Package 'ernm'

July 28, 2025

Type Package

Title Exponential-Family Random Network Models

Version 1.0.2

Date 2025-7-22

Description Estimation of fully and partially observed Exponential-Family Random Network Models (ERNM). Exponential-family Random Graph Models (ERGM) and Gibbs Fields are special cases of ERNMs and can also be estimated with the package. Please cite Fellows and Handcock (2012), ``Exponential-family Random Network Models" available at <doi:10.48550/arXiv.1208.0121>.

License LGPL-2.1

Depends R (>= 3.5.0), BH, methods, network, Rcpp

Imports dplyr, ggplot2, graphics, moments, rlang, stats, tidyr, trust

Suggests spelling, testthat

LinkingTo BH, Rcpp

Copyright 2014-2025 Ian Fellows

Encoding UTF-8

Language en-US

LazyData true

LazyLoad yes

RcppModules ernm

RoxygenNote 7.3.2

NeedsCompilation yes

Author Ian Fellows [aut],
Duncan Clark [aut, cre]

Maintainer Duncan Clark <dac6@williams.edu>

Repository CRAN

Date/Publication 2025-07-28 18:20:02 UTC

Contents

as.BinaryNet *Convert a network to either an UndirectedNet or DirectedNet object*

Description

Convert a network to either an UndirectedNet or DirectedNet object

Usage

```
as.BinaryNet(x, ...)
```

Arguments

x	the object
...	unused

Value

a BinaryNet object

as.network.DirectedNet *Conversion to network object*

Description

Conversion to network object

Usage

```
## S3 method for class 'DirectedNet'
as.network(x, ...)
```

Arguments

x	the object
...	unused

Value

a directed network object

`as.network.Rcpp_DirectedNet` *Convert and Rcpp_DirectedNet to a network object*

Description

Convert and Rcpp_DirectedNet to a network object

Usage

```
## S3 method for class 'Rcpp_DirectedNet'
as.network(x, ...)
```

Arguments

`x`	the object
`...`	unused

Value

a directed network object

`as.network.Rcpp_UndirectedNet` *Convert an Rcpp_UndirectedNet to a network object*

Description

Convert an Rcpp_UndirectedNet to a network object

Usage

```
## S3 method for class 'Rcpp_UndirectedNet'
as.network(x, ...)
```

Arguments

`x`	the object
`...`	unused

Value

a undirected network object

as.network.UndirectedNet *Conversion to network object*

Description

Conversion to network object

Usage

```
## S3 method for class 'UndirectedNet'
as.network(x, ...)
```

Arguments

x	the object
...	unused

Value

a undirected network object

BinaryNet *BinaryNet*

Description

BinaryNet

calculateStatistics *calculate model statistics from a formula*

Description

calculate model statistics from a formula

Usage

```
calculateStatistics(formula)
```

Arguments

formula	An ernm formula

Value

a list of statistics

call-symbols *Internal Symbols*

Description

Internal symbols used to access compiles code.

coef.ernm *Access ERNM parameters*

Description

Access ERNM parameters

Usage

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

Arguments

object	object
...	unused

Value

parameter vector

createCppModel *Creates a model*

Description

Creates a model

Usage

```
createCppModel(
  formula,
  ignoreMnar = TRUE,
  cloneNet = TRUE,
  theta = NULL,
  modelArgs = list(modelClass = "Model")
)
```

Arguments

formula	the model formula
ignoreMnar	ignore missing not at random offsets
cloneNet	should the network be cloned
theta	the model parameters.
modelArgs	additiional arguments for the model, e.g. tapering parameters

Value

a Model object

---

createCppSampler *Create a sampler*

Description

Create a sampler

Usage

```
createCppSampler(
  formula,
  modelArgs = list(modelClass = "Model"),
  dyadToggle = NULL,
  dyadArgs = list(),
  vertexToggle = NULL,
  vertexArgs = list(),
  nodeSamplingPercentage = 0.2,
  ignoreMnar = TRUE,
  theta = NULL,
  ...
)
```

Arguments

formula	the model formula
modelArgs	additiional arguments for the model, e.g. tapering parameters
dyadToggle	the method of sampling to use. Defaults to alternating between nodal-tie-dyad and neighborhood toggling.
dyadArgs	list of args for dyad
vertexToggle	the method of vertex attribuate sampling to use.
vertexArgs	list of args for vertex
nodeSamplingPercentage	how often the nodes should be toggled

ignoreMnar	ignore missing not at random offsets
theta	parameter values
...	additional parameters to be passed to createCppModel

Value

a MetropolisHastings object

`DirectedNet-class` *DirectedNet Class*

Description

An S4 (old-style) class representing a directed network.

`dutch_school` *Dutch School Data*

Description

This dataset contains network and actor attributes collected in early adolescence. It is provided by Andrea Knecht and stored in the package.

Usage

```
data(dutch_school)

data("dutch_school")
```

Format

An object of class `list` of length 4.

Data taken from https

//www.stats.ox.ac.uk/~snijders/siena/tutorial2010_data.htm. Processed as undirected networks.

Source

Knecht, A. (2004). *Network and actor attributes in early adolescence*. DANS Data Station Social Sciences and Humanities. DOI: doi:10.17026/dansz9bh2bp.

References

Snijders, T.A.B., Steglich, C.E.G., and van de Bunt, G.G. (2010), Introduction to actor-based models for network dynamics, Social Networks 32, 44-60, http://dx.doi.org/10.1016/j.socnet.2009.02.004.

ernm *Fits an ERNM model*

Description

Fits an ERNM model

Usage

```
ernm(
  formula,
  tapered = TRUE,
  tapering_r = 3,
  modelArgs = list(),
  nodeSamplingPercentage = 0.2,
  modelType = NULL,
  likelihoodArgs = list(),
  fullToggles = c("Compound_NodeTieDyad_Neighborhood", "DefaultVertex"),
 missingToggles = c("Compound_NodeTieDyadMissing_NeighborhoodMissing", "VertexMissing"),
  ...
)
```

Arguments

formula	model formula
tapered	should the model be tapered
tapering_r	the tapering parameter (tau = 1/(tapering_r^2 +5))
modelArgs	additiional arguments for the model, e.g. tapering parameters that override the defaults
nodeSamplingPercentage	how often are nodal variates toggled
modelType	either FullErnmModel or MissingErnmModel if NULL will check for missingness
likelihoodArgs	additiional arguments for the ernmLikelihood
fullToggles	a character vector of length 2 indicating the dyad and vertex toggle types for the unconditional simulations
missingToggles	a character vector of length 2 indicating the dyad and vertex toggle types for the conditional simulations
...	additional parameters for ernmFit

Value

a fitted model

`ernmFit` *Fit an ernm*

Description

Fit an ernm

Usage

```
ernmFit(
  sampler,
  theta0,
  mcmcBurnIn = 10000,
  mcmcInterval = 100,
  mcmcSampleSize = 10000,
  minIter = 3,
  maxIter = 40,
  objectiveTolerance = 0.5,
  gradTolerance = 0.25,
  meanStats,
  verbose = 1,
  method = c("bounded", "newton")
)
```

Arguments

`sampler`	the ErnmModel
`theta0`	initial starting values
`mcmcBurnIn`	burn in
`mcmcInterval`	interval
`mcmcSampleSize`	sample size
`minIter`	minimum number of iterations
`maxIter`	maximum number of iterations
`objectiveTolerance`	convergance criteria on change in log likelihood ratio
`gradTolerance`	convergance criteria on scaled gradient
`meanStats`	if non-missing, these are the target statistics
`verbose`	level of verbosity 0, 1, or 2
`method`	the optimization method to use

Value

ernm object

`ErnmModels` *Models*

Description

Models

`ernmPackageSkeleton` *Create an ERNM Package Skeleton*

Description

Creates a skeleton for a package extending the ernm package by copying an example package.

Usage

```
ernmPackageSkeleton(path = ".")
```

Arguments

`path`	A character string specifying the directory where the package skeleton will be created.

Value

A logical value indicating whether the copy was successful.

`ErnmSamplers` *Metropolis Samplers*

Description

Metropolis Samplers

ernm_gof *Goodness of fit for ERNM model*

Description

Goodness of fit plot for ERNM models, particularly suited for comparing models

Goodness of fit plot for ERNM models, particularly suited for comparing models

Usage

```
ernm_gof(
  models,
  observed_network = NULL,
  stats_formula,
  style = "histogram",
  scales = "fixed",
  print = TRUE,
  n_sim = 10000,
  burnin = 10000,
  interval = 100
)

ernm_gof(
  models,
  observed_network = NULL,
  stats_formula,
  style = "histogram",
  scales = "fixed",
  print = TRUE,
  n_sim = 10000,
  burnin = 10000,
  interval = 100
)
```

Arguments

models	named list of ernm models to be to be compared (can be length 1
observed_network	the observed network
stats_formula	the formula for the statistics
style	the style of the plot, either 'histogram' or 'boxplot'
scales	the scales of the plot, either 'fixed' or 'free'
print	whether to print the plot
n_sim	the number of simulations to run
burnin	the burnin for the MCMC simulation
interval	the samplling interval for MCMC simualtion

Value

A list containing goodness-of-fit plots and simulated statistics

A list containing goodness-of-fit plots and simulated statistics

---

`extract-methods` *Subsetting and assignment for Net objects*

Description

These methods allow standard subsetting (‘[‘) and assignment (‘[<-‘) for ‘DirectedNet‘ and ‘UndirectedNet‘ objects.

Usage

```
## S4 method for signature 'Rcpp_DirectedNet'
x[i, j, ..., maskMissing = TRUE, drop = TRUE]

## S4 method for signature 'Rcpp_UndirectedNet'
x[i, j, ..., maskMissing = TRUE, drop = TRUE]

## S4 replacement method for signature 'Rcpp_DirectedNet'
x[i, j, ...] <- value

## S4 replacement method for signature 'Rcpp_UndirectedNet'
x[i, j, ...] <- value
```

Arguments

`x`	A ‘DirectedNet‘ or ‘UndirectedNet‘ object.
`i, j`	Index vectors.
`...`	Currently unused.
`maskMissing`	Logical. Should missing values be masked by NA?
`drop`	Ignored (present for compatibility).
`value`	Values to assign (for ‘[<-‘ only).

Value

A modified object or extracted submatrix depending on the method.

`fullErnmLikelihood` *likelihood for a fully observed ernm*

Description

likelihood for a fully observed ernm

Usage

```
fullErnmLikelihood(
  theta,
  sample,
  theta0,
  stats,
  minEss = 5,
  damping = 0.05,
  method = c("cumulant", "sample"),
  order = 3
)
```

Arguments

`theta`	parameters
`sample`	mcmc sample
`theta0`	parameter values which generated sample
`stats`	observed statistics
`minEss`	minimum effective sample size
`damping`	a damping parameter
`method`	cumulant generating function approximation
`order`	the ordering

Value

a list with value, gradient, and hessian

FullErnmModel *creates an ERNM likelihood model*

Description

creates an ERNM likelihood model

Usage

```
FullErnmModel(sampler, logLik, ...)
```

Arguments

sampler	a sampler
logLik	a log likelihood function (optional)
...	additional parameters for the log likelihood

Value

a FullyObservedModel object

marErnmLikelihood *likelihood for an ernm with missing data*

Description

likelihood for an ernm with missing data

Usage

```
marErnmLikelihood(theta, sample, theta0, stats, minEss = 5, damping = 0.1)
```

Arguments

theta	parameters
sample	mcmc sample
theta0	parameter values which generated sample
stats	observed statistics
minEss	minimum effective sample size
damping	a damping parameter

Value

a list with value, gradient, and hessian

mcmcEss *MCMC Effective Sample Size*

Description

Computes the effective sample size from a statistic vector.

Usage

```
mcmcEss(x)
```

Arguments

x	A numeric vector.

Value

A numeric value representing the effective sample size.

References

Kass, R. E., Carlin, B. P., Gelman, A., & Neal, R. M. (1998). "Markov Chain Monte Carlo in Practice: A Roundtable Discussion." *The American Statistician*, 52(2), 93-100. DOI: doi:10.2307/2685466

mcmcse *MCMC Standard Error by Batch*

Description

Computes the MCMC standard error from a statistic vector using a batching method.

Usage

```
mcmcse(x, expon = 0.5)
```

Arguments

x	A numeric vector of statistics.
expon	A numeric value controlling the batch size; default is 0.5.

Value

A numeric value representing the estimated standard error.

MissingErnmModel *creates an ERNM likelihood model*

Description

creates an ERNM likelihood model

Usage

```
MissingErnmModel(observedSampler, unobservedSampler, ...)
```

Arguments

observedSampler	a sampler
unobservedSampler	a sampler conditional upon the observed values
...	additional parameters for the log likelihood

Value

a MarModel object

plot.ernm *Plot an ernm object*

Description

Plot an ernm object

Usage

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

Arguments

x	the object
...	unused

Value

No return value, plots the likelihood history

plot.Rcpp_DirectedNet *Plot an Rcpp_UnirectedNet object*

Description

Plot an Rcpp_UnirectedNet object

Usage

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

Arguments

x	the object
...	additional parameters for plot.network

Value

No return value, invisibly NULL

plot.Rcpp_UndirectedNet *Plot an UndirectedNet object*

Description

Plot an UndirectedNet object

Usage

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

Arguments

x	the object
...	additional parameters for plot.network

Value

No return value, invisibly NULL

print.ernm *Print ernm object*

Description

Print ernm object

Usage

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

Arguments

x	x
...	unused

Value

No return value, prints summary

registerDirectedStatistic *Register Statistics*

Description

Register Statistics

Usage

```
registerDirectedStatistic
```

Value

no return value

register_rcpp_net_methods *Register setter methods for Rcpp net objeccts*

Description

Register setter methods for Rcpp net objeccts

Usage

```
register_rcpp_net_methods()
```

Value

no return value

runErnmCppTests *runErnmCppTests*

Description

Runs the internal C++ tests for the ernm package.

Value

A logical value indicating whether all tests passed.

Examples

```
runErnmCppTests()
```

samplike *Sampson's Monks Data*

Description

This dataset represents the social network of relationships among monks in a monastery, as studied by Samuel F. Sampson. The data were collected during a period of instability and document both positive and negative interactions among the monks.

Usage

```
data(samplike)

data("samplike")
```

Format

An object of class network of length 5.

Details

The study recorded friendships, antagonisms, and other social relationships among the monks before and after a significant schism occurred in the monastery.

NOTE COPIED FROM ERGM PACKAGE

Mislabeling in Versions Prior to 3.6.1: In ergm version 3.6.0 and earlier, the adjacency matrices of the datasets reflected an older ordering of the names.

Source

Sampson, S. F. (1969). *Crisis in a cloister*. Unpublished Ph.D. dissertation, Cornell University.

References

White, H.C., Boorman, S.A. and Breiger, R.L. (1976). *Social structure from multiple networks. I. Blockmodels of roles and positions.* American Journal of Sociology, 81(4), 730-780.

See Also

florentine, network, plot.network, ergm

---

simulateStatistics *Simulate statistics*

Description

Simulate statistics

Usage

```
simulateStatistics(
  formula,
  theta,
  nodeSamplingPercentage = 0.2,
  mcmcBurnIn = 10000,
  mcmcInterval = 100,
  mcmcSampleSize = 100,
```

```
  ignoreMnar = TRUE,
  modelArgs = list(modelClass = "Model"),
  ...
)
```

Arguments

formula	the model formula
theta	model parameters
nodeSamplingPercentage	how often the nodes should be toggled
mcmcBurnIn	burn in
mcmcInterval	interval
mcmcSampleSize	sample size
ignoreMnar	ignore missing not at random offsets
modelArgs	additiional arguments for the model, e.g. tapering parameters
...	additional arguments to createCppSampler

Value

a list of statistics

---

summary.ernm *Summary for ernm object*

Description

Summary for ernm object

Usage

```
## S3 method for class 'ernm'
summary(object, include_AIC = TRUE, ...)
```

Arguments

object	object
include_AIC	whether to include AIC in the summary, will slow down
...	unused

Value

a data frame summary of the model

taperedErnmLikelihood *(E(g(X)) - g(x_o)^2 for TaperedModel*

Description

(E(g(X)) - g(x_o)^2 for TaperedModel

Usage

```
taperedErnmLikelihood(
  theta,
  centers,
  tau,
  sample,
  theta0,
  stats,
  minEss = 5,
  damping = 0.05
)
```

Arguments

theta	parameters
centers	center of statistics
tau	tapering parameter
sample	mcmc sample
theta0	parameter values which generated sample
stats	observed statistics
minEss	minimum effective sample size
damping	a damping parameter

Value

a list with value, gradient, and hessian

UndirectedNet-class *UndirectedNet Class*

Description

An S4 (old-style) class representing an undirected network.

vcov.ernm *Parameter covariance matrix*

Description

Parameter covariance matrix

Usage

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

Arguments

object	object
...	unused

Value

covariance matrix

Index