Package 'IsingSampler'

July 16, 2026

Type Package

Title Sampling Methods and Distribution Functions for the Ising Model

Version 0.5.0

Maintainer Sacha Epskamp <mail@sachaepskamp.com>

Description Sample states from the Ising model and compute the probability of states. Sampling can be done for any number of nodes, but due to the intractability of the Ising model the distribution can only be computed up to roughly 10 nodes. The Blume-Capel model, an Ising model with an additional on-site quadratic (crystal-field) term, is also supported.

License GPL-2

Imports Rcpp (>= 0.10.4), plyr, magrittr, nnet, dplyr

Depends R (>= 3.0.0)

LinkingTo Rcpp

URL https://github.com/SachaEpskamp/IsingSampler

NeedsCompilation yes

Author Sacha Epskamp [aut, cre],
Jesse Boot [ctb],
Adela Maria Isvoranu [ctb]

Repository CRAN

Date/Publication 2026-07-16 07:50:02 UTC

Contents

`IsingSampler-package` *Sampling methods and distribution functions for the Ising model*

Description

This package can be used to sample states from the Ising model and compute the probability of states. Sampling can be done for any number of nodes, but due to the intractability of the Ising model the distribution can only be computed up to ~10 nodes.

Author(s)

Sacha Epskamp

Maintainer: Sacha Epskamp <mail@sachaepskamp.com>

Examples

```
## This code compares the different sampling algorithms to the expected
## distribution of states in a tractible number of nodes.

## In the end are examples on how to obtain the distribution.

# Input:
N <- 5 # Number of nodes
nSample <- 1000 # Number of samples

# Ising parameters:
Graph <- matrix(sample(0:1,N^2,TRUE,prob = c(0.7, 0.3)),N,N) * rnorm(N^2)
Graph <- pmax(Graph,t(Graph)) / N
diag(Graph) <- 0
Thresh <- -(rnorm(N)^2)
Beta <- 1

# Response options (0,1 or -1,1):
Resp <- c(0L,1L)

# All posible states:
AllStates <- do.call(expand.grid,lapply(1:N,function(x)Resp))

# Simulate with metropolis:
MetData <- IsingSampler(nSample, Graph, Thresh, Beta, 1000/N,
  responses = Resp, method = "MH")

# Simulate exact samples (CFTP):
ExData <- IsingSampler(nSample, Graph, Thresh, Beta, 100,
```

```
  responses = Resp, method = "CFTP")

# Direct simulation:
DirectData <- IsingSampler(nSample, Graph, Thresh, Beta, method = "direct")

# State distributions:
MetDist <- apply(AllStates,1,function(x)sum(colSums(t(MetData) == x)==N))
ExDist <- apply(AllStates,1,function(x)sum(colSums(t(ExData) == x)==N))
DirectDist <- apply(AllStates,1,function(x)sum(colSums(t(DirectData) == x)==N))
ExpDist <- exp(- Beta * apply(AllStates,1,function(s)IsingSampler:::H(Graph,s,Thresh)))
ExpDist <- ExpDist/sum(ExpDist) * nSample

## Plot to compare distributions:
plot(MetDist, type="l", col="blue", pch=16, xlab="State", ylab="Freq",
  ylim=c(0,max(MetDist,DirectDist,ExDist)))
points(DirectDist,type="l",col="red",pch=16)
points(ExpDist,type="l",col="green",pch=16)
points(ExDist,type="l",col="purple",pch=16)
legend("topright", col=c("blue","red","purple","green"),
  legend=c("Metropolis","Direct","Exact","Expected"),lty=1,bty='n')

## Likelihoods:

# Sumscores:
IsingSumLikelihood(Graph, Thresh, Beta, Resp)

# All states:
IsingLikelihood(Graph, Thresh, Beta, Resp)

# Single state:
IsingStateProb(rep(Resp[1],N),Graph, Thresh, Beta, Resp)
```

`BlumeCapelSampler` *Sample states from the Blume-Capel model*

Description

Samples states from the Blume-Capel model, an Ising model with an additional on-site quadratic ("single-ion anisotropy" or "crystal-field") term. This is a thin wrapper around `IsingSampler` that sets ordered (spin-1) response options by default and requires the quadratic parameter `delta`.

Usage

```
BlumeCapelSampler(n, graph, thresholds, delta, beta = 1, nIter = 100,
    responses = c(-1L, 0L, 1L), method = c("MH", "direct"),
    CFTPretry = 10, constrain)
```

Arguments

`n`	Number of states to draw.
`graph`	Square matrix indicating the weights of the network. Must be symmetrical with 0 as diagonal.
`thresholds`	Vector indicating the thresholds (external field). A single value is recycled over all nodes.
`delta`	The quadratic (Blume-Capel) parameter, entering the Hamiltonian as `delta_i * s_i^2`. Either a single value (recycled over all nodes) or one value per node. Must be supplied. A positive `delta` favours the middle category; a negative `delta` favours the extreme categories.
`beta`	Scalar indicating the inverse temperature.
`nIter`	Number of iterations in the Metropolis sampler.
`responses`	Ordered response options, defaulting to the spin-1 values `c(-1L, 0L, 1L)`. Other (numeric) response options may be supplied, but the `delta` term only has an identifiable effect with more than two options.
`method`	The sampling method, either `"MH"` (Metropolis-Hastings, default) or `"direct"`. Exact sampling (`"CFTP"`) is implemented for two response options only and is therefore not available here; `delta` has no identifiable effect on binary responses in any case.
`CFTPretry`	Passed to `IsingSampler`; unused for the methods offered here.
`constrain`	A (number of samples) by (number of nodes) matrix with samples that need be constrained; `NA` indicates that the sample is unconstrained.

Details

The Blume-Capel Hamiltonian implemented here is

$$H(s) = -\sum_i \tau_i s_i - \sum_{i<j} \omega_{ij} s_i s_j + \sum_i \delta_i s_i^2,$$

with thresholds τ, network weights ω (`graph`) and quadratic terms δ (`delta`); states are drawn with probability proportional to $\exp(-\beta H(s))$. Setting `delta = 0` recovers the ordinary (multi-level) Ising model, so `BlumeCapelSampler(..., delta = 0)` is equivalent to `IsingSampler`.

Value

A matrix containing samples of states.

Author(s)

Sacha Epskamp (mail@sachaepskamp.com)

See Also

`IsingSampler`, `IsingLikelihood`

Examples

```
## Not run:
# A small ferromagnetic network:
P <- 5
W <- matrix(0.5, P, P); diag(W) <- 0

# Low crystal field: ordered (mostly +1 / -1):
X0 <- BlumeCapelSampler(1000, W, thresholds = 0, delta = 0, beta = 1)

# High crystal field: disordered "0" phase (mostly the middle category):
X3 <- BlumeCapelSampler(1000, W, thresholds = 0, delta = 3, beta = 1)

mean(X0 == 0)  # low
mean(X3 == 0)  # high

## End(Not run)
```

`EstimateIsing` *non-regularized estimation methods for the Ising Model*

Description

This function can be used for several non-regularized estimation methods of the Ising Model. See details.

Usage

```
EstimateIsing(data, responses, beta = 1, method = c("uni", "pl",
                 "bi", "ll"), adj = matrix(1, ncol(data), ncol(data)),
                 ...)
EstimateIsingUni(data, responses, beta = 1, adj = matrix(1, ncol(data),
                ncol(data)), min_sum = -Inf, thresholding = FALSE, alpha = 0.01,
                 AND = TRUE, ...)
EstimateIsingBi(data, responses, beta = 1, ...)
EstimateIsingPL(data, responses, beta = 1, ...)
EstimateIsingLL(data, responses, beta = 1, adj = matrix(1, ncol(data),
                 ncol(data)), ...)
```

Arguments

`data`	Data frame with binary responses to estimate the Ising model over
`responses`	Vector of length two indicating the response coding (usually `c(0L, 1L)` or `c(-1L, 1L)`). When supplied, it must match the coding actually present in the data; if either response option is absent from the data a warning is given (and the results are likely meaningless). If missing, the coding is taken from the data as `sort(unique(c(data)))`.

beta	Inverse temperature parameter. The Ising measure depends on the parameters only through the products beta * graph and beta * thresholds, so the parameters are only identified up to this scaling. All estimators return graph and thresholds such that, combined with the supplied beta, they reproduce the data-generating distribution; equivalently, the returned parameters equal the estimates obtained with beta = 1 divided by beta. For pl this is achieved by maximizing the pseudolikelihood at the supplied beta; for uni, bi and ll the parameters are estimated at beta = 1 and divided by beta. The raw results element is not rescaled.
method	The method to be used. pl uses pseudolikelihood estimation, uni sequential univariate regressions, bi bivariate regressions and ll estimates the Ising model as a loglinear model.
adj	Adjacency matrix of the Ising model.
min_sum	The minimum sum score that is artifically possible in the dataset. Defaults to -Inf. Set this only if you know a lower sum score is not possible in the data, for example due to selection bias.
thresholding	Logical, should the model be thresholded for significance?
alpha	Alpha level used in thresholding
AND	Logical, should an AND-rule (both regressions need to be significant) or OR-rule (one of the regressions needs to be significant) be used?
...	Arguments sent to estimator functions

Details

The following algorithms can be used (see Epskamp, Maris, Waldorp, Borsboom; in press).

pl Estimates the Ising model by maximizing the pseudolikelihood (Besag, 1975).

uni Estimates the Ising model by computing univariate logistic regressions of each node on all other nodes. This leads to a single estimate for each threshold and two estimates for each network parameter. The two estimates are averaged to produce the final network. Uses glm.

bi Estimates the Ising model using multinomial logistic regression of each pair of nodes on all other nodes. This leads to a single estimate of each network parameter and p estimates of each threshold parameter. Uses multinom.

ll Estimates the Ising model by phrasing it as a loglinear model with at most pairwise interactions. Uses loglin.

Value

A list containing the estimation results:

graph	The estimated network, scaled such that beta * graph corresponds to the data-generating measure
thresholds	The estimated thresholds, scaled such that beta * thresholds corresponds to the data-generating measure
results	The results object used in the analysis (on the raw estimation scale, not rescaled by beta)

Note

Method "bi" (EstimateIsingBi), which fits a multinomial logistic regression for each pair of nodes, can be unreliable when the joint high–high response category (both nodes taking the high response, i.e. the (1, 1) pair) is rarely or never observed. In that situation the corresponding multinomial cell is (near-)empty and the estimates for the affected parameters become unstable or fail to be identified. Prefer another method (for example "pl" or "uni") when such response pairs are sparse.

Author(s)

Sacha Epskamp (mail@sachaepskamp.com)

References

Epskamp, S., Maris, G., Waldorp, L. J., and Borsboom, D. (in press). Network Psychometrics. To appear in: Irwing, P., Hughes, D., and Booth, T. (Eds.), Handbook of Psychometrics. New York: Wiley.

Besag, J. (1975), Statistical analysis of non-lattice data. The statistician, 24, 179-195.

Examples

```
# Input:
N <- 5 # Number of nodes
nSample <- 500 # Number of samples

# Ising parameters:
Graph <- matrix(sample(0:1,N^2,TRUE,prob = c(0.7, 0.3)),N,N) * rnorm(N^2)
Graph <- Graph + t(Graph)
diag(Graph) <- 0
Thresholds <- rep(0,N)
Beta <- 1

# Response options (0,1 or -1,1):
Resp <- c(0L,1L)
Data <- IsingSampler(nSample,Graph, Thresholds)

# Pseudolikelihood:
resPL <- EstimateIsing(Data, method = "pl")
cor(Graph[upper.tri(Graph)], resPL$graph[upper.tri(resPL$graph)])

# Univariate logistic regressions:
resUni <- EstimateIsing(Data, method = "uni")
cor(Graph[upper.tri(Graph)], resUni$graph[upper.tri(resUni$graph)])

# bivariate logistic regressions:
resBi <- EstimateIsing(Data, method = "bi")
cor(Graph[upper.tri(Graph)], resBi$graph[upper.tri(resBi$graph)])

# Loglinear model:
resLL <- EstimateIsing(Data, method = "ll")
cor(Graph[upper.tri(Graph)], resLL$graph[upper.tri(resLL$graph)])
```

`IsingEntropy` *Entropy of the Ising Model*

Description

Returns (marginal/conditional) Shannon entropy of the Ising model.

Usage

```
IsingEntropy(graph, thresholds, beta = 1, conditional = numeric(0),
          marginalize = numeric(0), base = 2, responses = c(0L, 1L),
          delta = 0)
```

Arguments

`graph`	Weights matrix
`thresholds`	Thresholds vector
`beta`	Inverse temperature
`conditional`	Indices of nodes to condition on
`marginalize`	Indices of nodes to marginalize over
`base`	Base of the logarithm
`responses`	Vector of outcome responses.
`delta`	Optional per-node quadratic (Blume-Capel) term added to the Hamiltonian as `delta_i * s_i^2`; a single value (recycled over nodes) or one value per node. The default `0` corresponds to the ordinary Ising model. See `BlumeCapelSampler`.

Note

The old (misspelled) name `IsingEntrophy` is deprecated; use `IsingEntropy` instead. `IsingEntrophy` is retained as a thin wrapper that emits a deprecation warning and forwards to `IsingEntropy`.

Author(s)

Sacha Epskamp <mail@sachaepskamp.com>

IsingLikelihood *Likelihood of all states from tractible Ising model.*

Description

This function returns the likelihood of all possible states. Is only tractible up to rougly 10 nodes.

Usage

```
IsingLikelihood(graph, thresholds, beta, responses = c(0L, 1L),
                potential = FALSE, delta = 0)
```

Arguments

graph	Square matrix indicating the weights of the network. Must be symmetrical with 0 as diagonal.
thresholds	Vector indicating the thresholds, also known as the external field.
beta	Scalar indicating the inverse temperature.
responses	Response options. Typically set to `c(-1L, 1L)` or `c(0L, 1L)` (default). Two or more numeric response options are supported, including non-integer values such as `seq(-1, 1, by = 0.5)`.
potential	Logical, return the potential instead of the probability of each state?
delta	Optional per-node quadratic (Blume-Capel) term added to the Hamiltonian as `delta_i * s_i^2`; a single value (recycled over nodes) or one value per node. The default `0` corresponds to the ordinary Ising model. See `BlumeCapelSampler`.

Author(s)

Sacha Epskamp

IsingPL *Pseudo-likelihood*

Description

Computes the pseudolikelihood of a dataset given an Ising Model.

Usage

```
IsingPL(x, graph, thresholds, beta, responses = c(0L, 1L))
```

Arguments

x	A binary dataset
graph	Square matrix indicating the weights of the network. Must be symmetrical with 0 as diagonal.
thresholds	Vector indicating the thresholds, also known as the external field.
beta	Scalar indicating the inverse temperature.
responses	Response options. Typically set to `c(-1L, 1L)` or `c(0L, 1L)` (default). Must be integers!

Value

The pseudolikelihood

Author(s)

Sacha Epskamp (mail@sachaepskamp.com)

IsingSampler *Sample states from the Ising model*

Description

This function samples states from the Ising model using one of three methods. See details.

Usage

```
IsingSampler(n, graph, thresholds, beta = 1, nIter = 100, responses = c(0L, 1L),
    method = c("MH", "CFTP", "direct"), CFTPretry = 10, constrain, delta = 0)
```

Arguments

n	Number of states to draw
graph	Square matrix indicating the weights of the network. Must be symmetrical with 0 as diagonal.
thresholds	Vector indicating the thresholds, also known as the external field.
beta	Scalar indicating the inverse temperature.
nIter	Number of iterations in the Metropolis and exact sampling methods.
responses	Response options. Typically set to `c(-1L, 1L)` or `c(0L, 1L)` (default). Two or more response options are supported, and they may be any numeric values, including non-integers (e.g. `c(-1L, 0L, 1L)`, `1:5` or `seq(-1, 1, by = 0.5)`). With more than two options the conditional distribution of each node is the categorical (softmax) generalization of the binary Ising conditional. The pairwise interaction always enters the Hamiltonian as `J_ij * s_i * s_j`, i.e. through the (numeric) product of the response values, and each node has a single threshold. When all response options are integer-valued the result is an integer matrix (as before); otherwise it is a numeric (double) matrix.

`method` The sampling method to use. Must be `"MH"`, `"CFTP"` or `"direct"`. See details.

`CFTPretry` The amount of times a sample from CFTP may be retried. If after 100 couplings from the past the chain still results in `NA` values the chain is reset with different random numbers. Be aware that data that requies a lot of CFTP resets might not resemble exact samples anymore.

`constrain` A (number of samples) by (number of nodes) matrix with samples that need be constrained; `NA` indicates that the sample is unconstrained. Defaults to a matrix of `NA`s. Non-`NA` values must be members of `responses`. Constraints are honored by all three sampling methods: constrained (clamped) nodes are fixed to the given value and the remaining nodes are sampled from the conditional distribution given the clamped nodes (for `"MH"` and `"CFTP"` the clamped nodes are simply never updated; for `"direct"` the state space is restricted to matching states and the probabilities are renormalized).

`delta` Optional per-node quadratic (Blume-Capel single-ion / crystal-field) term added to the Hamiltonian as `delta_i * s_i^2`. Either a single value (recycled over nodes) or a vector with one value per node. The default `0` reproduces the ordinary Ising model. A positive `delta` favours response options near zero (the middle category); a negative `delta` favours the extreme categories. This term only has an identifiable effect with more than two response options; with two response options it is absorbed into the thresholds (and is constant, hence has no effect, for symmetric responses such as `c(-1, 1)`). See `BlumeCapelSampler`.

Details

This function uses one of three sampling methods. `"MH"` can be used to sample using a Metropolis-Hastings algorithm. The chain is initiated with random values from the response options, then for each iteration each node is redrawn from its full-conditional distribution given all other nodes and parameters. For two response options this is the usual binary update; for more than two response options the node is drawn from the categorical (softmax) full-conditional over all response options. Typically, 100 of such iterations should suffice for the chain to converge, though strongly coupled models with more than two response options may require a larger `nIter`.

The second method, `"CFTP"` enhances the Metropolis-Hastings algorithm with Coupling from the Past (CFTP; Murray, 2007) to draw exact samples from the distribution. This is slower than the default Metropolis-Hastings but guarantees exact samples. However, it does depend on the graph structure and the number of nodes if these exact samples can be obtained in feasable time. `"CFTP"` is currently only implemented for two response options; for more than two response options use `"MH"` or `"direct"`.

The third option, `"direct"`, simply computes for every possibly state the probability and draws samples directly from the distribution of states by using these probabilities. This also guarantees exact samples, but quickly becomes intractible (roughly above 10 nodes).

Value

A matrix containing samples of states.

Author(s)

Sacha Epskamp (mail@sachaepskamp.com)

References

Murray, I. (2007). Advances in Markov chain Monte Carlo methods.

See Also

IsingSampler-package for examples; BlumeCapelSampler for the Blume-Capel (quadratic / crystal-field) extension.

Examples

```
## See IsingSampler-package help page for the main usage examples.

## Constraining (clamping) nodes via the 'constrain' argument -----------------
## 'constrain' is a (number of samples) by (number of nodes) matrix; NA marks a
## free (unconstrained) node, a non-NA value clamps that node to that response.
N <- 3
graph <- matrix(0.5, N, N)
diag(graph) <- 0
thresholds <- rep(0, N)

# Clamp node 1 to 1 in every one of the 10 samples, leave nodes 2 and 3 free:
constrain <- matrix(NA_integer_, 10, N)
constrain[, 1] <- 1L

# All three methods honor the constraint (node 1 is always 1):
set.seed(1)
sMH     <- IsingSampler(10, graph, thresholds, method = "MH",     constrain = constrain)
sCFTP   <- IsingSampler(10, graph, thresholds, method = "CFTP",   constrain = constrain)
sDirect <- IsingSampler(10, graph, thresholds, method = "direct", constrain = constrain)

all(sMH[, 1] == 1L)      # TRUE
all(sCFTP[, 1] == 1L)    # TRUE
all(sDirect[, 1] == 1L)  # TRUE
```

IsingStateProb *Likelihood of single state from tractible Ising model.*

Description

This function returns the likelihood of a single possible state. Is only tractible up to rougly 10 nodes.

Usage

```
IsingStateProb(s, graph, thresholds, beta, responses = c(0L, 1L), delta = 0)
```

Arguments

s	Vector contaning the state to evaluate.
graph	Square matrix indicating the weights of the network. Must be symmetrical with 0 as diagonal.
thresholds	Vector indicating the thresholds, also known as the external field.
beta	Scalar indicating the inverse temperature.
responses	Response options. Typically set to `c(-1L, 1L)` or `c(0L, 1L)` (default). Two or more numeric response options are supported, including non-integer values such as `seq(-1, 1, by = 0.5)`.
delta	Optional per-node quadratic (Blume-Capel) term added to the Hamiltonian as `delta_i * s_i^2`; a single value (recycled over nodes) or one value per node. The default `0` corresponds to the ordinary Ising model. See `BlumeCapelSampler`.

Author(s)

Sacha Epskamp (mail@sachaepskamp.com)

IsingSumLikelihood *Likelihood of sumscores from tractible Ising model.*

Description

This function returns the likelihood of all possible sumscores. Is only tractible up to rougly 10 nodes.

Usage

```
IsingSumLikelihood(graph, thresholds, beta, responses = c(0L, 1L), delta = 0)
```

Arguments

graph	Square matrix indicating the weights of the network. Must be symmetrical with 0 as diagonal.
thresholds	Vector indicating the thresholds, also known as the external field.
beta	Scalar indicating the inverse temperature.
responses	Response options. Typically set to `c(-1L, 1L)` or `c(0L, 1L)` (default). Two or more numeric response options are supported, including non-integer values such as `seq(-1, 1, by = 0.5)`.
delta	Optional per-node quadratic (Blume-Capel) term added to the Hamiltonian as `delta_i * s_i^2`; a single value (recycled over nodes) or one value per node. The default `0` corresponds to the ordinary Ising model. See `BlumeCapelSampler`.

Author(s)

Sacha Epskamp (mail@sachaepskamp.com)

LinTransform *Transform parameters for linear transformations on response catagories*

Description

This function is mainly used to translate parameters estimated with response options set to 0 and 1 to a model in which the response options are -1 and 1, but can be used for any linear transformation of response options.

Usage

```
LinTransform(graph, thresholds, from = c(0L, 1L), to = c(-1L, 1L), a, b)
```

Arguments

graph	A matrix containing an Ising graph
thresholds	A vector containing thresholds
from	The original response encoding
to	The response encoding to transform to
a	The slope of the transformation. Overwrites to.
b	The intercept of the transformation. Overwrites to.

Author(s)

Sacha Epskamp <sacha.epskamp@gmail.com>

Examples

```
N <- 4 # Number of nodes

# Ising parameters:
Graph <- matrix(sample(0:1,N^2,TRUE,prob = c(0.7, 0.3)),N,N) * rnorm(N^2)
Graph <- pmax(Graph,t(Graph)) / N
diag(Graph) <- 0
Thresh <- -(rnorm(N)^2)
Beta <- 1

p1 <- IsingLikelihood(Graph, Thresh, Beta, c(0,1))

a <- 2
b <- -1

# p2 <- IsingLikelihood(Graph/(a^2), Thresh/a - (b*rowSums(Graph))/a^2, Beta, c(-1,1))

p2 <- IsingLikelihood(LinTransform(Graph,Thresh)$graph,
                      LinTransform(Graph,Thresh)$thresholds ,
```

```
                      Beta, c(-1,1))
LinTransform

round(cbind(p1[,1],p2[,1]),5)

plot(p1[,1],p2[,1])
abline(0,1)
```

`NodeInformation` *Mutual information between each node and the rest of the network*

Description

Computes, for each node of a (tractable) Ising model, the mutual information $I(\text{node}; \text{rest of network})$ between that node and all remaining nodes. This quantifies how much knowing the state of a single node reduces the uncertainty (Shannon entropy) about the joint state of the remaining nodes.

Usage

```
NodeInformation(graph, thresholds, beta = 1, base = 2,
          responses = c(0L, 1L), delta = 0)
```

Arguments

`graph`	Weights matrix
`thresholds`	Thresholds vector
`beta`	Inverse temperature
`base`	Base of the logarithm
`responses`	Vector of outcome responses.
`delta`	Optional per-node quadratic (Blume-Capel) term added to the Hamiltonian as `delta_i * s_i^2`; a single value (recycled over nodes) or one value per node. The default `0` corresponds to the ordinary Ising model. See `BlumeCapelSampler`.

Details

For each node i the returned value is computed via `IsingEntropy` as the entropy of the rest of the network with node i marginalised out, minus the entropy of the rest of the network conditional on node i:

$$I_i = H(\text{rest}) - H(\text{rest} \mid \text{node}_i).$$

This is the standard decomposition of the mutual information $I(\text{node}_i; \text{rest})$. As mutual information it is non-negative (up to numerical error). The computation is only tractable for small networks (roughly up to 10 nodes), since it enumerates the full state space via `IsingLikelihood`.

Value

A numeric vector of length `ncol(graph)`, giving the mutual information (in units set by `base`) between each node and the remaining nodes of the network.

Author(s)

Sacha Epskamp <mail@sachaepskamp.com>

See Also

IsingEntropy

Examples

```
# Small 3-node network:
N <- 3
graph <- matrix(0, N, N)
graph[upper.tri(graph)] <- c(0.5, 0.2, 0.8)
graph <- graph + t(graph)
thresholds <- rep(0, N)

# Mutual information of each node with the rest of the network:
NodeInformation(graph, thresholds)
```

Index