# Package 'cusna'

October 29, 2026

**Type** Package

**Title** Native Simulation and Estimation of Network Models

**Version** 0.1.1

**Author** Artem Maltsev [aut, cre]

**Maintainer** Artem Maltsev <MaltsevSNA@proton.me>

**Description** A self-contained native engine (a C interface over C++ host logic) for stochastic actor-oriented models (the model family of 'RSiena'), exponential random graph models (cross-sectional, temporal, and separable temporal), and models for binary actor attributes, callable from R without a Python runtime. The CRAN build is CPU-only (C++17, multi-threaded with 'OpenMP'); a 'CUDA' backend implements the same interface and is built separately from source. The data preparation, host statistics ('RSiena' Appendix B conventions), and moment targets are validated bit-for-bit against the reference implementation and reproduce 'RSiena' targets on public datasets to machine precision; the estimators match 'RSiena', 'ergm', 'btergm', and 'tergm' on public benchmark models. The method-of-moments estimator returns standard fit objects with the simulated networks on request, and the simulator also plugs into the unmodified 'RSiena' estimator.

**License** MIT + file LICENSE

**URL** https://github.com/artemmaltsev74-techcom/cusna

**BugReports** https://github.com/artemmaltsev74-techcom/cusna/issues

**Encoding** UTF-8

**SystemRequirements** C++17

**Depends** R (>= 3.5.0)

**LinkingTo** cpp11

**Imports** stats, utils

**Suggests** testthat (>= 3.0.0), jsonlite, litedown, RSiena (>= 1.4)

**VignetteBuilder** litedown

**LazyData** true

**Config/testthat/edition** 3

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2026-10-29 12:10:02 UTC

# Contents

cusna-package . . . 2
alaam_mcmle . . . 3
alaam_mple . . . 4
alaam_simulate . . . 5
cusna_abi_version . . . 6
cusna_behavior_stats . . . 7
cusna_effect . . . 7
cusna_fit . . . 9
cusna_fran . . . 10
cusna_gof_distribution . . . 11
cusna_network_stats . . . 12
cusna_set_threads . . . 13
ergm_mcmle . . . 13
ergm_mple . . . 14
ergm_simulate . . . 15
ergm_stats . . . 16
ergm_term . . . 17
mom_estimate . . . 17
s50 . . . 19
saom_data . . . 20
saom_multinet_data . . . 21
saom_simulate . . . 22
stergm_cmle . . . 24
stergm_simulate . . . 25
tergm_mple . . . 26
tergm_simulate . . . 27

**Index** **29**

---

cusna-package *cusna: native simulation and estimation of network models*

---

## Description

A self-contained native engine (a C interface over C++ host logic) for SAOM/RSiena and ERGM, callable from R without a Python runtime. The package builds the CPU backend of the engine (C++17, multi-threaded with OpenMP; see `cusna_set_threads`); a CUDA backend implements the same interface and is built separately from source.

## Details

The package covers four model families, all on the same compiled engine:

- **SAOM** (the RSiena model family): `saom_data` + `cusna_effect` + `mom_estimate` run the full Method-of-Moments (Robbins–Monro) estimator in R over the native simulator, including behavior co-evolution, composition change, and multi-network models (`mom_estimate_multinet`). The data preparation and moment targets are validated bit-for-bit against the reference implementation; the estimates agree within simulation standard errors.
- **ERGM**: `ergm_simulate` (native TNT sampler), `ergm_stats`, `ergm_mple`, and the `ergm_mcmle` maximum-likelihood estimator.
- **Temporal ERGM**: `tergm_mple` (pooled MPLE with block bootstrap), `tergm_simulate`, and the separable `stergm_cmle` (formation/persistence CMLE).
- **ALAAM** (autologistic actor attribute models): `alaam_mple`, `alaam_mcmle`, `alaam_simulate`.

The validated host statistics – the RSiena Appendix B convention carriers (`cusna_network_stats`, `cusna_behavior_stats`, `cusna_gof_distribution`) – reproduce RSiena targets on public datasets to machine precision and remain available as low-level entry points.

The `s50` panel (three waves of friendship among 50 girls, with alcohol use) is included for the examples.

---

`alaam_mcmle` *Maximum likelihood estimate of an ALAAM (MCMC-MLE)*

---

## Description

Fits an ALAAM by MCMC maximum likelihood – an external moment-matching loop over the native Gibbs sampler, the attribute-DV twin of `ergm_mcmle`. The autologistic model is a canonical exponential family, so its Fisher information is the covariance of the sufficient statistics (density, contagion, covariate products); each iteration simulates the attribute at the current parameters, forms their mean and covariance, and takes a damped Newton step until the observed statistics are matched. Unlike `alaam_mple` (a pseudo-likelihood), the fitted model reproduces the observed moments. Pure R over the native engine (no Python).

## Usage

```
alaam_mcmle(y, net, cov = NULL, contagion = "out", coef0 = NULL,
            nsim = 500L, sweeps = NULL, maxit = 30L, tol = 0.12,
            damp = 0.5, seed = 1234L, verbose = FALSE)
```

## Arguments

| | |
|---|---|
| `y` | Binary actor attribute (0/1), length n. |
| `net` | Integer/logical n*n adjacency of the fixed network. |
| `cov` | Optional actor covariate(s) (length-n vector or n*p matrix). |

| | |
|---|---|
| contagion | "out" (undirected neighbour count, default) or "total" (adds in-neighbours for a directed network). |
| coef0 | Optional starting parameters (default: the MPLE). |
| nsim | Chains simulated per iteration (must exceed the parameter count). |
| sweeps | Gibbs sweeps per chain (default scales with n). |
| maxit, tol, damp | Outer iterations, convergence tolerance on the standardized deviations, and Hummel-style step damping. |
| seed | Integer RNG seed. |
| verbose | Print the per-iteration trajectory. |

## Value

An alaam_mcmle object with coef, se (from the inverse Fisher information), tstat, iterations, converged, sim_mean and obs.

## Examples

```
set.seed(1)
net <- matrix(as.integer(runif(30 * 30) < 0.1), 30, 30)
net <- ((net + t(net)) > 0) * 1L
diag(net) <- 0L
y <- as.integer(runif(30) < 0.4)
fit <- alaam_mcmle(y, net, nsim = 300L, maxit = 20L)
fit
```

---

# alaam_mple *Autologistic actor attribute model by maximum pseudo-likelihood (ALAAM)*

---

## Description

Fits an ALAAM (Robins, Pattison & Elliott 2001) – a model for a binary actor attribute at a fixed, exogenous network – by maximum pseudo-likelihood, i.e. a logistic regression of the attribute on the native contagion change statistic (the number of network neighbours that also carry the attribute, a masked popcount) plus any actor covariates. The dependent variable is the attribute, not a tie; the network enters through the same bit-kernel primitive applied to (network-row AND attribute-vector). Best suited to an undirected network, where the contagion is the neighbour count with y = 1. Host C++ (no Python; works in the CPU-only build).

## Usage

```
alaam_mple(y, net, cov = NULL, contagion = "out", directed = FALSE)
```

### Arguments

| | |
|---|---|
| y | Binary actor attribute (0/1), length n – the dependent variable. |
| net | Integer/logical n*n adjacency of the fixed (exogenous) network. |
| cov | Optional actor covariate(s): a length-n vector or an n*p matrix, entered as change statistics (the covariate value at each actor). |
| contagion | Which contagion change statistic to include: "out" (out-neighbours with y = 1, the default and the neighbour count when undirected), "in", "both", or FALSE for none. |
| directed | Logical: is net directed? (documentation only; the contagion columns are selected by contagion). |

### Value

An alaam_mple object with coef, se, z, pval (from the logistic regression), glm (the fitted model) and n.

### Examples

```
set.seed(1)
net <- matrix(as.integer(runif(30 * 30) < 0.1), 30, 30)
net <- ((net + t(net)) > 0) * 1L        # undirected
diag(net) <- 0L
y <- as.integer(runif(30) < 0.4)
fit <- alaam_mple(y, net)
fit
```

---

## alaam_simulate *Simulate the actor attribute from an ALAAM (Gibbs sampler)*

---

### Description

Draws nsim binary attribute vectors from an ALAAM at a fixed network with the native Gibbs sampler: each full conditional is a logistic regression of y_i on the contagion (the number of neighbours with y = 1, the masked popcount used by alaam_mple, so estimator and simulator share one statistic) plus the actor covariates. Python-free (host engine, CPU-only).

### Usage

```
alaam_simulate(net, coef, cov = NULL, y0 = NULL, contagion = "out",
               nsim = 100L, sweeps = 200L, seed = 1234L)
```

### Arguments

| | |
|---|---|
| net | Integer/logical n*n adjacency of the fixed network. |
| coef | Parameters in the order density (intercept), contagion, then one per covariate column of cov. |
| cov | Optional actor covariate(s) (length-n vector or n*p matrix) matching the covariate coefficients in coef. |
| y0 | Optional start attribute (length n); default all zero. |
| contagion | "out" (default) uses out-neighbours; "total" adds in-neighbours (the joint-consistent contagion for a directed network). |
| nsim | Number of independent chains. |
| sweeps | Gibbs sweeps per chain (burn-in over the attribute vector). |
| seed | Integer RNG seed. |

### Value

A list with y (a nsim x n matrix of simulated attributes) and stats (a nsim x 2 matrix of density and contagion).

### Examples

```
set.seed(1)
net <- matrix(as.integer(runif(30 * 30) < 0.1), 30, 30)
net <- ((net + t(net)) > 0) * 1L        # undirected
diag(net) <- 0L
sim <- alaam_simulate(net, coef = c(density = -1, contagion = 0.3), nsim = 20L)
colMeans(sim$stats)
```

---

## cusna_abi_version *Native cusna engine: ABI version and CUDA availability*

---

### Description

cusna_abi_version() returns the C ABI version compiled into the package. cusna_has_cuda() returns TRUE only for a build that links the CUDA backend of the engine and finds a usable device; the R package builds the CPU backend, so it returns FALSE.

### Usage

```
cusna_abi_version()

cusna_has_cuda()
```

### Value

An integer (cusna_abi_version) or logical (cusna_has_cuda).

## Examples

```
cusna_abi_version()
cusna_has_cuda()
```

---

`cusna_behavior_stats` *Host behavior evaluation statistics*

---

## Description

Wrapper over the native `cusna_behavior_stats`, a port of `behavior_stats` (RSiena convention 9: totSim subtracts nb*simMean, avSim divides by nb, avAlt uses the full out-degree).

## Usage

```
cusna_behavior_stats(z, bty, xnet, bkdist, zbar, brange, bsimmean)
```

## Arguments

| | |
|---|---|
| `z` | Integer M*n matrix of end behavior values (rows = chains). |
| `bty` | Integer vector (kb) of behavior effect type ids (0 linear, 1 quad, 2 avAlt, 3 avSim, 4 totSim). |
| `xnet` | Integer/logical n*n predictor network (start-of-period, less missing ties). |
| `bkdist` | Integer vector (n): 1 if the actor counts (kmask_dist). |
| `zbar`, `brange`, `bsimmean` | Numeric: overall mean, range and simMean. |

## Value

Numeric M*kb matrix (rows = chains).

---

`cusna_effect` *Specify SAOM model effects*

---

## Description

Constructors for the terms of a stochastic actor-oriented model. They return lightweight R objects; `mom_estimate` translates them into the native engine's effect descriptors at fit time.

## Usage

```
cusna_effect(name, covariate = NULL, parameter = NULL, dyn = FALSE,
             type = c("eval", "creation", "endow"), net_ref = NULL)

cusna_beh_effect(name)

cusna_rate_effect(name, covariate = NULL)

cusna_interaction(components)
```

## Arguments

| | |
|---|---|
| name | effect name. For cusna_effect, a structural effect (e.g. "density", "recip", "transTrip", the gwesp/gwdsp families), a covariate effect ("egoX", "altX", "simX", "sameX", ...), or a dyadic-covariate effect ("X"). For cusna_beh_effect, one of "linear", "quad", "avAlt", "avSim", "totSim". For cusna_rate_effect, one of "RateX", "outRate", "inRate". |
| covariate | covariate values. Actor covariate: a numeric vector of length n (constant) or an n by (waves) matrix (changing). Dyadic covariate: an n by n matrix, or n by n by (periods) for a changing one. Centered by the engine. |
| parameter | optional internal effect parameter (RSiena's parameter); NULL uses the effect's default. The engine truncates this to an integer for gwesp/gwdsp/threshold effects, matching RSiena. |
| dyn | logical; if TRUE the covariate is the co-evolving dependent behavior (choice dynamics use the current values, statistics use the start-of-period values). |
| type | effect type: "eval" (default), "creation", or "endow" (endowment). |
| net_ref | for cross-network effects in multi-network models (see mom_estimate_multinet), the 1-based index of the source network; NULL otherwise. |
| components | for cusna_interaction, a length-2 or length-3 vector of 1-based positions of the component effects in the effects list. The components must appear earlier in that list. |

## Value

A lightweight specification object of class cusna_effect, cusna_beh_effect, cusna_rate_effect, or cusna_interaction, to be passed to mom_estimate.

## See Also

mom_estimate, saom_data.

## Examples

```
effects <- list(
  cusna_effect("density"),
  cusna_effect("recip"),
  cusna_effect("transTrip")
)

# A covariate effect and a user interaction of effects 1 and 2.
alcohol <- runif(50)
list(
  cusna_effect("egoX", covariate = alcohol),
  cusna_effect("altX", covariate = alcohol),
  cusna_interaction(c(1, 2))
)

# Behavior and rate effects.
cusna_beh_effect("linear")
cusna_rate_effect("outRate")
```

cusna_fit *Fitted SAOM model*

## Description

The object returned by mom_estimate and mom_estimate_multinet, with the usual R accessors.

## Usage

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

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

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

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

## S3 method for class 'cusna_fit'
as.data.frame(x, ...)
```

## Arguments

| | |
|---|---|
| x, object | a cusna_fit object. |
| ... | ignored. |

## Value

coef returns the named parameter estimates; vcov the delta-method covariance matrix (whose diagonal squares to the reported standard errors); as.data.frame a data frame with columns effect, estimate, se, and t_conv. print and summary report the fit and return the object invisibly.

A cusna_fit is a list with elements coefficients, se, tconv, estimates (data frame), rates (conditional rate estimates, or NULL), tconv_max, n_sims, wall_time, backend, conditional, targets, sim_means, D (derivative matrix), msf (moment covariance), and sims / sims_behavior: the phase-3 end networks (one n3 x n x n integer array per period) and, for co-evolution, end behavior (one n3 x n matrix per period) when the fit was made with return_nets = TRUE, otherwise NULL.

## See Also

mom_estimate.

## Examples

```
# Conditional method-of-moments fit of the s50 panel (short schedule).
fit <- mom_estimate(
  saom_data(s50$waves),
  effects = list(cusna_effect("density"), cusna_effect("recip")),
  conditional = TRUE,
  control = mom_control(n1 = 200, nsub = 1, n2 = 15, batch2 = 50,
                        nD = 200, n3 = 300, seed = 2))
print(fit)
summary(fit)
coef(fit)
vcov(fit)
as.data.frame(fit)
```

---

# cusna_fran *Native simulation backend for RSiena's siena07()*

---

## Description

Returns a FRAN closure that plugs the native SAOM simulator into the unmodified `RSiena::siena07()` estimator. RSiena keeps its full Robbins–Monro machinery (and hence its convergence behavior); only the network simulation runs on the compiled engine. The full bridge (mapping an arbitrary `siena07` effects object onto the native descriptors) is under development; this closure covers the structural and single-covariate effects named in `effect_names`.

## Usage

```
cusna_fran(waves, effect_names, covariate = NULL, conditional = TRUE)
```

## Arguments

| | |
|---|---|
| `waves` | a list of adjacency matrices (0/1), one per wave, matching the `RSiena` data object. |
| `effect_names` | character vector of the included effect names, in the same order as the `RSiena` effects object rows (after the basic rates). |
| `covariate` | optional actor-covariate values (numeric vector) used by the covariate effects (`egoX`/`altX`/`simX`/`sameX`/...). |
| `conditional` | logical; must match the `cond` setting of the `RSiena` algorithm. |

## Details

The simulation seeds are drawn from R's random number stream, which `siena07()` initialises from the algorithm's `seed`, so a run with a fixed seed is reproducible. The closure checks that the parameter vector passed by `siena07()` matches `effect_names` and `conditional` and stops with an explanatory error otherwise. Simulated networks are not returned to **RSiena** (`returnDeps` = `TRUE` has no effect); use `mom_estimate` with `return_nets` = `TRUE` for goodness-of-fit work on the native engine.

## Value

A function suitable for assignment to `alg$FRAN`. Each call simulates one chain per period at the current `theta` and returns, following **RSiena**'s `simstats0c` contract, a list with `fra` and `sc` (periods-by-parameters matrices of statistics and scores), `ntim0` (one duration per period), `OK` and `feasible`. The number of calls made so far is kept in `attr(fran, "counter")$i`.

## See Also

`mom_estimate` for the standalone native estimator.

## Examples

```
if (requireNamespace("RSiena", quietly = TRUE)) {
  # The s50 panel as an RSiena data object ...
  net <- RSiena::sienaDependent(array(unlist(s50$waves), dim = c(50, 50, 3)))
  dat <- RSiena::sienaDataCreate(net)
  eff <- RSiena::includeEffects(RSiena::getEffects(dat), "transTrip",
                                character = TRUE)
  # ... estimated by RSiena's own siena07(), simulating on the native engine.
  alg <- RSiena::sienaAlgorithmCreate(projname = NULL, nsub = 2, n3 = 200,
                                      seed = 123)
  alg$FRAN <- cusna_fran(s50$waves, c("density", "recip", "transTrip"))
  ans <- RSiena::siena07(alg, data = dat, effects = eff, batch = TRUE,
                         verbose = FALSE, silent = TRUE, useCluster = FALSE)
  ans
  attr(alg$FRAN, "counter")$i   # number of simulation calls
}
```

---

# cusna_gof_distribution

*sienaGOF auxiliary distributions*

---

## Description

Wrapper over the native `cusna_gof_distribution` (port of `gof.py`): cumulative indegree/outdegree actor counts, or geodesic-distance ordered-pair counts for a batch of simulated end networks.

## Usage

```
cusna_gof_distribution(a, kind, levls)
```

## Arguments

| | |
|---|---|
| `a` | Integer M*n*n array of end networks (first dim = chains). |
| `kind` | Integer: 0 indegree, 1 outdegree, 2 geodesic. |
| `levls` | Integer vector of cut levels. |

## Value

Numeric matrix with `M` rows and `length(levls)` columns (kind 0/1) or `length(levls) + 1` columns (kind 2, the extra column is the unreachable pair count).

## Examples

```
# Cumulative indegree counts and geodesic-distance pair counts of the three
# observed s50 waves, treated as a batch of three networks.
a <- aperm(simplify2array(s50$waves), c(3, 1, 2))   # 3 x 50 x 50
cusna_gof_distribution(a, kind = 0L, levls = 0:6)
cusna_gof_distribution(a, kind = 2L, levls = 1:5)
```

---

# `cusna_network_stats` *Host network evaluation statistics (RSiena conventions)*

---

## Description

Thin R wrapper over the native `cusna_network_stats`, a bit-exact port of the reference `network_stats_ex`. On public datasets these statistics reproduce the RSiena targets to machine precision.

## Usage

```
cusna_network_stats(a, type_ids, p1, p2, vmat, vmiss, dyn_flags, vb,
                    od0, id0, wtab, symflag = 0L)
```

## Arguments

| | |
|---|---|
| `a` | Integer/logical n*n adjacency matrix (the masked statistic network `(A & keep) \| force`), 0/1, zero diagonal. |
| `type_ids` | Integer vector (k) of effect type ids. |
| `p1`, `p2` | Numeric vectors (k) of effect parameters 1 and 2. |
| `vmat` | Numeric k*n matrix of per-effect covariate values (rows = effects). |
| `vmiss` | Integer/logical k*n matrix of covariate missing flags. |
| `dyn_flags` | Integer vector (k): 1 if the effect covariate is behavior. |
| `vb` | Numeric vector (n): centered start behavior (for dyn effects). |
| `od0`, `id0` | Integer vectors (n): start out/in degrees. |
| `wtab` | Numeric k*(n+1) matrix of gwesp/gwdsp weight tables (rows = effects). |
| `symflag` | Integer: 1 for symmetric-network conventions (density halved). |

## Value

Numeric vector of length `k`.

## cusna_set_threads *Set the OpenMP thread count for the native CPU backend*

### Description

Explicitly caps the number of threads the CPU backend uses for its OpenMP-parallel batch loops (SAOM simulation, ERGM sampling and MPLE). Unlike `Sys.setenv(OMP_NUM_THREADS = ...)`, this takes effect immediately via the OpenMP runtime API and does not depend on when the environment variable would otherwise be read. Has no effect on a build without OpenMP (e.g. the default macOS toolchain without `libomp`).

### Usage

```
cusna_set_threads(n)
```

### Arguments

| | |
|---|---|
| `n` | Requested thread count (coerced to an integer, at least 1). |

### Value

Invisibly, the previous maximum thread count (`0L` if OpenMP is not compiled in).

### Examples

```
old <- cusna_set_threads(1L)
cusna_set_threads(old)
```

## ergm_mcmle *Maximum likelihood estimate of an ERGM (MCMC-MLE)*

### Description

Fits an ERGM by MCMC maximum likelihood: an external moment-matching loop over the native TNT sampler. For the canonical exponential family the Fisher information is the covariance of the sufficient statistics, so each iteration simulates from the current parameters, forms the mean and covariance of the sufficient statistics, and takes a damped Newton step until the observed statistics are matched. The loop is pure R over the native engine (no Python). This closes the sampler + MPLE -> maximum likelihood gap; full MCMC-MLE for arbitrary specifications remains the domain of the `ergm` package.

### Usage

```
ergm_mcmle(x, terms, attr = NULL, directed = FALSE, coef0 = NULL,
           nsim = 500L, burnin = NULL, interval = 1024L, maxit = 25L,
           tol = 0.1, damp = 0.5, seed = 1234L, verbose = FALSE)
```

### Arguments

| | |
|---|---|
| x | Integer/logical n*n observed adjacency (0/1, zero diagonal). |
| terms | A non-empty list of ergm_term objects. |
| attr | Optional numeric node covariate of length n; NULL for none. |
| directed | Logical: is the network directed? |
| coef0 | Optional starting parameters (default: edges from density, rest 0). |
| nsim | Networks simulated per iteration (must exceed the number of terms). |
| burnin, interval | MCMC schedule per iteration; chains start from the observed network, so a modest burn-in decorrelates near the MLE. Default burn-in scales with the dyad count. |
| maxit | Maximum outer iterations. |
| tol | Convergence: stop when every standardized deviation \|t\| is below it. |
| damp | Step-length damping (Hummel-style partial stepping when far off). |
| seed | Integer RNG seed. |
| verbose | Print the per-iteration trajectory. |

### Value

An ergm_mcmle object with coef, se (from the inverse Fisher information), tstat, iterations, converged, sim_mean and obs.

### Examples

```
set.seed(1)
a <- matrix(as.integer(runif(18 * 18) < 0.3), 18, 18)
diag(a) <- 0L
fit <- ergm_mcmle(a, list(ergm_term("edges"), ergm_term("mutual")),
                  directed = TRUE, nsim = 200L, seed = 1L)
fit
```

---

## ergm_mple *Maximum pseudo-likelihood estimate (directed edges/mutual/ttriple demo)*

---

### Description

A logistic regression of tie presence on the dyadic change statistics the native engine computes (Strauss & Ikeda 1990). This is the MPLE demo for the three directed terms edges / mutual / transitive-triple; full MCMC-MLE stays with the ergm package (framing M-3).

### Usage

```
ergm_mple(x)
```

### Arguments

| | |
|---|---|
| x | Integer/logical n*n directed adjacency matrix (0/1, zero diagonal). |

### Value

A named numeric vector of pseudo-MLE coefficients (`edges`, `mutual`, `transitive`).

### Examples

```
set.seed(1)
a <- matrix(as.integer(runif(20 * 20) < 0.2), 20, 20)
diag(a) <- 0L
ergm_mple(a)
```

---

## `ergm_simulate` *Simulate networks from an ERGM (TNT sampler)*

---

### Description

Draws `nsim` independent networks with the native TNT Metropolis sampler (the CPU mirror, validated distributionally against `ergm::simulate`) and returns their sufficient statistics. Full MCMC-MLE for ERGMs remains the domain of the `ergm` package; cusna provides the simulator and MPLE.

### Usage

```
ergm_simulate(x, coef, terms, nsim = 100, attr = NULL, directed = FALSE,
              burnin = 16384L, interval = 1024L, seed = 1234L,
              return_nets = FALSE)
```

### Arguments

| | |
|---|---|
| x | Integer/logical n*n start adjacency matrix (0/1, zero diagonal). |
| coef | Numeric vector of natural parameters, one per term. |
| terms | A non-empty list of `ergm_term` objects. |
| nsim | Number of independent chains / simulated networks. |
| attr | Optional numeric node covariate of length n; `NULL` for none. |
| directed | Logical: is the network directed? |
| burnin, interval | Metropolis burn-in and interval; raise the burn-in to ~10x the dyad count for large networks so chains forget the observed start. |
| seed | Integer RNG seed. |
| return_nets | Logical: if `TRUE`, also return the simulated networks. |

## Value

If `return_nets = FALSE` (default) a `nsim` x `length(terms)` matrix of sampled statistics (named columns). If `return_nets = TRUE`, a list with `stats` (that matrix) and `nets` (a list of `nsim` n*n integer adjacency matrices, one per simulated network).

## Examples

```
a <- matrix(0L, 6, 6)
a[1, 2] <- a[2, 1] <- a[2, 3] <- a[3, 2] <- 1L
sf <- ergm_simulate(a, coef = -1, terms = list(ergm_term("edges")),
                    nsim = 8, burnin = 2000L, seed = 1L)
colMeans(sf)
```

---

# ergm_stats *Observed ERGM statistics of a network*

---

## Description

Native host statistics for the ERGM term list, a bit-exact port of the reference `network_stats_ergm` (validated to machine zero in `native/test`).

## Usage

```
ergm_stats(x, terms, attr = NULL, directed = FALSE)
```

## Arguments

| | |
|---|---|
| `x` | Integer/logical n*n adjacency matrix (0/1, zero diagonal). |
| `terms` | A non-empty list of `ergm_term` objects. |
| `attr` | Optional numeric node covariate of length n (for nodecov/nodematch/absdiff); `NULL` for none. |
| `directed` | Logical: is the network directed (enables `mutual`)? |

## Value

A named numeric vector of length `length(terms)`.

## Examples

```
a <- matrix(0L, 4, 4)
a[1, 2] <- a[2, 1] <- a[2, 3] <- a[3, 2] <- 1L
ergm_stats(a, list(ergm_term("edges")))
```

## ergm_term *Construct an ERGM term*

### Description

Construct an ERGM term for use with `ergm_stats` and `ergm_simulate`.

### Usage

```
ergm_term(name, parameter = 0)
```

### Arguments

| | |
|---|---|
| `name` | A single ERGM term name: one of `edges`, `mutual`, `nodecov`, `nodematch`, `absdiff`, `gwesp`, `gwdsp`, or the temporal (tergm-only) terms `memory` (lagged tie) and `delrecip` (lagged reciprocity). |
| `parameter` | Numeric term parameter (e.g. the gwesp/gwdsp decay); ignored by terms that take none. |

### Value

An `ergm_term` object.

### Examples

```
ergm_term("edges")
ergm_term("gwesp", 0.25)
```

## mom_estimate *Method-of-Moments estimation of a SAOM*

### Description

Fits a stochastic actor-oriented model by the RSiena-style method of moments (Robbins–Monro), driving the compiled native simulator – no Python. The moment targets and per-period masks reproduce the reference implementation bit-for-bit; the estimates agree with the reference estimator within simulation standard errors.

### Usage

```
mom_estimate(data, effects, beh_effects = NULL, rate_effects = NULL,
             conditional = FALSE, backend = c("cpu", "gpu"), maxdegree = 0,
             control = mom_control(), verbose = FALSE, return_nets = FALSE)

mom_control(firstg = 0.4, nsub = 4, n2 = c(40, 40, 60, 80),
            batch2 = 256, n1 = 1000, n3 = 4000, nD = 1000,
            diagonalize = 0.2, seed = 1234)
```

## Arguments

| | |
|---|---|
| data | a cusna_data object from saom_data. |
| effects | a non-empty list of cusna_effect objects (network evaluation/creation/endowment effects) and, optionally, cusna_interaction objects placed after their components. |
| beh_effects | optional list of cusna_beh_effect objects for network–behavior co-evolution (requires behavior in the data; forces unconditional estimation). |
| rate_effects | optional list of cusna_rate_effect objects (outRate/inRate/RateX); unconditional only. |
| conditional | logical; FALSE (default) estimates the basic rate parameters jointly with the evaluation effects. TRUE conditions each period on the observed distance (RSiena's default for a single dependent network) and reports the rates separately. |
| backend | "cpu" (default; the from-source CRAN build) or "gpu" (needs a CUDA build). |
| maxdegree | optional maximum out-degree constraint (0 = none). |
| control | a list of tuning parameters from mom_control. |
| verbose | logical; if TRUE, print per-phase progress. |
| return_nets | logical; if TRUE, keep the end networks of the phase-3 simulations (RSiena's returnDeps) in the returned fit as sims: a list with one integer array of dimension c(n3, n, n) per period (chains first, the layout cusna_gof_distribution takes). Co-evolution fits also return the simulated behavior as sims_behavior (one n3 x n matrix per period). The arrays hold n3 * n^2 integers per period, so choose n3 accordingly. |
| firstg | initial Robbins–Monro gain. |
| nsub | number of phase-2 subphases. |
| n2 | integer vector (length >= nsub) of phase-2 iteration counts per subphase. |
| batch2 | simulation batch size within phase-2 iterations. |
| n1 | number of phase-1 simulations (derivative estimation). |
| n3 | number of phase-3 simulations (standard errors and convergence). |
| nD | simulations used to re-estimate the derivative each subphase. |
| diagonalize | shrinkage of the derivative matrix towards its diagonal, in [0, 1]; higher is more robust on ill-conditioned models. |
| seed | random seed for the simulation stream. |

## Details

For nearly collinear specifications the Robbins–Monro estimator may not converge from a cold start; increase diagonalize or the phase counts, or run **RSiena**'s own estimator on the native simulator with cusna_fran.

The default schedule (65,320 simulated panels) is deliberately larger than **RSiena**'s default, which gives phase-3 standard errors and convergence ratios with less Monte Carlo error; on the CPU build a smaller schedule (for example n3 = 1000) finishes correspondingly faster. The estimator drives the CPU backend of the engine; backend = "gpu" is reserved for a build that links the CUDA backend.

The simulated end networks returned with return_nets = TRUE are the states at the end of each period (in conditional estimation, when the observed distance is reached), before the statistic masks for missing, structural and composition-change ties are applied.

## Value

`mom_estimate` returns a `cusna_fit` object. `mom_control` returns a list of tuning parameters (class `cusna_control`).

## See Also

`cusna_fit`, `cusna_effect`, `saom_data`, `mom_estimate_multinet`, `cusna_fran`.

## Examples

```
# The s50 panel: three waves of friendship among 50 girls.
dat <- saom_data(s50$waves)
effects <- list(cusna_effect("density"), cusna_effect("recip"),
                cusna_effect("transTrip"))

# A short schedule keeps the example quick; use the defaults in practice.
ctl <- mom_control(n1 = 200, nsub = 2, n2 = c(10, 10), batch2 = 50,
                   nD = 200, n3 = 300, seed = 1)
fit <- mom_estimate(dat, effects, conditional = TRUE, control = ctl,
                    return_nets = TRUE)
summary(fit)

# The phase-3 networks: indegree distribution of the simulated third wave.
dim(fit$sims[[2]])
indeg <- cusna_gof_distribution(fit$sims[[2]], kind = 0L, levls = 0:6)
colMeans(indeg)
```

---

# s50 — *Friendship and alcohol use of 50 girls (Teenage Friends and Lifestyle Study)*

---

## Description

A three-wave panel of friendship nominations among 50 randomly chosen girls from the Teenage Friends and Lifestyle Study, with their alcohol use. It is the standard small panel for stochastic actor-oriented models (no missing values, quick to fit) and is also distributed with **RSiena** as `s501`, `s502`, `s503` and `s50a`; the examples and vignettes of this package use it.

## Usage

```
s50
```

## Format

A list with two elements:

`waves` a list of three $50 \times 50$ integer adjacency matrices (0/1, zero diagonal), the friendship networks at time points 1–3; entry `[i, j]` is 1 when girl `i` names girl `j` as a friend.

`alcohol` a $50 \times 3$ integer matrix of alcohol use at the three time points, coded 1–5 (higher values indicate higher consumption).

## Source

https://www.stats.ox.ac.uk/~snijders/siena/s50_data.htm; identical to the **RSiena** datasets `s501`, `s502`, `s503` and `s50a`.

## References

West, P. and Sweeting, H. (1995). *Background Rationale and Design of the West of Scotland 11-16 Study*. Working Paper No. 52, MRC Medical Sociology Unit Glasgow.

## Examples

```
str(s50)
dat <- saom_data(s50$waves)
dat
# ties per wave
vapply(s50$waves, sum, numeric(1))
```

---

# saom_data *Create a SAOM data panel*

---

## Description

Builds a longitudinal network panel for estimation, applying RSiena's imputation and per-period masks (first-wave missings to 0, later waves carried forward; structural values fixed; composition-change gating) in native R code – no Python. The construction is validated bit-for-bit against the reference implementation on public datasets.

## Usage

```
saom_data(waves, behavior = NULL, composition = NULL, cc_option = 1)
```

## Arguments

| | |
|---|---|
| `waves` | a list of at least two square adjacency matrices, one per observation wave, all on the same actor set. Missing ties are `NA` (or the code 9); structural zeros and ones are the codes 10 and 11. |
| `behavior` | optional co-evolving dependent behavior: a numeric matrix with one row per actor and one column per wave (`NA` for missing). |
| `composition` | optional composition change (joiners/leavers): a list with one entry per actor, each a numeric vector `c(start, end)` (or several stacked pairs) of 1-based wave numbers the actor is present for. Estimation is then unconditional, as in RSiena. |
| `cc_option` | the `sienaCompositionChange` option (1–3) controlling how absent actors' entries are treated. |

## Value

An object of class `cusna_data` with fields `n` (actors), `n_waves`, and `n_periods`, carrying the prepared engine data in `$internal`.

## See Also

`mom_estimate`, `cusna_effect`, `saom_multinet_data`.

## Examples

```
dat <- saom_data(s50$waves)
dat

# with the co-evolving alcohol use as a dependent behavior
saom_data(s50$waves, behavior = s50$alcohol)
```

---

# saom_multinet_data *Multi-network co-evolution*

---

## Description

Several dependent networks that co-evolve over one actor set, with cross-network effects. `saom_multinet_data` builds the panel and `mom_estimate_multinet` fits it by the method of moments on the native multi-network simulator (unconditional only, as in RSiena for more than one dependent variable).

## Usage

```
saom_multinet_data(waves_list, composition = NULL, cc_option = 1)

mom_estimate_multinet(data, effects_by_net, backend = c("cpu", "gpu"),
                      control = mom_control(), verbose = FALSE)
```

## Arguments

| | |
|---|---|
| `waves_list` | a list with one entry per dependent network; each entry is itself a list of wave matrices (as in `saom_data`). All networks share the actor set and the number of waves. |
| `composition` | optional composition change applied to the shared actor set (see `saom_data`). |
| `cc_option` | the composition-change option (1–3). |
| `data` | a `cusna_multinet_data` object. |
| `effects_by_net` | a list with one effects list per network. Within-network effects are ordinary `cusna_effect` objects; cross-network effects (`"crprod"`, `"crprodRecip"`, `"crprodMutual"`, `"from"`, `"to"`) additionally take `net_ref`, the 1-based index of the source network. |
| `backend` | `"cpu"` (default) or `"gpu"`. |
| `control` | tuning parameters from `mom_control`. |
| `verbose` | logical; print per-phase progress. |

## Details

Cross-network statistics read the other network at its period-start state; the dynamics read the current co-evolving state. Multi-network models do not support behavior co-evolution, interactions, or degree bounds.

## Value

`saom_multinet_data` returns a `cusna_multinet_data` object. `mom_estimate_multinet` returns a `cusna_fit`; its parameters are laid out per network (rates then effects), following RSiena's convention.

## See Also

`mom_estimate`, `cusna_effect`.

## Examples

```
# Friendship (the s50 waves) co-evolving with a second, synthetic network on
# the same 50 actors.
set.seed(11)
second <- lapply(s50$waves, function(w) {
  m <- w * (matrix(runif(2500), 50, 50) < 0.5); diag(m) <- 0L; m
})
dat <- saom_multinet_data(list(s50$waves, second))
dat

# A short schedule keeps the example quick; use the defaults in practice.
fit <- mom_estimate_multinet(dat, effects_by_net = list(
  list(cusna_effect("density"), cusna_effect("recip")),
  list(cusna_effect("density"), cusna_effect("crprod", net_ref = 1))
), control = mom_control(n1 = 150, nsub = 1, n2 = 10, batch2 = 40,
                         nD = 150, n3 = 200, seed = 5))
summary(fit)
```

---

# saom_simulate *Simulate a SAOM at given parameters*

---

## Description

Forward-simulates every period of the panel, starting each period from the observed wave, at the parameter vector `theta` on the native engine. The statistics are the moment statistics that `mom_estimate` matches, so a simulation at the estimated parameters reproduces the targets on average; the end networks serve goodness-of-fit checks and counterfactual scenarios. This is the SAOM counterpart of `ergm_simulate`.

## Usage

```
saom_simulate(data, effects, theta, nsim = 100, beh_effects = NULL,
              conditional = FALSE, maxdegree = 0, seed = NULL,
              return_nets = TRUE)

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

## Arguments

| | |
|---|---|
| data | a cusna_data object from saom_data. |
| effects | a non-empty list of cusna_effect (and cusna_interaction) objects, as for mom_estimate. |
| theta | the parameter vector, in the layout of coef() of the matching fit (unconditional: the basic rates per period, then the effects; conditional: the effects only), or a cusna_fit. |
| nsim | number of independent chains simulated for each period. |
| beh_effects | optional list of cusna_beh_effect objects for network–behavior co-evolution. |
| conditional | logical; if TRUE, each period runs until the observed distance is reached, as in conditional estimation. |
| maxdegree | optional maximum out-degree constraint (0 = none). |
| seed | optional seed; when given, set.seed(seed) is called before the per-period simulation seeds are drawn from R's random number stream. |
| return_nets | logical; return the simulated end networks. |
| x | a cusna_sim object. |
| ... | ignored. |

## Details

Chains are independent and use counter-based random numbers, so a simulation with a given seed is identical for any number of threads (cusna_set_threads).

## Value

A cusna_sim list with elements stats (an nsim-by-$p$ matrix of the moment statistics, summed over periods, with the observed targets alongside), theta, durations (conditional simulation: an nsim-by-periods matrix of the simulated durations, whose means estimate the rates), nets (when return_nets: one nsim x n x n integer array of end networks per period, chains first), behavior (co-evolution: one nsim x n matrix of end behavior per period), nsim, n_periods and conditional. The print method compares the simulated means of the statistics with the observed targets.

## See Also

mom_estimate (whose return_nets keeps the phase-3 networks of an estimation run), cusna_gof_distribution.

## Examples

```
dat <- saom_data(s50$waves)
effects <- list(cusna_effect("density"), cusna_effect("recip"),
                cusna_effect("transTrip"))

# Conditional simulation at parameter values close to the s50 estimates.
sim <- saom_simulate(dat, effects, theta = c(-2.7, 2.4, 0.63), nsim = 200,
                     conditional = TRUE, seed = 1)
sim
colMeans(sim$durations)          # simulated durations ~ the rate estimates

# Outdegree distribution of the simulated third wave
outdeg <- cusna_gof_distribution(sim$nets[[2]], kind = 1L, levls = 0:6)
round(colMeans(outdeg), 1)
```

---

`stergm_cmle` *Separable temporal ERGM by conditional MLE (STERGM CMLE)*

---

## Description

Fits a STERGM (Krivitsky & Handcock 2014) to a network sequence by conditional maximum likelihood. Each transition separates into two conditionally independent ERGMs on discord-constrained networks: a formation model on the union (only dyads absent at t-1 are free to toggle) and a dissolution model on the intersection (only dyads present at t-1 are free), the latter in the persistence parameterisation (positive coefficients favour tie persistence, matching `tergm`'s `Persist()`). The CMLE is two ERGM MCMC-MLE fits (a reuse of `ergm_mcmle`) with the native TNT sampler restricted to the free dyads; statistics are pooled across transitions. Python-free (CPU engine).

## Usage

```
stergm_cmle(nets, formation, dissolution = formation, attr = NULL,
            directed = TRUE, nsim = 500L, burnin = NULL, interval = 1024L,
            maxit = 30L, tol = 0.1, damp = 0.5, seed = 1234L, verbose = FALSE)
```

## Arguments

| | |
|---|---|
| `nets` | A list of >= 2 n*n adjacency matrices (0/1, zero diagonal), in time order. |
| `formation` | A non-empty list of [ergm_term()] objects for the formation model. |
| `dissolution` | A list of [ergm_term()] objects for the dissolution (persistence) model; defaults to the formation terms. |
| `attr` | Optional numeric node covariate of length n (for nodecov/nodematch/absdiff); `NULL` for none. |
| `directed` | Logical: are the networks directed? |
| `nsim` | Networks simulated per iteration per transition. |
| `burnin, interval` | MCMC schedule (default burn-in scales with the dyad count). |

| | |
|---|---|
| `maxit`, `tol`, `damp` | Outer iterations, convergence tolerance and step damping. |
| `seed` | Integer RNG seed. |
| `verbose` | Print the per-iteration trajectory. |

## Value

A `stergm_cmle` object with `formation` and `dissolution` sub-fits (each a list with `coef`, `se`, `tstat`, `iterations`, `converged`, `sim_mean`, `obs`) and `transitions`.

## Examples

```
set.seed(1)
base <- matrix(as.integer(runif(20 * 20) < 0.2), 20, 20)
diag(base) <- 0L
nets <- list(base)
for (t in 2:3) {
  nx <- base
  flip <- matrix(runif(20 * 20) < 0.1, 20, 20)
  nx[flip] <- 1L - nx[flip]
  diag(nx) <- 0L
  nets[[t]] <- nx
  base <- nx
}
fit <- stergm_cmle(nets, formation = list(ergm_term("edges"), ergm_term("mutual")),
                   nsim = 200L, maxit = 20L)
fit
```

---

# `stergm_simulate` *Simulate a STERGM transition (constrained draw of the next wave)*

---

## Description

Draws `nsim` networks from a separable temporal ERGM (Krivitsky & Handcock 2014) given the previous wave, the simulation counterpart of `stergm_cmle`. The transition separates into two conditionally independent constrained ERGMs, sampled with the native masked TNT sampler on exactly the discord-constrained space the fit uses: a formation draw $F \sim$ ERGM(`coef_formation`) on $\{y \supseteq y_{lag}\}$ (only dyads absent at t-1 toggle) and a dissolution draw $D \sim$ ERGM(`coef_dissolution`) on $\{y \subseteq y_{lag}\}$ (persistence parameterisation, only dyads present at t-1 toggle). The two draws are independent (separate seed streams, `seed` and `seed` + `500000L`, as in the fit) and composed into $y^t = (F \setminus y_{lag}) \cup D$: newly formed ties where none existed, plus surviving ties where one did. Python-free (CPU engine).

## Usage

```
stergm_simulate(y_lag, formation, dissolution = formation,
                coef_formation, coef_dissolution, attr = NULL,
                directed = TRUE, nsim = 100, burnin = NULL,
                interval = 1024L, seed = 1234L)
```

### Arguments

| | |
|---|---|
| y_lag | Integer/logical n*n lagged network $Y^{t-1}$ (0/1, zero diagonal). |
| formation | A non-empty list of [ergm_term()] objects for the formation model. |
| dissolution | A list of [ergm_term()] objects for the dissolution (persistence) model; defaults to the formation terms. |
| coef_formation | Numeric formation parameters, one per formation term. |
| coef_dissolution | Numeric dissolution parameters, one per dissolution term. |
| attr | Optional numeric node covariate of length n (for nodecov/nodematch/absdiff); NULL for none. |
| directed | Logical: are the networks directed? |
| nsim | Number of independent transition draws. |
| burnin, interval | MCMC schedule per chain (default burn-in scales with the dyad count). |
| seed | Integer RNG seed (dissolution uses seed + 500000L). |

### Value

A list with stats_formation (a nsim x k matrix of the formation sufficient statistics on the union space), stats_dissolution (the same for the dissolution/persistence space) and nets (a list of nsim n*n integer adjacency matrices, the composed $Y^t$).

### Examples

```
set.seed(1)
y_lag <- matrix(as.integer(runif(20 * 20) < 0.2), 20, 20)
diag(y_lag) <- 0L
sim <- stergm_simulate(y_lag,
                       formation = list(ergm_term("edges"), ergm_term("mutual")),
                       coef_formation = c(-2, 0.5),
                       coef_dissolution = c(1.5, 0.2),
                       nsim = 10, burnin = 2000L)
sim$nets[[1]]
```

---

## tergm_mple *Temporal ERGM by bootstrap pseudo-likelihood (btergm style)*

---

### Description

Fits a temporal ERGM to a sequence of directed networks by pooled maximum pseudo-likelihood – the btergm estimator (Desmarais & Cranmer 2012). For every transition the native engine builds the 0->1 change statistics of every dyad of the current network (edges / mutual / transitive triple) and the temporal terms are read off the lagged network (memory = lagged tie, delrecip = lagged reciprocity); the rows are pooled across transitions and fit by weighted logistic regression, with bootstrap confidence intervals. The design matrix is host C++ (no Python; works in the CPU-only build).

### Usage

```
tergm_mple(nets, mutual = TRUE, ttriple = TRUE, memory = TRUE,
           delrecip = TRUE, R = 200L, level = 0.95, seed = 1234L)
```

### Arguments

| | |
|---|---|
| nets | A list of >= 2 directed n*n adjacency matrices (0/1, zero diagonal), in time order. |
| mutual, ttriple | Include the cross-sectional mutual / transitive-triple terms (from the native change-statistic matrix). |
| memory, delrecip | Include the temporal terms (lagged tie / lagged reciprocity). |
| R | Bootstrap replications for the confidence intervals. |
| level | Confidence level for the bootstrap intervals. |
| seed | Integer RNG seed for the bootstrap. |

### Value

A tergm_mple object with coef, ci (bootstrap interval), boot (replicate coefficients), nobs (pooled dyads) and transitions.

### Examples

```
set.seed(1)
nets <- lapply(1:3, function(t) {
  m <- matrix(as.integer(runif(20 * 20) < 0.2), 20, 20)
  diag(m) <- 0L
  m
})
fit <- tergm_mple(nets, R = 50L)
fit
```

---

## tergm_simulate *Simulate from a temporal ERGM (TNT sampler with a lagged network)*

---

### Description

Draws nsim directed networks from the conditional ERGM of the current wave given the previous one, with the native CPU TNT sampler. The term list may mix the cross-sectional ERGM terms (edges, mutual, nodecov, nodematch, absdiff) with the temporal terms memory (lagged tie) and delrecip (lagged reciprocity). Python-free (CPU engine). The pooled estimator is tergm_mple.

### Usage

```
tergm_simulate(x_start, lag, coef, terms = NULL, attr = NULL,
               mutual = TRUE, memory = TRUE, delrecip = TRUE,
               nsim = 100, burnin = NULL, seed = 1234L,
               return_nets = FALSE)
```

## Arguments

| | |
|---|---|
| x_start | Integer/logical n*n start adjacency (0/1, zero diagonal), the chain's starting state. |
| lag | Integer/logical n*n lagged network (the previous wave). |
| coef | Numeric parameters, one per active term (matching terms, or the legacy order edges, [mutual], [memory], [delrecip] when terms is NULL). |
| terms | A list of ergm_term objects (may include the temporal terms memory / delrecip). If NULL (default) the term list is assembled from the deprecated logical flags for backward compatibility. |
| attr | Optional numeric node covariate of length n (for the nodecov / nodematch / absdiff cross-sectional terms); NULL for none. |
| mutual, memory, delrecip | Deprecated: include the corresponding term. Used only when terms is NULL; the assembled order is edges, mutual, memory, delrecip. |
| nsim | Number of independent chains / simulated networks. |
| burnin | MCMC burn-in per chain (default scales with the dyad count). |
| seed | Integer RNG seed. |
| return_nets | Logical: if TRUE, also return the simulated networks. |

## Value

If return_nets = FALSE (default) a nsim x k matrix of sampled statistics (named columns). If return_nets = TRUE, a list with stats (that matrix) and nets (a list of nsim n*n integer adjacency matrices).

## Examples

```
set.seed(1)
lag <- matrix(as.integer(runif(20 * 20) < 0.2), 20, 20); diag(lag) <- 0L
sf <- tergm_simulate(lag, lag, coef = c(-2, 1, 1.5, 0.5), nsim = 20,
                     burnin = 3000L, seed = 1L)
colMeans(sf)
```

# Index


∗ **datasets**
    s50, 19

alaam_mcmle, *3*, 3
alaam_mple, *3*, 4, *5*
alaam_simulate, *3*, 5
as.data.frame.cusna_fit *(*cusna_fit*)*, 9

coef.cusna_fit *(*cusna_fit*)*, 9
cusna *(*cusna-package*)*, 2
cusna-package, 2
cusna_abi_version, 6
cusna_beh_effect, *18*, *23*
cusna_beh_effect *(*cusna_effect*)*, 7
cusna_behavior_stats, *3*, 7
cusna_effect, *3*, 7, *18, 19*, *21–23*
cusna_fit, 9, *19*, *22, 23*
cusna_fran, 10, *18, 19*
cusna_gof_distribution, *3*, 11, *18*, *23*
cusna_has_cuda *(*cusna_abi_version*)*, 6
cusna_interaction, *18*, *23*
cusna_interaction *(*cusna_effect*)*, 7
cusna_network_stats, *3*, 12
cusna_rate_effect, *18*
cusna_rate_effect *(*cusna_effect*)*, 7
cusna_set_threads, *2*, 13, *23*

ergm_mcmle, *3*, 13, *24*
ergm_mple, *3*, 14
ergm_simulate, *3*, 15, *17*, *22*
ergm_stats, *3*, 16, *17*
ergm_term, *14–16*, 17, *28*

mom_control, *21*
mom_control *(*mom_estimate*)*, 17
mom_estimate, *3*, *7–11*, 17, *21–23*
mom_estimate_multinet, *3*, *8, 9*, *19*
mom_estimate_multinet
        *(*saom_multinet_data*)*, 21

print.alaam_mcmle *(*alaam_mcmle*)*, 3
print.alaam_mple *(*alaam_mple*)*, 4
print.cusna_fit *(*cusna_fit*)*, 9
print.cusna_sim *(*saom_simulate*)*, 22
print.ergm_mcmle *(*ergm_mcmle*)*, 13
print.stergm_cmle *(*stergm_cmle*)*, 24
print.tergm_mple *(*tergm_mple*)*, 26

s50, *3*, 19
saom_data, *3*, *8*, *18, 19*, 20, *21*, *23*
saom_multinet_data, *21*, 21
saom_simulate, 22
stergm_cmle, *3*, 24, *25*
stergm_simulate, 25
summary.cusna_fit *(*cusna_fit*)*, 9

tergm_mple, *3*, 26, *27*
tergm_simulate, *3*, 27

vcov.cusna_fit *(*cusna_fit*)*, 9