Package 'BBNI'

July 15, 2026

Title Bayesian Inference of Boolean Genetic Networks

Version 0.1.1

Description Implements a fully Bayesian Markov chain Monte Carlo (MCMC) approach for inferring the topology and Boolean logic transition functions of gene regulatory networks from noisy, binary time-series expression data. Network structure and Boolean rules are sampled jointly from their posterior distribution, providing principled uncertainty quantification rather than a single point estimate. Method described in Han et al. (2014) <doi:10.1371/journal.pone.0115806>.

Depends R (>= 4.1.0)

License BSD_3_clause + file LICENSE

URL https://anson-li8.github.io/BBNI/,
https://github.com/anson-li8/BBNI

BugReports https://github.com/anson-li8/BBNI/issues

Encoding UTF-8

Language en-US

RoxygenNote 8.0.0

Imports bitops, stats

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Anson Li [aut, cre],
Shengtong Han [aut]

Maintainer Anson Li <liyuanrui618@gmail.com>

Repository CRAN

Date/Publication 2026-07-15 17:50:18 UTC

Contents

GenerateNetwork 2
GenerateSample 3
run_bbni 4

Index **7**

`GenerateNetwork` *Generate Initial Network Topology*

Description

Randomly generates a valid directed acyclic graph (DAG) topology T and assigns a corresponding Boolean transition function F to each node. The algorithm samples parent set configurations, keeping the constraint that the maximum in-degree for any node is 2, and further ensures the resulting structure does not contain directed cyclic loops.

Usage

```
GenerateNetwork(num.node)
```

Arguments

`num.node`	An integer representing the total number of genes/nodes in the network.

Value

A square transition function matrix combining the initial DAG topology with randomly assigned Boolean logic functions. Elements with a value of 0 indicate no directed edge, while positive integers indicate the presence of an edge and specify the defining Boolean function type (1-14).

Examples

```
# Generate a true network topology and Boolean rules for 5 nodes
set.seed(123)
true_network <- GenerateNetwork(num.node = 5)
print(true_network)
```

GenerateSample *Simulate Time-Series Observation Dataset*

Description

Simulates a synthetic time-series observation dataset (G). It starts by running independent Bernoulli trials on root nodes. The remaining non-root nodes are calculated from time $t-1$ to time t using their assigned Boolean logic functions. A pre-generated binary noise matrix is applied via a bitwise XOR operation to occasionally flip the Boolean outputs, injecting natural biological noise expected by the model.

Usage

```
GenerateSample(trans_matrix, num.node, SampleSize, para, error)
```

Arguments

trans_matrix	A square matrix combining the network topology T and integer-coded Boolean logic functions F assigned to each directed edge.
num.node	An integer representing the total number of network nodes.
SampleSize	An integer representing the total number of time points to simulate.
para	A numeric vector of baseline success probabilities (θ_i) used to generate the expression states of root nodes via independent Bernoulli trials.
error	A pre-generated binary noise matrix applied to occasionally flip Boolean outputs, injecting natural noise.

Value

A simulated binary gene expression matrix G, where rows represent individual genes/nodes and columns represent sequential points in time.

Examples

```
# 1. Generate a 5-node network
set.seed(123)
num_nodes <- 5
sample_size <- 10
true_network <- GenerateNetwork(num.node = num_nodes)

# 2. Set baseline probabilities and simulate zero-noise error matrix
root_probs <- rep(0.5, num_nodes)
error_matrix <- matrix(0, nrow = num_nodes, ncol = sample_size)

# 3. Generate the synthetic time-series data
dummy_data <- GenerateSample(
  trans_matrix = true_network,
  num.node = num_nodes,
```

```
  SampleSize = sample_size,
  para = root_probs,
  error = error_matrix
)
print(dummy_data)
```

`run_bbni` *Execute Metropolis-within-Gibbs MCMC Sampler for Boolean Networks*

Description

Executes a Metropolis-within-Gibbs Markov chain Monte Carlo (MCMC) algorithm to sample from the joint posterior distribution of Directed Acyclic Graph (DAG) topologies (T) and Boolean logic transition functions (F). The algorithm iterates through individual network nodes and proposes parent set mutations (edge additions, removals, or swaps) paired with transition function reassignments to one of 14 candidate Boolean rules. Proposed states transitions are strictly verified to follow the DAG constraint and evaluated with a Metropolis-Hastings acceptance threshold using log-posterior values.

Usage

```
run_bbni(
  GeneData,
  num.node,
  SampleSize,
  prior_para,
  num_update,
  penalty,
  prop.ratio,
  verbose = FALSE
)
```

Arguments

`GeneData`	A binary empirical observation matrix of the binary expression data (G).
`num.node`	An integer representing the total number of network nodes.
`SampleSize`	An integer representing the total number of time points in the dataset.
`prior_para`	A matrix of Beta prior hyperparameters α and β for root node probabilities and the global noise parameter e.
`num_update`	An integer representing the total number of MCMC iterations to perform.
`penalty`	A numeric value representing the structural prior probability per edge used to penalize network complexity $P(T)$.
`prop.ratio`	A numeric probability threshold used to decide whether to sample a move from the empirical proposal distribution or a uniform random distribution.
`verbose`	Logical. If TRUE, prints verbose MCMC iteration progress to the console. Default is FALSE.

Value

A list containing the full trajectory of the MCMC chain. Specifically, `networks` (a list of sampled transition function matrices) and `log_posterior` (a numeric vector of log-posterior scores for each iteration). These represent samples drawn from the marginal posterior distribution $P(T, F|G)$ used for Bayesian model averaging.

Examples

```
# 1. Define network parameters
set.seed(235)
num_nodes <- 10
sample_size <- 50

# 2. Generate true network and simulate data
true_network <- GenerateNetwork(num.node = num_nodes)

# Set up Beta priors for root-node probabilities and the noise rate
prior_para <- matrix(3, nrow = num_nodes + 1, ncol = 2)
prior_para[num_nodes + 1, 1] <- 2
prior_para[num_nodes + 1, 2] <- 100

# Simulate parameters
para <- numeric(num_nodes + 1)
for (i in 1:(num_nodes + 1)) {
  para[i] <- stats::rbeta(1, prior_para[i, 1], prior_para[i, 2])
}
para[num_nodes + 1] <- 0.1 # Fixed noise rate for simulation

error_matrix <- matrix(stats::rbinom(num_nodes * sample_size, 1, para[num_nodes + 1]),
  nrow = num_nodes, ncol = sample_size
)

dummy_data <- GenerateSample(
  trans_matrix = true_network,
  num.node = num_nodes,
  SampleSize = sample_size,
  para = para,
  error = error_matrix
)

# 3. Run the MCMC sampler (silently)
mcmc_results <- run_bbni(
  GeneData = dummy_data,
  num.node = num_nodes,
  SampleSize = sample_size,
  prior_para = prior_para,
  num_update = 100, # Scaled down for example speed
  penalty = 0.1,
  prop.ratio = 0.1
)

# 4. Inspect results
```

```
tail(mcmc_results$log_posterior)
```

Index

GenerateNetwork, 2
GenerateSample, 3

run_bbni, 4