020): a positive tie means that two senators co-sponsored significantly more bills than expected, a negative tie significantly fewer.

Usage

```
senators70_am
```

Format

A list of 4 signed adjacency matrices (`S99`, `S100`, `S101`, `S102`), each 70 x 70 and symmetric with entries -1, 0 and 1. Each matrix carries the attribute `senators`, a data frame with columns `ID`, `Senator` (name), `State` and `Party` (`"D"` or `"R"`), which can be passed as the `vertex.attr` argument of `signed_series()`.

Source

Neal, Z. P. (2020). A sign of the times? Weak and strong polarization in the U.S. Congress, 1973-2016. *Social Networks*, 60, 103-112. Data: https://figshare.com/articles/dataset/A_Sign_of_the_Times/8096429?file=20758701

References

Neal, Z. P. (2014). The backbone of bipartite projections: Inferring relationships from co-authorship, co-sponsorship, co-attendance and other co-behaviors. *Social Networks*, 39, 84-97.

Examples

```
data(senators70_am)
sapply(senators70_am, function(m) c(positive = sum(m > 0) / 2, negative = sum(m < 0) / 2))
table(attr(senators70_am$S99, "senators")$Party)
```

---

signed_adjacency *Signed adjacency matrix*

Description

Returns the signed adjacency matrix (entries -1, 0, 1) of a signed network in any of the forms accepted by `as_signed_network()`.

Usage

```
signed_adjacency(y, sign.attr = "sign")
```

Arguments

`y`	a signed network: a square symmetric matrix with entries in {-1, 0, 1}; an undirected network whose edges carry the edge attribute `sign` (or a different attribute given by `sign.attr`); or a two-layer `ergm.multi::Layer()` network with layers `positive` and `active` as used by earlier versions of this package.
`sign.attr`	name of the edge attribute holding the signs when `y` is a network.

Value

A square symmetric integer matrix.

signed_series *Sequence of signed networks*

Description

Creates the data object used by `bstergm_sign()` (and `bstergm()`): a sequence of undirected signed networks $y_0, \ldots, y_T$ observed on the same set of nodes.

Usage

```
signed_series(nets, vertex.attr = NULL, sign.attr = "sign")

## S3 method for class 'signed_series'
print(x, ...)

## S3 method for class 'signed_series'
summary(object, ...)

## S3 method for class 'signed_series'
plot(
  x,
  layout = NULL,
  vertex.col = NULL,
  edge.col = c(positive = "grey15", negative = "grey60"),
  ...
)
```

Arguments

`nets`	a list of at least two signed networks, each given as a signed adjacency matrix (entries -1, 0, 1), a `network` with a `sign` edge attribute, a network built by `as_signed_network()` or a two-layer `ergm.multi::Layer()` network with layers `positive` and `active`.
`vertex.attr`	optional data frame (or named list) of vertex attributes, one row per node, made available to the model terms (e.g. `nodematch("Party")`).
`sign.attr`	name of the edge attribute holding the signs when the networks are `network` objects.
`x`	a `signed_series` object.
`...`	unused.
`object`	a `signed_series` object.
`layout`	optional matrix of node coordinates; by default a common layout of the union of all interactions is used.
`vertex.col`	vertex colours, or the name of a vertex attribute to colour by (levels in alphabetical order get grey shades from dark to light).
`edge.col`	grey levels of positive (solid) and negative (dotted) ties.

Details

Each transition $y_{t-1} \to y_t$ is decomposed, for both the interaction layer x ($x_{ij} = |y_{ij}|$) and the sign layer z ($z_{ij} = 1$ if $y_{ij} = +1$), into

formation $x_t^F = x_{t-1} \cup x_t$, $z_t^F = z_{t-1} \cup z_t$;

persistence $x_t^P = x_{t-1} \cap x_t$, $z_t^P = z_{t-1} \cap z_t$.

These networks are returned by `transition_networks()`.

Value

An object of class `signed_series`: a list with the signed adjacency matrices `y`, the number of nodes `n`, the time labels `times` and the vertex attributes `vertex.attr`.

`summary()` returns a list with tables of tie counts by time point, tie counts of the formation and persistence networks of each transition, and the signed triad census by time point.

See Also

`bstergm_sign()`, `bstergm()`, `transition_networks()`

Examples

```
data(senators70_am)
senate <- signed_series(senators70_am[1:3],
                        vertex.attr = attr(senators70_am$S99, "senators"))
senate
summary(senate)
```

---

signed_triad_census *Signed triad census*

Description

Counts the triangles of a signed network by their number of positive ties. Under structural balance theory, triangles with three or one positive ties are balanced; under weak balance, all-negative triangles are balanced too.

Usage

```
signed_triad_census(y, sign.attr = "sign")
```

Arguments

`y`	a signed network: a square symmetric matrix with entries in {-1, 0, 1}; an undirected network whose edges carry the edge attribute `sign` (or a different attribute given by `sign.attr`); or a two-layer `ergm.multi::Layer()` network with layers `positive` and `active` as used by earlier versions of this package.
`sign.attr`	name of the edge attribute holding the signs when `y` is a network.

Value

A named integer vector with the counts of `"+++"`, `"++-"`, `"+--"` and `"---"` triangles.

Examples

```
data(senators70_am)
sapply(senators70_am, signed_triad_census)
```

---

simulate.bstergm *Simulate from a fitted model*

Description

Posterior predictive simulation of the formation and persistence networks of every observed transition, given the observed interactions (sign process) or the observed previous networks (interaction process), together with the implied networks at each time point, $y_t = y_t^P \cup (y_t^F \setminus y_{t-1})$.

Usage

```
## S3 method for class 'bstergm'
simulate(
  object,
  nsim = 1,
  seed = NULL,
  coef = NULL,
  output = c("networks", "stats"),
  aux.iters = NULL,
  cores = getOption("Bstergm.cores", 1L),
  ...
)
```

Arguments

`object`	a `bstergm()` or `bstergm_sign()` fit.
`nsim`	number of simulations.
`seed`	random seed.
`coef`	optional parameter values (free parameters, as in `coef(object)`): a list with elements `F` and/or `P`. By default, `nsim` draws spread over the posterior sample are used.
`output`	`"networks"` (signed or binary adjacency matrices) or `"stats"` (model statistics).
`aux.iters`	number of Metropolis-Hastings steps of each simulation; by default five times the number used for fitting.
`cores`	number of processes. Each simulation has its own random number stream, so results do not depend on the number of cores.
`...`	unused.

Details

For the sign process, the simulated positive ties at time t are $z_t = \{z_t^P \cup (z_t^F \setminus z_{t-1})\} \cap x_t$, and all other interactions of x_t are negative.

Value

For `output = "stats"`, a list with, for each component, an array of statistics (simulations x statistics x transitions). For `output = "networks"`, a list of `nsim` simulations, each a list with the simulated formation (`F`) and persistence (`P`) networks of every transition and, when both components were fitted, the implied networks `y` at times $1, \ldots, T$.

See Also

`bgof()`, `simulate_stats()`

Examples

```
data(senators70_am)
senate <- signed_series(senators70_am[1:3],
                        vertex.attr = attr(senators70_am$S99, "senators"))
fit <- bstergm_sign(~ edges + nodematch("Party") + gwesf_pos(0.6) + gwese_pos(0.6),
               data = senate, iters = 100, burn.in = 50, aux.iters = 1000, seed = 1,
               verbose = FALSE)
sims <- simulate(fit, nsim = 2, seed = 3)
sapply(sims[[1]]$y, signed_triad_census)
```

---

simulate_stats *Simulate the statistics of each transition*

Description

Simulates the formation and persistence networks of every transition of a sequence of signed networks for given parameter values and returns their statistics. Each simulation starts at the observed formation or persistence network.

Usage

```
simulate_stats(
  formula,
  data,
  coef_F = NULL,
  coef_P = NULL,
  process = c("sign", "interaction"),
  nsim = 1,
  aux.iters = NULL,
  control = control.bstergm(),
  seed = NULL,
  ...
)
```

Arguments

formula	a one-sided ergm formula (see bstergm()).
data	a signed_series() object or a list of signed networks.
coef_F, coef_P	parameters of the formation and persistence models (excluding statistics that are constant on the sample space, such as the L(~edges, ~active) term of the multilayer formulation). Either may be NULL to skip that component.
process	"sign" or "interaction".
nsim	number of simulations.
aux.iters	number of Metropolis-Hastings steps of each simulation (see control.bstergm() for the default).
control	a control.bstergm() list.
seed	random seed.
...	arguments of version 0.1 of this function: y_t (used as data) and ergm_sim_iters (used as aux.iters) are translated; L.x_F, L.x_P, L.z_F, L.z_P, times and model.dim are no longer needed and are ignored, as are network and network.stats (use simulate.bstergm() for simulated networks).

Value

A list with elements sz._F and sz._P: matrices (statistics x transitions) for nsim = 1, or arrays (simulations x statistics x transitions) otherwise. Constant statistics are included.

See Also

simulate.bstergm()

Examples

```
data(senators70_am)
senate <- signed_series(senators70_am[1:3])
simulate_stats(~ edges + gwesf_pos(0.6) + gwese_pos(0.6), senate,
               coef_F = c(-2, 0.5, -0.2), coef_P = c(-2, 0.5, -0.2), seed = 1)
```

---

summary.bstergm *Summarise a bstergm fit*

Description

Posterior summaries of each fitted component: mean, standard deviation, naive and time-series standard errors of the mean, quantiles, effective sample size and, with several chains, the potential scale reduction factor of Gelman and Rubin.

Usage

```
## S3 method for class 'bstergm'
summary(object, probs = c(0.025, 0.5, 0.975), ...)
```

Arguments

object	a bstergm() or bstergm_sign() fit.
probs	quantiles to report.
...	unused.

Value

An object of class summary.bstergm: a list with one table per component plus the acceptance rates.

---

transition_networks *Formation and persistence networks of each transition*

Description

Formation and persistence networks of each transition

Usage

```
transition_networks(data)
```

Arguments

data	a signed_series() object.

Value

A list with components F and P, each a list of signed adjacency matrices (one per transition), and prev, the signed adjacency matrices of the previous time points.

See Also

signed_series()

---

triangle_signed-ergmTerm *Signed triangle census*

Description

Number of triangles (three mutually active dyads) by sign composition. Under structural balance theory "+++" and "+--" are balanced, "++-" is unbalanced and "---" is balanced only under weak balance. Because the active dyads are fixed, the four counts sum to a constant, so at most three of them can enter a model together.

Usage

```
# binary: triangle_signed(type = c("+++", "++-", "+--", "---"))
```

Arguments

type	a character vector of sign patterns; the order of the signs within a pattern is irrelevant.

See Also

gwesp_signed, signed_triad_census()

Index

∗ **datasets**
  senators70_am, 23
∗ **signed**
  active_signs-ergmConstraint, 2
  degree_neg-ergmTerm, 14
  esp_signed-ergmTerm, 15
  gwdegree_neg-ergmTerm, 15
  gwese_neg-ergmTerm, 16
  gwese_pos-ergmTerm, 16
  gwesf_neg-ergmTerm, 17
  gwesf_pos-ergmTerm, 17
  gwesp_signed-ergmTerm, 18
  triangle_signed-ergmTerm, 30
∗ **triad-related**
  esp_signed-ergmTerm, 15
  gwese_neg-ergmTerm, 16
  gwese_pos-ergmTerm, 16
  gwesf_neg-ergmTerm, 17
  gwesf_pos-ergmTerm, 17
  gwesp_signed-ergmTerm, 18
  triangle_signed-ergmTerm, 30
∗ **undirected**
  degree_neg-ergmTerm, 14
  esp_signed-ergmTerm, 15
  gwdegree_neg-ergmTerm, 15
  gwese_neg-ergmTerm, 16
  gwese_pos-ergmTerm, 16
  gwesf_neg-ergmTerm, 17
  gwesf_pos-ergmTerm, 17
  gwesp_signed-ergmTerm, 18
  triangle_signed-ergmTerm, 30

active_signs-ergmConstraint, 2
aea_ads *(*aea_amh*)*, 3
aea_amh, 3
as.mcmc.bstergm *(*as.mcmc.list.bstergm*)*, 4
as.mcmc.list.bstergm, 4
as.mcmc.list.bstergm(), *11*
as_signed_network, 4
as_signed_network(), *2, 3*, *16–18*, *24, 25*
aux_diagnostics, 5
aux_diagnostics(), *11–13*

bgof, 7
bgof(), *6*, *11, 12*, *20*, *28*
bstergm, 8
bstergm(), *4*, *6–8*, *13, 14*, *19*, *21, 22*, *25–27*, *29, 30*
bstergm_sign *(*bstergm*)*, 8
bstergm_sign(), *3, 4*, *6, 7*, *13, 14*, *21, 22*, *25–27*, *30*

coda::mcmc(), *4*
coda::mcmc.list(), *4*
coef.bstergm, 12
control.bstergm, 13
control.bstergm(), *6*, *10*, *12*, *29*

degree_neg, *16*
degree_neg-ergmTerm, 14

ergm.multi::Layer(), *5*, *24–26*
ergm::control.simulate.formula(), *14*
esp_signed, *18*
esp_signed-ergmTerm, 15

graphics::boxplot(), *20*
graphics::plot(), *21*
gwdegree_neg, *9*, *15*
gwdegree_neg-ergmTerm, 15
gwese_neg, *9*, *18*
gwese_neg-ergmTerm, 16
gwese_pos, *5*, *9*, *18*
gwese_pos-ergmTerm, 16
gwesf_neg, *18*
gwesf_neg-ergmTerm, 17
gwesf_pos, *9*, *18*
gwesf_pos-ergmTerm, 17
gwesp_signed, *15–17*, *31*
gwesp_signed-ergmTerm, 18

gwespL, *18*

InitErgmConstraint.active_signs (active_signs-ergmConstraint), 2
InitErgmTerm.degree_neg (degree_neg-ergmTerm), 14
InitErgmTerm.esp_signed (esp_signed-ergmTerm), 15
InitErgmTerm.gwdegree_neg (gwdegree_neg-ergmTerm), 15
InitErgmTerm.gwese_neg (gwese_neg-ergmTerm), 16
InitErgmTerm.gwese_pos (gwese_pos-ergmTerm), 16
InitErgmTerm.gwesf_neg (gwesf_neg-ergmTerm), 17
InitErgmTerm.gwesf_pos (gwesf_pos-ergmTerm), 17
InitErgmTerm.gwesp_signed (gwesp_signed-ergmTerm), 18
InitErgmTerm.triangle_signed (triangle_signed-ergmTerm), 30

network, *4, 5*, *24*, *26*
network_series, 19
network_series(), *10*, *12*

plot.aux_diagnostics_bstergm (aux_diagnostics), 5
plot.bgof_bstergm, 20
plot.bgof_bstergm(), *8*
plot.bstergm, 20
plot.bstergm(), *11*
plot.signed_series (signed_series), 25
predict.bstergm, 21
predict.bstergm(), *12*
print.aux_diagnostics_bstergm (aux_diagnostics), 5
print.bgof_bstergm (bgof), 7
print.signed_series (signed_series), 25

senators70_am, 23
signed_adjacency, 24
signed_adjacency(), *5*, *22*
signed_series, 25
signed_series(), *10*, *12*, *19*, *23*, *29, 30*
signed_triad_census, 26
signed_triad_census(), *5*, *31*
simulate.bstergm, 27
simulate.bstergm(), *23*, *29*
simulate_stats, 28
simulate_stats(), *28*
summary.bgof_bstergm (bgof), 7
summary.bstergm, 29
summary.bstergm(), *11, 12*
summary.bstergm_forecast (predict.bstergm), 21
summary.signed_series (signed_series), 25

transition_networks, 30
transition_networks(), *26*
triangle_signed, *18*
triangle_signed-ergmTerm, 30