Package 'baycn'

July 31, 2020

Type Package

Title Bayesian Inference for Causal Networks

Version 1.2.0

Author Evan A Martin [aut, cre], Audrey Qiuyan Fu [aut]

Maintainer Evan A Martin <evanamartin@gmail.com>

Description A Bayesian hybrid approach for inferring Directed Acyclic Graphs (DAGs) for continuous, discrete, and mixed data. The algorithm can use the graph inferred by another more efficient graph inference method as input; the input graph may contain false edges or undirected edges but can help reduce the search space to a more manageable size. A Bayesian Markov chain Monte Carlo algorithm is then used to infer the probability of direction and absence for the edges in the network.
References:
Martin and Fu (2019) <arXiv:1909.10678>.

License GPL-3 | file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 3.5.0)

Imports egg, ggplot2, gtools, igraph, MASS, methods

RoxygenNote 7.1.1

Suggests testthat

NeedsCompilation no

Repository CRAN

Date/Publication 2020-07-31 18:40:06 UTC

R topics documented:

`baycn-class` *baycn class*

Description

baycn class

Slots

- `burnIn` The percentage of MCMC iterations that will be discarded from the beginning of the chain.
- `chain` A matrix where the rows contain the vector of edge states for the accepted graph.
- `decimal` A vector of decimal numbers. Each element in the vector is the decimal of the accepted graph.
- `iterations` The number of iterations for which the Metropolis-Hastings algorithm is run.
- `posteriorES` A matrix of posterior probabilities for all three edge states for each edge in the network.
- `posteriorPM` A posterior probability adjacency matrix.
- `likelihood` A vector of log likelihood values. Each element in the vector is the log likelihood of the accepted graph.
- `stepSize` The number of iterations discarded between each iteration that is kept.
- `time` The runtime of the Metropolis-Hastings algorithm in seconds.

`drosophila` *Tissue type and transcription factor binding data during Drosophila mesoderm development*

Description

Zinzen et al. (2009) measured in vivo transcription factor binding for five key transcription factors using ChIP-chip assays at two hour intervals during drosophila mesoderm development. The five transcription factors are: Twist (Twi), Tinman (Tin), Myocyte enhancing factor 2 (Mef2), Bagpipe (Bap), and Biniou (Bin). Both Twi and Tin were assayed from 2-8 hours, Mef2 from 2-12 hours, Bap from 6-8 hours, and Bin from 6-12 hours. In addition, Zinzen et al. identified six tissue types based on tissue specific expression: mesoderm (Meso), somatic muscle (SM), visceral muscle (VM), cardiac muscle (CM), mesoderm and somatic muscle (Meso&SM), and somatic and visceral muscle (SM&VM). All data are binary and measured at 310 cis-regulatory modules.

Usage

```
data(drosophila)
```

Format

An object of class "list" containing tissue type and transcription factor binding data. The first element in the list is a matrix which contains the data in the original form. The second element in the list is a matrix that contains the binary data. For this matrix any value greater than zero to changed to a one. For both data sets the tissue type data appears in the first six columns of the data matrix and the remaining 15 columns contain the transcription factor binding data.

Source

https://www.nature.com/articles/nature08531#Sec22

References

Zinzen, R., Girardot, C., Gagneur, J. et al. Combinatorial binding predicts spatio-temporal cis-regulatory activity. Nature 462, 65–70 (2009) doi:10.1038/nature08531

Examples

```
# Load the data.
data(drosophila)

# Display the first 5 rows and 8 columns of the continuous data matrix.
drosophila$continuous[1:5, 1:8]

# Display the first 5 rows and 8 columns of the discrete data matrix.
drosophila$discrete[1:5, 1:8]
```

geuvadis *Genotype and gene expression data from the GEUVADIS project*

Description

The GEUVADIS (Genetic European Variation in Disease) project (Lappalainen et al., 2013) measured the gene expression in Lymphoblastoid Cell Lines (LCLs) from a subset of individuals from the 1,000 Genomes Project (Auton et al., 2015), which measured the genotypes of individuals from multiple ethnicities. This data set contains genotype and gene expression data from 373 European individuals. The data are divided into 46 sets which each contain data from an expression quantitative trait locus (eQTL) and its associated genes.

Usage

```
data(geuvadis)
```

Format

An object of class "`list`" containing 46 eQTL-gene sets. Each element of the list is a matrix with the observations down the rows and the eQTL and genes across the columns. The first column in the matrix contains the genotype (eQTL) data and the remaining columns contain the gene expression data.

Source

https://www.ebi.ac.uk/arrayexpress/files/E-GEUV-1/analysis_results/

References

Lappalainen, T., Sammeth, M., Friedländer, M. et al. Transcriptome and genome sequencing uncovers functional variation in humans. Nature 501, 506–511 (2013) doi:10.1038/nature12531

Auton, A., Abecasis, G., Altshuler, D. et al. A global reference for human genetic variation. Nature 526, 68–74 (2015) doi:10.1038/nature15393

Examples

```
# Load the data.
data(geuvadis)

# Display the first 6 rows of the eQTL-gene set Q8.
head(geuvadis$Q8)
```

`mhEdge` *mhEdge*

Description

The main function for the Metropolis-Hastings algorithm. It returns the posterior distribution of the edge directions.

Usage

```
mhEdge(
  data,
  adjMatrix,
  prior = c(0.05, 0.05, 0.9),
  nCPh = 0,
  nGV = 0,
  pmr = FALSE,
  burnIn = 0.2,
  iterations = 1000,
  thinTo = 200,
  progress = TRUE
)
```

Arguments

`data` A matrix with the variables across the columns and the observations down the rows. If there are genetic variants in the data these variables must come before the remaining variables. If there are clinical phenotypes in the data these variables must come after all other variables. For example, if there is a data set with one genetic variant variable, three gene expression variables, and one clinical phenotype variable the first column in the data matrix must contain the genetic variant data, the next three columns will contain the gene expression data, and the last column will contain the clinical phenotype data.

`adjMatrix` An adjacency matrix indicating the edges that will be considered by the Metropolis-Hastings algorithm. An adjacency matrix is a matrix of zeros and ones. The ones represent an edge and its direction between two nodes.

`prior` A vector containing the prior probability for the three edge states.

`nCPh` The number of clinical phenotypes in the graph.

`nGV` The number of genetic variants in the graph.

`pmr` Logical. If true the Metropolis-Hastings algorithm will use the Principle of Mendelian Randomization (PMR). This prevents the direction of an edge pointing from a gene expression or a clinical phenotype node to a genetic variant node.

`burnIn` A number between 0 and 1 indicating the percentage of the sample that will be discarded.

`iterations` An integer for the number of iterations to run the MH algorithm.

`thinTo` An integer indicating the number of observations the chain should be thinned to.

`progress` Logical. If TRUE the runtime in seconds for the cycle finder and log likelihood functions and a progress bar will be printed.

Value

An object of class baycn containing 9 elements:

- burnIn – The percentage of MCMC iterations that will be discarded from the beginning of the chain.
- chain – A matrix where each row contains the vector of edge states for the accepted graph.
- decimal – A vector of decimal numbers. Each element in the vector is the decimal of the accepted graph.
- iterations – The number of iterations for which the Metropolis-Hastings algorithm is run.
- posteriorES – A matrix of posterior probabilities for all three edge states for each edge in the network.
- posteriorPM – A posterior probability adjacency matrix.
- likelihood – A vector of log likelihood values. Each element in the vector is the log likelihood of the accepted graph.
- stepSize – The number of MCMC iterations discarded between each accepted graph.
- time – The runtime of the Metropolis-Hastings algorithm in seconds.

References

Martin, E. A. and Fu, A. Q. (2019). A Bayesian approach to directed acyclic graphs with a candidate graph. *arXiv preprint arXiv:1909.10678*.

Examples

```
# Generate data under topology m1_gv.
# Use ?simdata for a description and graph of m1_gv.
data_m1 <- simdata(b0 = 0,
                   N = 200,
                   s = 1,
                   graph = 'm1_gv',
                   ss = 1,
                   q = 0.27)

# Create an adjacency matrix with the true edges.
am_m1 <- matrix(c(0, 1, 0,
                  0, 0, 1,
                  0, 0, 0),
                byrow = TRUE,
                nrow = 3)

# Run the Metropolis-Hastings algorithm on the data from m1_gv using the
# Principle of Mendelian Randomization (PMR) and the true edges as the input.
mh_m1_pmr <- mhEdge(data = data_m1,
                    adjMatrix = am_m1,
                    prior = c(0.05,
                              0.05,
                              0.9),
                    nCPh = 0,
                    nGV = 1,
                    pmr = TRUE,
                    burnIn = 0.2,
                    iterations = 1000,
                    thinTo = 200,
                    progress = FALSE)

summary(mh_m1_pmr)

# Generate data under topology gn4.
# Use ?simdata for a description and graph of gn4.
data_gn4 <- simdata(b0 = 0,
                    N = 200,
                    s = 1,
                    graph = 'gn4',
                    ss = 1)

# Create an adjacency matrix with the true edges.
am_gn4 <- matrix(c(0, 1, 1, 0,
                   0, 0, 0, 1,
                   0, 0, 0, 0,
                   0, 0, 1, 0),
                 byrow = TRUE,
                 nrow = 4)

# Run the Metropolis-Hastings algorithm on the data from gn4 with the true
# edges as the input.
mh_gn4 <- mhEdge(data = data_gn4,
                 adjMatrix = am_gn4,
                 prior = c(0.05,
                           0.05,
                           0.9),
                 nCPh = 0,
                 nGV = 0,
                 pmr = FALSE,
                 burnIn = 0.2,
                 iterations = 1000,
                 thinTo = 200,
                 progress = FALSE)

summary(mh_gn4)
```

`mse` *mse*

Description

Calculates either the edge-wise or whole graph MSE. If more than one data set is input the mean and standard deviation of the MSE will be calculated.

Usage

```
mse(posterior, expected, adjMatrix, type = "emse")
```

Arguments

`posterior` A list of baycn objects or posterior probability adjacency matrices. Each element in posterior can also be a list. For example, if multiple data sets were generated under the same simulation scheme and baycn was run on each data set, the input list could be a list containing the output from each run of baycn. In this scenario the mean and standard deviation of of the edgewise or whole graph MSE would be output.

`expected` The expected edge state probabilities for each edge. This must be a matrix with the expected probability for the three edge states across the columns and each edge down the rows.

`adjMatrix` This is only required if the input to posterior is a posterior probability adjacency matrix (for example, output from the BiDAG or structmcmc pacakges). This matrix will be used to convert the posterior probability adjacency matrix to an edge state matrix. Only the edges in adjMatrix will be converted to the edge state matrix. To consider all possible edges use a fully connected adjacency matrix.

`type` A character string indicating the type of MSE to be calculated. The two options are ’emse’ - edge-wise MSE or ’gmse’ - whole graph MSE.

Examples

```
set.seed(3)

# Generate data from topology M2.
data_m2 <- simdata(graph = 'm2_ge',
                   N = 200,
                   b0 = 0,
                   ss = 1,
                   s = 1)

# Adjacency matrix for topology M2
am_m2 <- matrix(c(0, 1, 1,
                  0, 0, 1,
                  0, 0, 0),
                byrow = TRUE,
                nrow = 3)

# Run baycn on the data from topology M2.
baycn_m2 <- mhEdge(data = data_m2,
                   adjMatrix = am_m2,
                   prior = c(0.05,
                             0.05,
                             0.9),
                   nCPh = 0,
                   nGV = 0,
                   pmr = FALSE,
                   iterations = 1000,
                   burnIn = 0.2,
                   thinTo = 500,
                   progress = FALSE)

# Expected probabilities for topology M2.
ep_m2 <- matrix(c(1, 0, 0,
                  0, 0, 1,
                  0, 1, 0),
                byrow = TRUE,
                ncol = 3)

# Calculate the edgewise MSE.
emse_m2 <- mse(posterior = list(baycn_m2),
               expected = ep_m2,
               type = 'emse')


# When a list with two levels is passed posterior the mean and standard
# deviation of the MSE is returned. Below is an example demonstrating this
# feature.
# Adjacency matrix for topology M2
am_m2 <- matrix(c(0, 1, 1,
                  0, 0, 1,
                  0, 0, 0),
                byrow = TRUE,
                nrow = 3)

# Create a list to hold multiple M2 data sets.
data_m2 <- vector(mode = 'list',
                  length = 5)

# Create a list to hold the output from baycn.
baycn_m2 <- vector(mode = 'list',
                   length = 5)

for (e in 1:5) {

  # Generate data for topology M2.
  data_m2[[e]] <- simdata(graph = 'm2_ge',
                          N = 200,
                          b0 = 0,
                          ss = 1,
                          s = 1)

  # Run baycn on the data simulated for topology M2.
  baycn_m2[[e]] <- mhEdge(data = data_m2[[e]],
                          adjMatrix = am_m2,
                          prior = c(0.05,
                                    0.05,
                                    0.9),
                          nCPh = 0,
                          nGV = 0,
                          pmr = FALSE,
                          iterations = 1000,
                          burnIn = 0.2,
                          thinTo = 500,
                          progress = FALSE)
}

# Expected probabilities for topology M2.
ep_m2 <- matrix(c(1, 0, 0,
                  0, 0, 1,
                  0, 1, 0),
                byrow = TRUE,
                ncol = 3)

# Calculate the edgewise MSE
emse_m2 <- mse(posterior = list(baycn_m2),
               expected = ep_m2,
               type = 'emse')
```

plot,baycn-method *plot*

Description

plot

Usage

```
## S4 method for signature 'baycn'
plot(
  x,
  presence = 0.4,
  direction = 0.2,
  edgeLabel = TRUE,
  mode = "directed",
  weighted = TRUE,
  ...
)
```

Arguments

`x`	An object of class baycn.
`presence`	A scalar between 0 and 1. This is the cutoff for considering an edge to be present. For example, if presence = 0.4 then an edge is considered to be present if the sum of the posterior proability for the two edge directions is greater than 0.4. The edge will be considered to be absent if this sum is less than 0.4.
`direction`	A scalar between 0 and 1. This is the cutoff for determining the direction of an edge. For example, if direction = 0.2 then an edge is considered to be directed if the difference between the posterior proability for the two edge directions is greater than 0.2. An edge will be considered undirected if the difference is less than 0.2.
`edgeLabel`	Logical - indicates whether the posterior probabilities should be included as edge labels in the plot. If edgeLabel is TRUE then weighted must also be set to TRUE.
`mode`	See `graph_from_adjacency_matrix` for details.
`weighted`	See `graph_from_adjacency_matrix` for details.
`...`	Other Arguments passed to plot.igraph.

prerec *prerec*

Description

Calculates the precision and recall (i.e., power) of the inferred graph.

Usage

```
prerec(amInferred, amTrue, cutoff)
```

Arguments

amInferred	A baycn object or a posterior probability adjacency matrix.
amTrue	The undirected adjacency matrix of the true graph. This will be a symmetric matrix with 0s along the diagonal.
cutoff	A number between 0 and 1 indicating the posterior probability threshold for considering an edge present.

Value

A list. The first element is the precision and the second element is the recall of the inferred graph.

Examples

```

set.seed(5)

# Generate data from topology GN4.
data_gn4 <- simdata(graph = 'gn4',
                    N = 200,
                    b0 = 0,
                    ss = 1,
                    s = 1)

# Adjacency matrix for topology GN4 - all possible edges.
am_gn4 <- matrix(c(0, 1, 1, 1,
                   0, 0, 1, 1,
                   0, 0, 0, 1,
                   0, 0, 0, 0),
                 byrow = TRUE,
                 nrow = 4)

# Run baycn on the data from topology GN4.
baycn_gn4 <- mhEdge(data = data_gn4,
                    adjMatrix = am_gn4,
                    prior = c(0.05,
                              0.05,
                              0.9),
                    nCPh = 0,
                    nGV = 0,
                    pmr = FALSE,
                    iterations = 1000,
                    burnIn = 0.2,
                    thinTo = 500,
                    progress = FALSE)

# Adjacency matrix with the true edges for topology GN4.
am_gn4_true <- matrix(c(0, 1, 1, 0,
                        1, 0, 0, 1,
                        1, 0, 0, 1,
                        0, 1, 1, 0),
                      byrow = TRUE,
                      nrow = 4)

# Calculate the precision and recall.
prerec_gn4 <- prerec(amInferred = baycn_gn4,
                     amTrue = am_gn4_true,
                     cutoff = 0.4)
```

show,baycn-method *show*

Description

show

Usage

```
## S4 method for signature 'baycn'
show(object)
```

Arguments

object	An object of class baycn.

simdata *simdata*

Description

A function for simulating data under various topologies for continuous and mixed data.

Usage

```
simdata(
  graph = "gn4",
  N = 500,
  b0 = 0,
  ss = 1,
  s = 1,
  p = 0.6,
  q = 0.45,
  ssc = 0.2,
  nConfounding = 2
)
```

Arguments

`graph` A character string of the graph for which data will be simulated. The graphs that can be chosen are m1_ge, m1_gv, m1_cp, m1_cc, m1_iv, m2_ge, m2_gv, m2_cp, m2_cc, m2_iv, m3_gv, m3_cp, m3_cc, m3_iv, mp_ge, mp_gv, gn4, gn5, gn8, gn11, layer, layer_cp, layer_iv, and star.

The following figures show the graph for each of the topologies listed above. The nodes with a circle around the name are normally distributed and the nodes with a diamond around the name are distributed multinomial. The nodes labeled with a C represent confounding variables. The Principle of Mendelian Randomization (PMR) can be used on graphs with discrete and continuous random variables. This introduces the constraint that the continuous random variables cannot be parents of discrete random variables and edges between these types of variables only have two states: absent and directed with the discrete random variable as the parent.

m1_ge - Topology M1 with three continuous random variables. In this case M1 has two other Markov equivalent graphs.

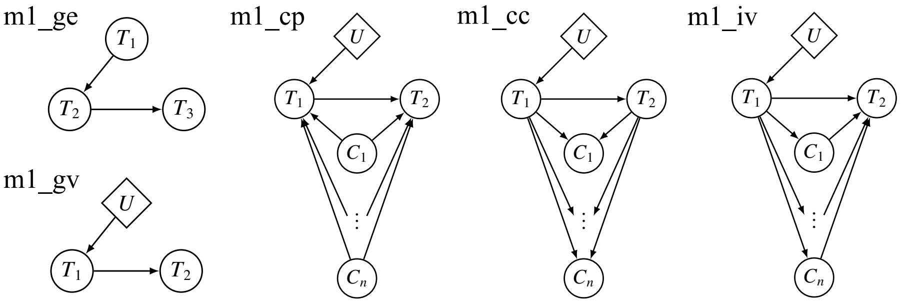
m1_gv - Topology M1 with one discrete random variable U and two continuous random variables. When using the PMR this graph does not have any other Markov equivalent graphs.

We consider three types of confounding variables (Yang et al., 2017):

m1_cp - Topology M1 with n common parent confounding variables.

m1_cc - Topology M1 with n common child confounding variables.

m1_iv - Topology M1 with n intermediate confounding variables.



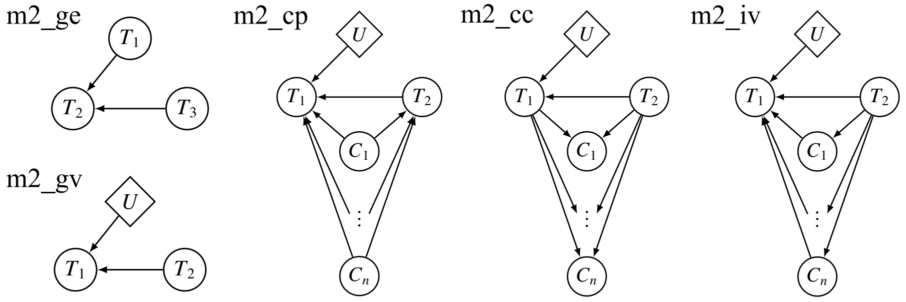
m2_ge - Topology M2 with three continuous random variables. This graph is a v structure and does not have any other Markov equivalent graphs.

m2_gv - Topolog M2 with one discrete random variable U and two continuous random variables. This graph is a v structure and does not have any other Markov equivalent graphs.

m2_cp - Topology M2 with n common parent confounding variables.

m2_cc - Topology M2 with n common child confounding variables.

m2_iv - Topology M2 with n intermediate confounding variables.

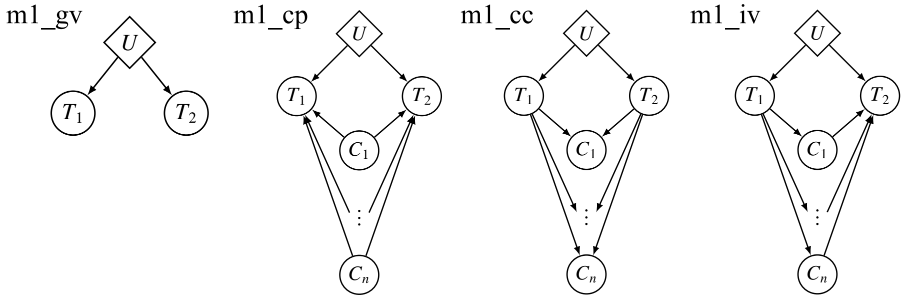


m3_gv - Topolog M3 with one discrete random variable U and two continuous random variables.

m3_cp - Topology M3 with n common parent confounding variables.

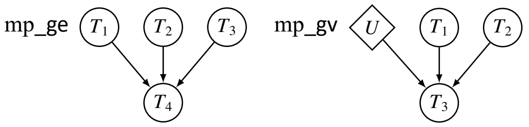
m3_cc - Topology M3 with n common child confounding variables.

m3_iv - Topology M3 with n intermediate confounding variables.



mp_ge - The multi-parent topology with continuous random variables. This graph is made up of multiple v structures and has no other Markov equivalent graphs.

mp_gv - The multi-parent topology with one discrete random variable. This graph is made up of multiple v structures and has no other Markov equivalent graphs.

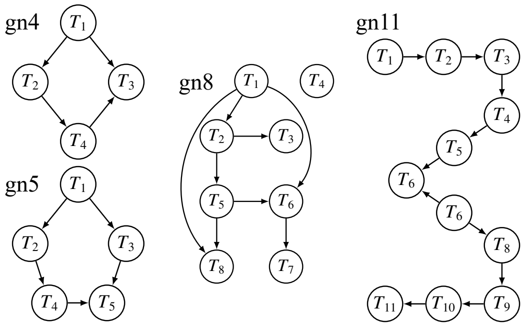


gn4 - Topology GN4 is formed by combining topologies M1 and M2. The Markov equivalence class for this topology is made up of three different graphs.

gn5 - Topology GN5 has three other Markov equivalent graphs.

gn8 - Topology GN8 has three overlapping cycles, two v structures, and two other Markov equivalent graphs.

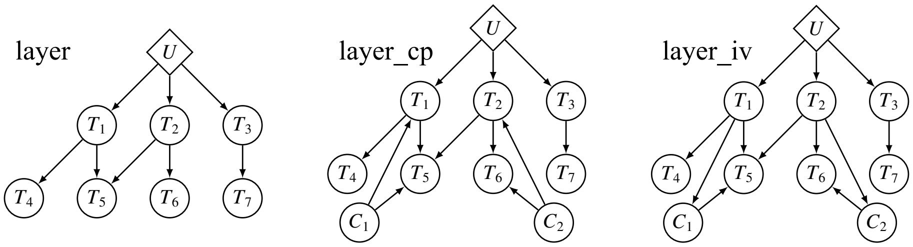
gn11 - Topology GN11 has two sub-graphs separated by a v structure at node T6.



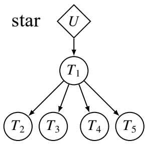
layer - The layer topology has no other Markov equivalent graphs when using the PMR and is made up of multiple M1 topologies.

layer_cp - The layer topology with 2 common parent confounding variables.

layer_iv - The layer topology with 2 intermediate confounding variables.



star - The star topology has no other Markov equivalent graphs when using the PMR and is made up of multiple M1 topologies.



`N` The number of observations to simulate.

`b0` The mean of the variable if it does not have any parents. If the variable has one or more parents it is the slope in the linear model that is the mean of the normally distributed variables.

`ss` The coefficient of the parent nodes (if there are any) in the linear model that is the mean of the normally distributed variables. This coefficient is referred to as the signal strength.

`s` The standard deviation of the normal distribution.

`p` The probability of success for a binomial random variable.

`q` The frequency of the reference allele.

`ssc` The signal strength of the confounding variables.

`nConfounding` The number of confounding variables to simulate.

Value

A matrix with the variables across the columns and the observations down the rows.

References

Yang, F., Wang, J., The GTEx Consortium, Pierce, B. L., and Chen, L. S. (2017). Identifying cis-mediators for trans-eQTLs across many human tissues using genomic mediation analysis. *Genome Res.* 27, 1859-1871.

Examples

```
# Generate data under topology GN4.
data_gn4 <- simdata(graph = 'gn4',
                    N = 500,
                    b0 = 1,
                    ss = 1,
                    s = 1)

# Display the first few rows of the data.
data_gn4[1:5, ]

# Generate data under topology M1 with 3 intermediate confounding variables.
data_m1_iv <- simdata(graph = 'm1_iv',
                      N = 500,
                      b0 = 0,
                      ss = 1,
                      s = 1,
                      q = 0.1,
                      ssc = 0.2,
                      nConfounding = 3)

# Show the first few rows of the data.
data_m1_iv[1:5, ]
```

summary,baycn-method *summary*

Description

summary

Usage

```
## S4 method for signature 'baycn'
summary(object, ...)
```

Arguments

`object`	An object of class baycn.
`...`	Other Arguments passed to methods.

`tracePlot` *tracePlot*

Description

Creates a trace plot of the log likelihood and graph decimal. The graph decimal is a decimal number calculated from the vector of edge states for the accepted graph.

Usage

```
tracePlot(x)
```

Arguments

`x`	An object of class baycn.

Index