

Package ‘ergmito’

February 12, 2020

Version 0.2-1

Date 2020-02-10

Title Exponential Random Graph Models for Small Networks

Description Simulation and estimation of Exponential Random Graph Models (ERGMs) for small networks using exact statistics. As a difference from the ‘ergm’ package, ‘ergmito’ circumvents using Markov-Chain Maximum Likelihood Estimator (MC-MLE) and instead uses Maximum Likelihood Estimator (MLE) to fit ERGMs for small networks. As exhaustive enumeration is computationally feasible for small networks, this R package takes advantage of this and provides tools for calculating likelihood functions, and other relevant functions, directly, meaning that in many cases both estimation and simulation of ERGMs for small networks can be faster and more accurate than simulation-based algorithms.

Depends R (>= 3.3.0)

Roxygen list(markdown = TRUE)

RoxygenNote 7.0.2

Encoding UTF-8

Imports ergm,
network,
MASS,
Rcpp,
texreg,
stats,
parallel,
utils,
methods,
graphics

LinkingTo Rcpp, RcppArmadillo

License MIT + file LICENSE

Suggests covr,
sna,
lmtest,
fmcmc,

coda,
knitr,
rmarkdown, tinytest

Collate 'RcppExports.R' 'boot.R' 'count_stats.R' 'data.R' 'ergm_blockdiagonal.R' 'ergmito-checkers.R'
'ergmito-package.R' 'ergmito.R' 'ergmito_surface.R' 'formulae.R' 'gof.R' 'induced_submat.R'
'network.R' 'powerset.R' 'predict.R' 'random.R' 'same_dist.R' 'sim.R' 'simulate.R' 'texreg.R'
'utils.R'

VignetteBuilder knitr

LazyData true

URL <https://muriteams.github.io/ergmito>

BugReports <https://github.com/muriteams/ergmito/issues>

Language en-US

R topics documented:

as_adjmat	3
blockdiagonalize	3
check_support	4
count_stats	6
ergmito	7
ergmito_boot	11
ergmito_formulae	12
ergmito_gof	13
exact_loglik	15
extract.ergmito	17
fivenets	18
geodesic	18
induced_submat	19
matrix_to_network	20
new_ergmito_ptr	22
new_rergmito	23
nvertex	25
plot.ergmito	27
powerset	28
predict.ergmito	29
rbernoulli	30
same_dist	31
simulate.ergmito	32

Index **33**

`as_adjmat`*An alternative to `as.matrix` to retrieve adjacency matrix fast*

Description

This function does not perform significant checks. Furthermore, this function won't keep the row/col names.

Usage

```
as_adjmat(x)

## S3 method for class 'network'
as_adjmat(x)

## S3 method for class 'matrix'
as_adjmat(x)

## S3 method for class 'list'
as_adjmat(x)

## S3 method for class 'formula'
as_adjmat(x)
```

Arguments

`x` An object to be coerced as an adjacency matrix.

`blockdiagonalize`*Block-diagonal models using ergm*

Description

These two functions are used to go back and forth from a pooled ergm vs a blockdiagonal model, the latter to be fitted using [ergm::ergm](#).

Usage

```
blockdiagonalize(x, attrname = "block")

splitnetwork(x, attrname)

ergm_blockdiag(formula, ...)
```

Arguments

x	In the case of <code>blockdiagonalize</code> , a list of networks or matrices. For <code>splitnetwork</code> a single network object with a vertex attribute that can be used to split the data.
attrname	Name of the attribute that holds the block ids.
formula	An ergm model which networks' will be wrapped with <code>blockdiagonalize</code> (see details).
...	Further arguments passed to the method.

Details

The function `ergm_blockdiag` is a wrapper function that takes the model's network, stacks the networks into a single block diagonal net, and calls `ergm::ergm` with the option `constraints = blockdiag("block")`.

One side effect of this function is that it loads the `ergm` package via `requireNamespace`, so after executing the function `ergm` the package will be loaded.

Value

An object of class `ergm::ergm`.

Examples

```
library(ergm)
data(fivenets)

fivenets2 <- blockdiagonalize(fivenets, attrname = "block") # A network with
ans0 <- ergm(
  fivenets2 ~ edges + nodematch("female"),
  constraints = ~blockdiag("block")
)
ans1 <- ergmito(fivenets ~ edges + nodematch("female"))

# This is equivalent
ans2 <- ergm_blockdiag(fivenets ~ edges + nodematch("female"))
```

check_support

Check the convergence of ergmito estimates

Description

This is an internal function used to check the convergence of the optim function.

Usage

```
check_support(target.stats, stats.statmat, threshold = 0.8, warn = TRUE)

check_convergence(optim_output, model, support, crit = 5)
```

Arguments

target.stats, stats.statmat	See ergmito_formulae .
threshold	Numeric scalar. Confidence range for flagging an observed statistic as potentially near the boundary.
warn	logical scalar.
optim_output	A list output from the stats::optim function.
model	An object of class ergmito_loglik .
support	As returned by <code>check_support</code> .
crit	Numeric scalar. Level at which a parameter estimate will be questioned.

Value

A list with the following components:

- `par` Updated set of parameters
- `vcov` Updated variance-covariance matrix
- `valid` Vector of integers with the parameters that are marked as OK.
- `status` Return code of the analysis. See details.
- `note` A note describing the status.

Return codes

The function makes an analysis of the outcome of the model and makes the corresponding adjustments when required. In particular, we check:

1. Whether the optimization algorithm converged or not
2. If the obtained estimates maximize the function. If this is not the case, the function checks whether the MLE may not exist. This usually happens when the log-likelihood function can improve by making increments to parameters that are already tagged as large. If the ll improves, then the value is replaced with Inf (+- depending on the sign of the parameter).
3. If the Hessian is semi-positive-definite, i.e. if it is invertible. If it is not, it usually means that the function did not converged, in which case we will use [MASS::ginv](#) instead.

The return codes are composed of two numbers, the first number gives information regarding of the parameter estimates, while the second number give information about the variance-covariance matrix.

Column 1:

- 0: Converged and estimates at the max.
- 1: It did not converged, but I see no issue in the max.
- 2: One or more estimates went to +/-Inf
- 3: All went to hell. All estimates went to +/-Inf

Column 2:

- 0: Hessian is p.s.d.
- 1: Hessian is not not p.s.d.

Possible codes and corresponding messages:

- 00 All OK (no message).
- 01 optim converged, but the Hessian is not p.s.d.. % Convergence, but the hessian is not psd
- 10 optim did not converged, but the estimates look OK.. % Optim did not reported convergence, but things look OK.
- 11 optim did not converged, and the Hessian is not p.s.d.. % Optim did not converged, but the hessian is not psd.
- 20 A subset of the parameters estimates was replaced with +/-Inf.. % One or more estimates went to inf, all finite were able to be inverted.
- 21 A subset of the parameters estimates was replaced with +/-Inf, and the Hessian matrix is not p.s.d.. % One or more are inf, hessian is not psd
- 30 All parameters went to +/-Inf suggesting that the MLE may not exists.. % All estimates went to Inf (degenerate distribution).

count_stats

Count Network Statistics

Description

This function is similar to what [ergm::summary_formula](#) does, but it provides a fast wrapper suited for matrix class objects.

Usage

```
count_stats(X, ...)

AVAILABLE_STATS()

## S3 method for class 'formula'
count_stats(X, ...)

## S3 method for class 'list'
count_stats(X, terms, attrs = NULL, ...)
```

Arguments

X	List of square matrices. (networks)
...	Passed to the method.
terms	Character vector with the names of the statistics to calculate. Currently, the only available statistics are: 'mutual', 'edges', 'triad', 'ctriad', 'ctriple', 'nodeicov', 'nodeocov', 'nodematch', 'triangle', 'balance', 't300', 't102', 'absdiff', 'idegree1.5', 'odegree1.5', 'ostar1', 'ostar2', 'ostar3', 'ostar4', 'istar1', 'istar2', 'istar3', 'istar4'.
attrs	A list of vectors. This is used when term has a nodal attribute such as <code>nodeicov(attrname="")</code> .

Value

A matrix of size $\text{length}(X) * \text{length}(\text{terms})$ with the corresponding counts of statistics.

Examples

```
# DGP
x <- powerset(5)
ans0 <- count_stats(x[1:20], c("mutual", "edges"))

# Calculating using summary_formula
fm <- x[[i]] ~ mutual + edges
ans1 <- lapply(1:20, function(i) {
  environment(fm) <- environment()
  ergm::summary_formula(fm)
})

ans1 <- do.call(rbind, ans1)

# Comparing
all.equal(unname(ans0), unname(ans1))
```

 ergmito

Estimation of ERGMs using Maximum Likelihood Estimation (MLE)

Description

As a difference from [ergm::ergm](#), `ergmito` uses the exact log-likelihood function for fitting the model. This implies that all the $2^{n*(n-1)}$ graphs are generated for computing the normalizing constant of the ERGM model. As a rule of thumb, directed graphs with more than 5 vertices should not be fitted using MLE, but instead MC-MLE as implemented in the `ergm` package. The same applies for un-directed graphs with more than 8 vertices..

Usage

```
ergmito(
  model,
  gattr_model = NULL,
  stats.weights = NULL,
  stats.statmat = NULL,
  optim.args = list(),
  init = NULL,
  use.grad = TRUE,
  target.stats = NULL,
  ntries = 1L,
  keep.stats = TRUE,
  ...
)
```

```

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

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

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

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

## S3 method for class 'ergmito'
logLik(object, ...)

## S3 method for class 'ergmito'
nobs(object, ...)

## S3 method for class 'ergmito'
vcov(object, solver = NULL, ...)

## S3 method for class 'ergmito'
formula(x, ...)

```

Arguments

<code>model</code>	Model to estimate. See <code>ergm::ergm</code> . The only difference with <code>ergm</code> is that the LHS can be a list of networks.
<code>gattr_model</code>	A formula. Model specification for graph attributes. This is useful when using multiple networks.
<code>stats.weights</code>	Either an integer vector or a list of integer vectors (see <code>exact_loglik</code>).
<code>stats.statmat</code>	Either a matrix or a list of matrices (see <code>exact_loglik</code>).
<code>optim.args</code>	List. Passed to <code>stats::optim</code> .
<code>init</code>	A numeric vector. Sets the starting parameters for the optimization routine. Default is a vector of zeros.
<code>use.grad</code>	Logical. When TRUE passes the gradient function to <code>optim</code> . This is intended for testing only (internal use).
<code>target.stats</code>	A matrix of target statistics (see <code>ergm::ergm</code>).
<code>ntries</code>	Integer scalar. Number of tries to estimate the MLE (see details).
<code>keep.stats</code>	Logical scalar. When TRUE (the default), the matrices and vectors associated with the sufficient statistics will be returned. Otherwise the function discards them. This may be useful for saving memory space when estimating multiple models.
<code>...</code>	Further arguments passed to the method. In the case of <code>ergmito</code> , <code>...</code> are passed to <code>ergmito_formulae</code> .

x, object	An object of class <code>ergmito</code>
solver	Function. Used to compute the inverse of the hessian matrix. When not null, the variance-covariance matrix is recomputed using that function. By default, <code>ergmito</code> uses MASS::ginv .

Value

An list of class `ergmito`:

- `call` The program call.
- `coef` Named vector. Parameter estimates.
- `iterations` Integer. Number of times the log-likelihood was evaluated (see [stats::optim](#)).
- `mle.lik` Numeric. Final value of the objective function.
- `null.lik` Numeric. Final value of the objective function for the null model.
- `covar` Square matrix of size `length(coef)`. Variance-covariance matrix computed using the exact hessian as implemented in [exact_hessian](#).
- `coef.init` Named vector of length `length(coef)`. Initial set of parameters used in the optimization.
- `formulae` An object of class [ergmito_loglik](#).
- `nobs` Integer scalar. Number of networks in the model.
- `network` Networks passed via model.
- `optim.out,optim.args` Results from the `optim` call and arguments passed to it.
- `status,note` Convergence code. See [check_convergence](#)
- `best_try` Integer scalar. Index of the run with the highest log-likelihood value.
- `history` Matrix of size `ntries * (k + 1)`. History of the parameter estimates and the reached log-likelihood values.
- `timer` Vector of times (for benchmarking). Each unit marks the starting point of the step.

MLE

Maximum Likelihood Estimates are obtained using the [stats::optim](#) function. The default method for maximization is BFGS using both the log-likelihood function and its corresponding gradient.

Another important factor to consider is the existence of the MLE estimates As shown in Handcock (2003), if the observed statistics are near the border of the support function (e.g. too many edges or almost none), then, even if the MLE estimates exists, the optimization function may not be able to reach the optima. Moreover, if the target (observed) statistics live in the boundary, then the MLE estimates do not exists. In general, this should not be an issue in the context of the pooled model, as the variability of observed statistics should be enough to avoid those situations.

The function `ergmito` will try to identify possible cases of non-existence, of the MLE, and if identified then try to re estimate the model parameters using larger values than the ones obtained, if the log-likelihood is greater, then it is assumed that the model is degenerate and the corresponding values will be replaced with either `+Inf` or `-Inf`. By default, this behavior is checked anytime that the absolute value of the estimates is greater than 5, or the sufficient statistics were flagged as potentially outside of the interior of the support (close to zero or to its max).

In the case of `ntries`, the optimization is repeated that number of times, each time perturbing the `init` parameter by adding a Normally distributed vector. The result which reaches the highest log-likelihood will be the one reported as parameter estimates. This feature is intended for testing only. Anecdotally, `optim` reaches the max in the first try.

See Also

The function [plot.ergmito](#) for post-estimation diagnostics.

Examples

```
# Generating a small graph
set.seed(12)
n <- 4
net <- rbernoulli(n, p = .7)

model <- net ~ edges + mutual

library(ergm)
ans_ergmito <- ergmito(model)
ans_ergm <- ergm(model)

# The ergmito should have a larger value
ergm.exact(ans_ergmito$coef, model)
ergm.exact(ans_ergm$coef, model)

summary(ans_ergmito)
summary(ans_ergm)

# Example 2: Estimating an ERGMito using data with know DGP parameters -----
data(fivenets)

model1 <- ergmito(fivenets ~ edges + nodematch("female"))
summary(model1) # This data has know parameters equal to -2.0 and 2.0

# Example 3: Likelihood ratio test using the lmtest R package

if (require(lmtest)) {
  data(fivenets)
  model1 <- ergmito(fivenets ~ edges + nodematch("female"))
  model2 <- ergmito(fivenets ~ edges + nodematch("female") + mutual)

  lrtest(model1, model2)
  # Likelihood ratio test
  #
  # Model 1: fivenets ~ edges + nodematch("female")
  # Model 2: fivenets ~ edges + nodematch("female") + mutual
  # #Df LogLik Df Chisq Pr(>Chisq)
  # 1 2 -34.671
  # 2 3 -34.205 1 0.9312 0.3346
}
```

ergmito_boot	<i>Bootstrap of ergmito</i>
--------------	-----------------------------

Description

Bootstrap of ergmito

Usage

```
ergmito_boot(x, ..., R, ncpus = 1L, cl = NULL)

## S3 method for class 'formula'
ergmito_boot(x, ..., R, ncpus = 1L, cl = NULL)

## S3 method for class 'ergmito'
ergmito_boot(x, ..., R, ncpus = 1L, cl = NULL)

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

Arguments

x	Either a formula or an object of class ergmito .
...	Additional arguments passed to the method.
R	Integer. Number of replicates
ncpus	Integer Number of CPUs to use.
cl	An object of class <code>cluster</code> (see parallel::makePSOCKcluster)

Details

The resulting sample of parameters estimates is then used to compute the variance-covariance matrix of the model. Cases in which Inf/NaN/NA values were returned are excluded from the calculation.

Value

An object of class `ergmito_boot` and [ergmito](#)

Examples

```
# Simulating 20 bernoulli networks of size 4
nets <- replicate(20, rbernoulli(4), simplify = FALSE)
```

ergmito_formulae *Processing formulas in ergmito*

Description

Analyze formula objects returning the matrices of weights and sufficient statistics to be used in the model together with the log-likelihood and gradient functions for joint models.

Usage

```
ergmito_formulae(
  model,
  gattr_model = NULL,
  target.stats = NULL,
  stats.weights = NULL,
  stats.statmat = NULL,
  env = parent.frame(),
  ...
)
```

Arguments

model	A formula. The left-hand-side can be either a small network, or a list of networks.
gattr_model	A formula. Model specification for graph attributes. This is useful when using multiple networks.
target.stats	Observed statistics. If multiple networks, then a list, otherwise a named vector (see ergm::summary_formula).
stats.weights, stats.statmat	Lists of sufficient statistics and their respective weights.
env	Environment in which model should be evaluated.
...	Further arguments passed to ergm::ergm.allstats .

Value

A list of class `ergmito_loglik`.

- `loglik` A function. The log-likelihood function.
- `grad` A function. The gradient of the model.
- `stats.weights, stats.statmat` two list of objects as returned by [ergm::ergm.allstats](#).
- `model` A formula. The model passed.
- `npars` Integer. Number of parameters.
- `nnets` Integer. Number of networks in the model.
- `vertex.attr` Character vector. Vertex attributes used in the model.
- `term.names` Names of the terms used in the model.

Examples

```
data(fivenets)
model <- ergmito_formulae(fivenets ~ edges + nodematch("female"))
print(model)
model$loglik(c(-2, 2))
```

ergmito_gof

Goodness of Fit diagnostics for ERGMito models

Description

Goodness of Fit diagnostics for ERGMito models

Usage

```
gof_ergmito(
  object,
  GOF = NULL,
  probs = c(0.05, 0.95),
  sim_ci = FALSE,
  R = 50000L,
  ncores = 1L,
  ...
)

## S3 method for class 'ergmito_gof'
print(x, digits = 2L, ...)

## S3 method for class 'ergmito_gof'
plot(
  x,
  y = NULL,
  main = NULL,
  sub = NULL,
  tnames = NULL,
  sort_by_ci = FALSE,
  ...
)
```

Arguments

object	An object of class ergmito .
GOF	Formula. Additional set of parameters to perform the GOF.
probs	Numeric vector. Quantiles to plot (see details).
sim_ci	Logical scalar. If FALSE, the default, it will compute the quantiles analytically, otherwise it samples from the ERGM distribution.

R	Integer scalar. Number of simulations to generate (passed to <code>sample</code>). This is only used if <code>sim_ci = TRUE</code> .
ncores	Integer scalar. Number of cores to use for parallel computations (currently ignored).
...	Further arguments passed to <code>stats::quantile</code> .
x	An object of class <code>ergmito_gof</code> .
digits	Number of digits to used when printing
y	Ignored.
main, sub	Title and subtitle of the plot (see <code>graphics::title</code>).
tnames	A named character vector. Alternative names for the terms.
sort_by_ci	Logical scalar. When TRUE it will sort the x-axis by the width of the CI in for the first parameter of the model.

Details

The Goodness of Fit function uses the fitted ERGMito to calculate a given confidence interval for a set of sufficient statistics. By default (and currently the only available option), this is done on the sufficient statistics specified in the model.

In detail, the algorithm is executed as follow:

For every network in the list of networks do:

1. Calculate the probability of observing each possible graph in its support using the fitted model.
2. If `sim_ci = TRUE`, draw R samples from each set of parameters using the probabilities computed. Then using the `quantile` function, calculate the desired quantiles of the sufficient statistics. Otherwise, compute the quantiles using the analytic quantiles using the full distribution.'

The plot method is particularly convenient since it graphically shows whether the target statistics of the model (observed statistics) fall within the simulated range.

The print method tries to copy (explicitly) the print method of the `gof` function from the `ergm` R package.

Value

An object of class `ergmito_gof`. This is a list with the following components:

- `ci` A list of matrices of length `nnets(object)` with the corresponding confidence intervals for the statistics of the model.
- `target.stats` A matrix of the target statistics.
- `ergmito.probs` A list of numeric vectors of length `nnets(object)` with the probabilities associated to each possible structure of network.
- `probs` The value passed via `probs`.
- `model` The fitted model.
- `term.names` Character vector. Names of the terms used in the model.
- `quantile.args` A list of the values passed via ...

Examples

```
# Fitting the fivenets model
data(fivenets, package = "ergmito")
fit <- ergmito(fivenets ~ edges + nodematch("female"))

# Calculating the gof
ans <- gof_ergmito(fit)

# Looking at the results
ans
plot(ans)
```

exact_loglik

Vectorized calculation of ERGM exact log-likelihood

Description

This function can be compared to [ergm::ergm.exact](#) with the statistics not centered at x , the vector of observed statistics.

Usage

```
exact_loglik(x, params, ...)

## S3 method for class 'ergmito_ptr'
exact_loglik(x, params, ...)

## Default S3 method:
exact_loglik(x, params, stats.weights, stats.statmat, ...)

exact_gradient(x, params, ...)

## S3 method for class 'ergmito_ptr'
exact_gradient(x, params, ...)

## Default S3 method:
exact_gradient(x, params, stats.weights, stats.statmat, ...)

exact_hessian(x, params, stats.weights, stats.statmat)
```

Arguments

<code>x</code>	Matrix. Observed statistics
<code>params</code>	Numeric vector. Parameter values of the model.
<code>...</code>	Arguments passed to the default methods.
<code>stats.weights</code>	Either an integer vector or a list of integer vectors (see exact_loglik).
<code>stats.statmat</code>	Either a matrix or a list of matrices (see exact_loglik).

Sufficient statistics

One of the most important components of `ergmito` is calculating the full support of the model's sufficient statistics. Right now, the package uses the function `ergm::ergm.allstats` which returns a list of two objects:

- `weights`: An integer vector of counts.
- `statmat`: A numeric matrix with the rows as unique vectors of sufficient statistics.

Since `ergmito` can vectorize operations, in order to specify weights and statistics matrices for each network in the model, the user needs to pass two lists `stats.weights` and `stats.statmat`. While both lists have to have the same length (since its elements are matched), this needs not to be the case with the networks, as the user can specify a single set of weights and statistics that will be recycled (smartly).

Examples

```
data(fivenets)
ans <- ergmito(fivenets ~ edges + nodematch("female"))

# This computes the likelihood for all the networks independently
with(ans$formulae, {
  exact_loglik(
    x      = target.stats,
    params = coef(ans),
    stats.weights = stats.weights,
    stats.statmat = stats.statmat
  )
})

# This should be close to zero
with(ans$formulae, {
  exact_gradient(
    x      = target.stats,
    params = coef(ans),
    stats.weights = stats.weights,
    stats.statmat = stats.statmat
  )
})

# Finally, the hessian
with(ans$formulae, {
  exact_hessian(
    x      = target.stats,
    params = coef(ans),
    stats.weights = stats.weights,
    stats.statmat = stats.statmat
  )
})
```

extract.ergmito	<i>Extract function to be used with the texreg package.</i>
-----------------	---

Description

To be used with the **texreg** package. This function can be used to generate nice looking tables of ERGMitos estimates.

Usage

```
extract.ergmito(  
  model,  
  include.aic = TRUE,  
  include.bic = TRUE,  
  include.loglik = TRUE,  
  include.nnets = TRUE,  
  include.convergence = TRUE,  
  ...  
)
```

Arguments

model	An object of class ergmito.
include.aic, include.bic, include.loglik	See texreg::extract .
include.nnets	Logical. When true, it adds the Number of networks used to the list of gof statistics. This can be useful when running pooled models.
include.convergence	Logical. When true it, adds the convergence value of the stats::optim function (0 means convergence).
...	Further arguments passed to summary.ergmito .

Examples

```
library(texreg)  
data(fivenets)  
ans <- ergmito(fivenets ~ edges + nodematch("female"))  
screenreg(ans)
```

 fivenets

Example of a group of small networks

Description

This list of networks was generated using the [new_ergmito](#) sampler from a set of 5 baseline networks with a random vector of female. The sufficient statistics that generate this data are edges and `nodematch("female")` with parameters -2.0 and 2.0 respectively.

Usage

```
fivenets
```

Format

An object of class `list` of length 5.

 geodesic

Geodesic distance matrix (all pairs)

Description

Calculates the shortest path between all pairs of vertices in a network. This uses the power matrices to do so, which makes it efficient only for small networks.

Usage

```
geodesic(x, force = FALSE, ...)
```

```
geodesita(x, force = FALSE, ...)
```

```
## S3 method for class 'list'
geodesic(x, force = FALSE, ...)
```

```
## S3 method for class 'matrix'
geodesic(x, force = FALSE, simplify = FALSE, ...)
```

```
## S3 method for class 'network'
geodesic(x, force = FALSE, simplify = FALSE, ...)
```

Arguments

x	Either a list of networks (or square integer matrices), an integer matrix, a network, or an ergmito.
force	Logical scalar. If force = FALSE (the default) and nvertex(x) > 100 it returns with an error. To force computation use force = TRUE.
...	Further arguments passed to the method.
simplify	Logical scalar. When TRUE it returns a matrix, otherwise, a list of length nnets(x).

Examples

```
data(fivenets)
geodesic(fivenets)
```

induced_submat	<i>Extract a submatrix from a network</i>
----------------	---

Description

This is similar to [network::get.inducedSubgraph](#). The main difference is that the resulting object will always be a list of matrices, and it is vectorized.

Usage

```
induced_submat(x, v, ...)

## S3 method for class 'list'
induced_submat(x, v, ...)

## S3 method for class 'network'
induced_submat(x, v, ...)

## S3 method for class 'matrix'
induced_submat(x, v, ...)
```

Arguments

x	Either a list or single matrices or network objects.
v	Either a list or a single integer vector of vertices to subset.
...	Currently ignored.

Details

Depending on the lengths of x and v , the function can take the following strategies:

- If both are of the same size, then it will match the networks and the vector of indices.
- If `length(x) == 1`, then it will use that single network as a baseline for generating the subgraphs.
- If `length(v) == 1`, then it will generate the subgraph using the same set of vertices for each network.
- If both have more than one element, but different sizes, then the function returns with an error.

Value

A list of matrices as a result of the subsetting.

Examples

```
x <- rbernoulli(100)
induced_submat(x, c(1, 10, 30:50))

x <- rbernoulli(c(20, 20))
induced_submat(x, c(1:10))
```

matrix_to_network *Manipulation of network objects*

Description

This function implements a vectorized version of `network::network.adjmat`. It allows us to turn regular matrices into network objects quickly.

Usage

```
matrix_to_network(
  x,
  directed = rep(TRUE, length(x)),
  hyper = rep(FALSE, length(x)),
  loops = rep(FALSE, length(x)),
  multiple = rep(FALSE, length(x)),
  bipartite = rep(FALSE, length(x))
)

## S3 method for class 'matrix'
matrix_to_network(
  x,
  directed = rep(TRUE, length(x)),
  hyper = rep(FALSE, length(x)),
```

```

loops = rep(FALSE, length(x)),
multiple = rep(FALSE, length(x)),
bipartite = rep(FALSE, length(x))
)

## S3 method for class 'list'
matrix_to_network(
  x,
  directed = rep(TRUE, length(x)),
  hyper = rep(FALSE, length(x)),
  loops = rep(FALSE, length(x)),
  multiple = rep(FALSE, length(x)),
  bipartite = rep(FALSE, length(x))
)

```

Arguments

x	Either a single square matrix (adjacency matrix), or a list of these.
directed	Logical scalar, if FALSE then the function only checks the upper diagonal of the matrix assuming it is undirected.
hyper, multiple, bipartite	Currently Ignored. Right now all the network objects created by this function set these parameters as FALSE.
loops	Logical scalar. When FALSE (default) it will skip the diagonal of the adjacency matrix.

Details

This version does not support adding the name parameter yet. The function in the network package includes the name of the vertices as an attribute.

Just like in the network function, NA are checked and added accordingly, i.e. if there is an NA in the matrix, then the value is recorded as a missing edge.

Value

An object of class network. This is a list with the following elements:

- *me1 Master Edge List*: A named list with length equal to the number of edges in the network. The list itself has 3 elements: *in1* (tail), *out1* (head), and *at1* (attribute). By default *at1*, a list itself, has a single element: *na*.
- *gal Graph Attributes List*: a named list with the following elements:
 - *n* Number of nodes
 - *mnext* Number of edges + 1
 - *directed,hyper,loops,multiple,bipartite* The arguments passed to the function.
- *val Vertex Attributes List*
- *iel In Edgest List*
- *oel Out Edgest List*

Examples

```
set.seed(155)
adjmats <- rbernoulli(rep(5, 20))
networks <- matrix_to_network(adjmats)
```

new_ergmito_ptr	<i>Creates a new ergmito_ptr</i>
-----------------	----------------------------------

Description

After calculating the support of the sufficient statistics, the second most computationally expensive task is computing log-likelihoods, Gradients, and Hessian matrices of ERGMs. This function creates a pointer to an underlying class that is optimized to improve memory allocation and save computation time when possible.

Usage

```
new_ergmito_ptr(target_stats, stats_weights, stats_statmat)
```

Arguments

target_stats, stats_weights, stats_statmat
see [exact_loglik](#).

Details

This function is for internal use only. Non-advanced users are not encouraged to use it. See [ergmito_formulae](#) and [exact_loglik](#) for user friendly wrappers of this function.

Recycling computations

Some components of the likelihood, its gradient, and hessian can be pre-computed and recycled when needed. For example, it is usually the case that in optimization gradients are computed using a current state of the model's parameter, which implies that the normalizing constant and some other matrix products will be the same between the log-likelihood and the gradient. Because of this, the underlying class `ergmito_ptr` will only re-calculate these shared components if the parameter used changes as well. This saves a significant amount of computation time.

Scope of the class methods

To save space, the class creates pointers to the matrices of sufficient statistics that the model uses. This means that once these objects are deleted the log-likelihood and the gradient functions become invalid from the computational point of view.

new_ergmito	<i>ERGMito sampler</i>
-------------	------------------------

Description

Create a sampler object that allows you simulating streams of small networks fast.

Usage

```
new_ergmito(
  model,
  theta = NULL,
  sizes = NULL,
  cl = NULL,
  ncores = 1L,
  force = FALSE,
  ...
)

## S3 method for class 'ergmito_sampler'
x[i, j, ...]

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

Arguments

model	A formula.
theta	Named vector. Model parameters.
sizes	Integer vector. Values between 2 to 5 (6 becomes too intensive).
cl	An object of class cluster .
ncores	Integer. Number of processors to use.
force	Logical. When FALSE (default) will try to use ergmito's stat count functions (see count_stats). This means that if one of the requested statistics is not available in ergmito, then we will use ergm to compute them, which is significantly slower (see details).
...	Further arguments passed to ergm::ergm.allstats .
x	An object of class <code>ergmito_sampler</code> .
i, j	i is an integer vector indicating the indexes of the networks to draw, while j the corresponding sizes. These need not to be of the same size.

Details

While the **ergm** package is very efficient, it was not built to do some of the computations required in the **ergmito** package. This translates in having some of the functions of the package (**ergm**) with poor speed performance. This led us to "reinvent the wheel" in some cases to speed things up, this includes calculating observed statistics in a list of networks.

The indexing method, `[.ergmito_sampler`, allows extracting networks directly by passing indexes. `i` indicates the index of the networks to draw, which go from 1 through $2^{(n*(n-1))}$, and `j` indicates the requested size.

Value

An environment with the following objects:

- `calc_prob` A function to calculate each graph's probability under the specified model.
- `call` A language object with the call.
- `counts` A list with 3 elements: `stats` the sufficient statistics of each network, `weights` and `statmat` the overall matrices of sufficient statistics used to compute the likelihood.
- `network0` The baseline network used to either fit the model or obtain attributes.
- `networks` A list with the actual sample space of networks.
- `prob` A numeric
- `sample` A function to draw samples. `n` specifies the number of samples to draw, `s` the size of the networks, and `theta` the parameter to use to calculate the likelihoods.
- `theta` Named numeric vector with the current values of the model parameters.

The indexing method `[.ergmito_sampler` returns a named list of length `length(j)`.

Examples

```
# We can generate a sampler from a random graph
set.seed(7131)
ans <- new_rergmito(rbernoulli(4) ~ edges)

# Five samples
ans$sample(5, s = 4)

# or we can use some nodal data:
data(fivenets)
ans <- new_rergmito(
  fivenets[[3]] ~ edges + nodematch("female"),
  theta = c(-1, 1)
)

# Five samples
ans$sample(5, s = 4) # All these networks have a "female" vertex attr
```

`nvertex`*Utility functions to query network dimensions*

Description

Utility functions to query network dimensions

Usage

```
nvertex(x)

nedges(x, ...)

## S3 method for class 'network'
nedges(x, ...)

## S3 method for class 'list'
nedges(x, ...)

## S3 method for class 'matrix'
nedges(x, ...)

## S3 method for class 'ergmito'
nedges(x, ...)

## S3 method for class 'formula'
nedges(x, ...)

## S3 method for class 'network'
nvertex(x)

## S3 method for class 'matrix'
nvertex(x)

## S3 method for class 'list'
nvertex(x)

## S3 method for class 'ergmito'
nvertex(x)

## S3 method for class 'formula'
nvertex(x)

nnets(x)

## S3 method for class 'list'
nnets(x)
```

```

## S3 method for class 'matrix'
nnets(x)

## S3 method for class 'network'
nnets(x)

## S3 method for class 'ergmito'
nnets(x)

## S3 method for class 'formula'
nnets(x)

is_directed(x, check_type = FALSE)

## S3 method for class 'network'
is_directed(x, check_type = FALSE)

## S3 method for class 'list'
is_directed(x, check_type = FALSE)

## Default S3 method:
is_directed(x, check_type = FALSE)

## S3 method for class 'ergmito'
is_directed(x, check_type = FALSE)

## S3 method for class 'formula'
is_directed(x, check_type = FALSE)

```

Arguments

<code>x</code>	Either an object of class <code>ergmito</code> , <code>network</code> , <code>formula</code> , or <code>matrix</code> .
<code>...</code>	Further arguments passed to the method. Currently only <code>nedges.network</code> receives arguments (see <code>network::network.edgecount</code>).
<code>check_type</code>	Logical scalar. When checking for whether the network is directed or not, we can ask the function to return with an error if what we are checking is not an object of class <code>network</code> , otherwise it simply returns <code>false</code> .

Value

`is_directed` checks whether the passed networks are directed using the function `network::is.directed`. In the case of multiple networks, the function returns a logical vector. Only objects of class `network` can be checked, otherwise, if `check_type = FALSE`, the function returns `TRUE` by default.

Examples

```

set.seed(771)
net <- lapply(rbernoulli(c(4, 4)), network::network, directed = FALSE)

```

```

is_directed(net)
is_directed(net[[1]])
is_directed(net ~ edges)
## Not run:
  is_directed(net[[1]][1:4, 1:4], check_type = TRUE) # Error

## End(Not run)
is_directed(net[[1]][1:4, 1:4])

```

plot.ergmito

Function to visualize the optimization surface

Description

General diagnostics function. This function allows to visualize the surface to be maximize at around a particular point.

Usage

```

## S3 method for class 'ergmito'
plot(
  x,
  y = NULL,
  domain = NULL,
  plot. = TRUE,
  par_args = list(),
  image_args = list(),
  breaks = 50L,
  extension = 4L,
  params_labs = stats::setNames(names(coef(x)), names(coef(x))),
  ...
)

```

Arguments

x	An object of class ergmito .
y, ...	Ignored.
domain	A list.
plot.	Logical. When TRUE (default), the function will call graphics::image and plot all possible combination of parameters.
par_args	Further arguments to be passed to graphics::par
image_args	Further arguments to be passed to graphics::image
breaks	Integer scalar. Number of splits per dimension.
extension	Numeric. Range value of the function.
params_labs	Named vector. Alternative labels for the parameters. It should be in the form of <code>c("original name" = "new name")</code> .

Details

It calculates the surface coordinates for each pair of parameters included in the ERGMito.

Value

A list of length `choose(length(object$coef), 2)` (all possible combinations of pairs of parameters), each with the following elements:

- z A matrix
- z A vector
- y A vector
- xlabel A string. Name of the ERGM parameter in the x-axis.
- ylabel A string. Name of the ERGM parameter in the y-axis.

The list is returned invisible.

See Also

The [ergmito](#) function.

Examples

```
set.seed(12)
x <- rbernoulli(c(4, 4, 5))

ans <- ergmito(x ~ edges + balance)

plot(ans)
```

powerset

Power set of Directed Graphs of size n

Description

Power set of Directed Graphs of size n

Usage

```
powerset(n, directed = TRUE, force = FALSE, chunk_size = 2e+05)
```

Arguments

n	Integer. Number of edges.
directed	Logical scalar. Whether to generate the power set of directed or undirected graphs,
force	Logical scalar. When TRUE it generates the power set for n>5, otherwise it returns with error.
chunk_size	Number of matrices to process at a time. If n = 5, then stack memory on the computer may overflow if chunk_size is relatively large.

Examples

```
powerset(2)
powerset(4, directed = FALSE)
```

predict.ergmito	<i>Prediction method for ergmito objects</i>
-----------------	--

Description

Takes an [ergmito](#) object and makes prediction at tie level. See details for information regarding its implementation.

Usage

```
## S3 method for class 'ergmito'
predict(object, newdata = NULL, ...)
```

Arguments

object	An object of class ergmito .
newdata	New set of networks (or network) on which to make the prediction.
...	Passed to new_rergmito , the workhorse of this method.

Details

After fitting a model with a small network (or a set of them), we can use the parameter estimates to calculate the likelihood of observing any given tie in the network, this is, the marginal probabilities at the tie level.

In particular, the function takes the full set of networks on the support of the model and adds them up weighted by the probability of observing them as predicted by the ERGM, formally:

$$\hat{A} = \sum_i \Pr(A = a_i) \times a_i$$

Where \hat{A} is the predicted adjacency matrix, and a_i is the i-th network in the support of the model. This calculation is done for each individual network used in the model.

Value

A list of adjacency matrix of length `nnets(object)` or, if specified `nnets(newdata)`.

Examples

```
data(fivenets)

# bernoulli graph
fit <- ergmito(fivenets ~ edges)

# all ties have the same likelihood
# which is roughly equal to:
# mean(nedges(fivenets)/(nvertex(fivenets)*(nvertex(fivenets) - 1)))
predict(fit)

# If we take into account vertex attributes, now the story is different!
fit <- ergmito(fivenets ~ edges + nodematch("female"))

# Not all ties have the same likelihood, since it depends on homophily!
predict(fit)
```

rbernoulli

Random Bernoulli graph

Description

Random Bernoulli graph

Usage

```
rbernoulli(n, p = 0.5)
```

Arguments

n	Integer vector. Size of the graph. If <code>length(n) > 1</code> , then it will be a list of random graphs.
p	Probability of a tie. This may be either a scalar, or a vector of the same length of n.

Value

If n is a single number, a square matrix of size n with zeros in the diagonal. Otherwise it returns a list of `length(n)` square matrices of sizes equal to those specified in n.

Examples

```
# A graph of size 4
rbernoulli(4)

# 3 graphs of various sizes
rbernoulli(c(3, 4, 2))

# 3 graphs of various sizes and different probabilities
rbernoulli(c(3, 4, 6), c(.1, .2, .3))
```

same_dist	<i>Compare pairs of networks to see if those came from the same distribution</i>
-----------	--

Description

If two networks are of the same size, and their vertex attributes are equal in terms of set comparison, then we say those came from the same distribution

Usage

```
same_dist(net0, net1, ...)

## S3 method for class 'matrix'
same_dist(net0, net1, attrnames = NULL, ...)

## S3 method for class 'network'
same_dist(net0, net1, attrnames = NULL, ...)
```

Arguments

net0, net1	Networks to be compared.
...	Ignored.
attrnames	Character vector. (optional) Names of the vertex attributes to be compared on. This is ignored in the matrix case.

Details

This function is used during the call of [ergmito_formulae](#) to check whether the function can recycle previously computed statistics for the likelihood function. In the case of models that only contain structural terms, i.e. attribute less, this can save significant amount of computing time and memory.

Value

A logical with an attribute what. TRUE meaning that the two networks come from the same distribution, and FALSE meaning that they do not. If FALSE the what attribute will be equal to either "size" or the name of the attribute that failed the comparison.

Examples

```
data(fivenets)
same_dist(fivenets[[1]], fivenets[[2]]) # Yes, same size
same_dist(fivenets[[1]], fivenets[[2]], "female") # No, different attr dist
```

simulate.ergmito	<i>Draw samples from a fitted ergmito model</i>
------------------	---

Description

Draw samples from a fitted ergmito model

Usage

```
## S3 method for class 'ergmito'
simulate(object, nsim = 1, seed = NULL, which_networks = 1L, theta = NULL, ...)
```

Arguments

object	An object of class ergmito .
nsim	Integer scalar. Number of samples to draw from the selected set of networks.
seed	See stats::simulate
which_networks	Integer vector. Specifies what networks to sample from. It must be within 1 and nnets(object).
theta, ...	Further arguments passed to new_ergmito .

Examples

```
data(fivenets)
fit <- ergmito(fivenets ~ edges + nodematch("female"))

# Drawing 200 samples from networks 1 and 3 from the model
ans <- simulate(fit, nsim = 200, which_networks = c(1, 3))
```


Index

*Topic **Internal**

check_support, 4

*Topic **datasets**

fivenets, 18

[.ergmito_sampler (new_rergmito), 23

as_adjmat, 3

AVAILABLE_STATS (count_stats), 6

blockdiagonalize, 3

check_convergence, 9

check_convergence (check_support), 4

check_support, 4

cluster, 23

coef.ergmito (ergmito), 7

count_stats, 6, 23

ergm::ergm, 3, 4, 7, 8

ergm::ergm.allstats, 12, 16, 23

ergm::ergm.exact, 15

ergm::summary_formula, 6, 12

ergm_blockdiag (blockdiagonalize), 3

ergmito, 7, 11, 13, 26–29, 32

ergmito_boot, 11

ergmito_formulae, 5, 8, 12, 22, 31

ergmito_gof, 13

ergmito_loglik, 5, 9

ergmito_loglik (ergmito_formulae), 12

exact_gradient (exact_loglik), 15

exact_hessian, 9

exact_hessian (exact_loglik), 15

exact_loglik, 8, 15, 15, 22

extract.ergmito, 17

fivenets, 18

formula, 26

formula.ergmito (ergmito), 7

geodesic, 18

geodesita (geodesic), 18

gof_ergmito (ergmito_gof), 13

graphics::image, 27

graphics::par, 27

graphics::title, 14

induced_submat, 19

is_directed (nvertex), 25

logLik.ergmito (ergmito), 7

MASS::ginv, 5, 9

matrix, 26

matrix_to_network, 20

nedges (nvertex), 25

network, 26

network::get.inducedSubgraph, 19

network::is.directed, 26

network::network, 20

network::network.edgcount, 26

new_ergmito_ptr, 22

new_rergmito, 18, 23, 29, 32

nnets (nvertex), 25

nobs.ergmito (ergmito), 7

nvertex, 25

parallel::makePSOCKcluster, 11

plot.ergmito, 10, 27

plot.ergmito_gof (ergmito_gof), 13

powerset, 28

predict.ergmito, 29

print.ergmito (ergmito), 7

print.ergmito_boot (ergmito_boot), 11

print.ergmito_gof (ergmito_gof), 13

print.ergmito_sampler (new_rergmito), 23

print.ergmito_summary (ergmito), 7

rbernoulli, 30

requireNamespace, 4

same_dist, 31

sample, [14](#)
simulate.ergmito, [32](#)
splitnetwork (blockdiagonalize), [3](#)
stats::optim, [5](#), [8](#), [9](#), [17](#)
stats::quantile, [14](#)
stats::simulate, [32](#)
summary.ergmito, [17](#)
summary.ergmito (ergmito), [7](#)

texreg::extract, [17](#)

vcov.ergmito (ergmito), [7](#)