Package ‘BayesNetBP’

June 1, 2021

Type Package

Title Bayesian Network Belief Propagation

Version 1.5.9

Date 2021-06-01

Author Han Yu, Rachael Blair, Janhavi Moharil, Andrew Yan

Maintainer Han Yu <hyu9@buffalo.edu>

Depends R (>= 3.6.0), stats, utils

Imports igraph, RColorBrewer, fields, doBy, methods, graph, bnlearn, graphics

Suggests Rgraphviz, shiny, googleVis, cyjShiny, qtl, qtlnet

BugReports https://github.com/hyu-ub/BayesNetBP/issues

LazyData true

Description Belief propagation methods in Bayesian Networks to propagate evidence through the network. The implementation of these methods are based on the article: Cowell, RG (2005). Local Propagation in Conditional Gaussian Bayesian Networks <https://www.jmlr.org/papers/volume6/cowell05a/>. For details please see Yu et. al. (2020) BayesNetBP: An R Package for Probabilistic Reasoning in Bayesian Networks <doi:10.18637/jss.v094.i03>. The optional 'cyjShiny' package for running the Shiny app is available at <https://github.com/cytoscape/cyjShiny>. Please see the example in the documentation of 'runBayesNetApp' function for installing 'cyjShiny' package from GitHub.

License GPL (>= 2)

RoxygenNote 7.1.1

NeedsCompilation no

Repository CRAN

Date/Publication 2021-06-01 15:50:13 UTC
AbsorbEvidence

Description
Absorb evidence into the model

Usage
AbsorbEvidence(tree, vars, values)

Arguments
- tree: a ClusterTree object
- vars: a vector of the names of observed variables
- values: a list of observed values of the variables. Aside from a single value, the element of the list can also be a vector of likelihood values
Details

Absorb multiple types and pieces of evidences into a **ClusterTree** object. The discrete compartment of the **ClusterTree** will be automatically propagated after evidence absorption, so that the object will be ready for making queries and absorbing additional evidence.

Value

**ClusterTree** object with the evidence absorbed

Author(s)

Han Yu

References


Examples

```r
data(liver)
tree.init.p <- Initializer(dag=liver$dag, data=liver$data,
                          node.class=liver$node.class,
                          propagate = TRUE)
tree.post <- AbsorbEvidence(tree.init.p, c("Nr1i3", "chr1_42.65"), list(1,"1"))
```

---

**bn_to_graphNEL**

Convert a **bn** object to graphNEL object

Description

Convert a **bn** object to graphNEL object while removing isolated nodes

Usage

```r
bn_to_graphNEL(graph_bn)
```

Arguments

- **graph_bn**: a **bn** object of Bayesian network
Value

a graphNEL object

Author(s)

Han Yu

---

A simulated data from the Chest Clinic example

---

Description

Simulated data from the Chest Clinic example (also known as the Asia example) from Lauritzen and Spiegelhalter, 1988.

Usage

data(chest)

Format

The data set chest contains two objects:

data  a data.frame object of 10000 observations and 8 discrete variables.
dag  a graphNEL object specifying the network structure.

References


---

An S4 class of the cluster tree.

Description

The ClusterTree object is the computational object for belief propagation.
**ClusterTreeCompile**

*Description*

Get the cluster sets and strong semi-elimination tree from the Bayesian network

*Usage*

`ClusterTreeCompile(dag, node.class)`

*Arguments*

- **dag**
  - a graphNEL object of the Bayesian network
- **node.class**
  - a named vector of logical values, TRUE if node is discrete, FALSE if otherwise
Details

This function forms the cluster sets and the semi-elimination tree graph from the Bayesian network. The procedures include acquiring the elimination order, moralization, triangulation, obtaining cluster sets, forming strong elimination tree and strong semi-elimination tree. The cluster sets and the semi-elimination tree are required to initialize the cluster tree.

Value

tree.graph  a graphNEL object of semi-elimination tree.
dag  a graphNEL object of original Bayesian network.
cluster.sets a list of members of each cluster.
node.class  a named vector of logical values, TRUE if node is discrete, FALSE if otherwise
elimination.order  a vector of node names sorted by the elimination order.

Author(s)

Han Yu

References


See Also

ElimTreeInitialize

Examples

data(liver)
cst <- ClusterTreeCompile(dag=liver$dag, node.class=liver$node.class)

ComputeKLDs  Compute signed and symmetric Kullback-Leibler divergence

Description

Compute signed and symmetric Kullback-Leibler divergence of variables over a spectrum of evidence
**Usage**

```r
ComputeKLDs(
  tree, var0, vars, seq, 
  pbar = TRUE, method = "gaussian", 
  epsilon = 10^-6
)
```

**Arguments**

- `tree`: a `ClusterTree` object
- `var0`: the variable to have evidence absorbed
- `vars`: the variables to have divergence computed
- `seq`: a vector of numeric values as the evidences
- `pbar`: logical(1) whether to show progress bar
- `method`: method for divergence computation: `gaussian` for Gaussian approximation, for Monte Carlo integration
- `epsilon`: numeric(1) the KL divergence is undefined if certain states of a discrete variable have probabilities of 0. In this case, a small positive number `epsilon` is assigned as their probabilities for calculating the divergence. The probabilities of other states are shrunked proportionally to ensure they sum up to 1.

**Details**

Compute signed and symmetric Kullback-Leibler divergence of variables over a spectrum of evidence. The signed and symmetric Kullback-Leibler divergence is also known as Jeffery’s signed information (JSI) for continuous variables.

**Value**

a `data.frame` of the divergence

**Author(s)**

Han Yu

**References**


Examples

```r
## Not run:
data(liver)
tree.init.p <- Initialize(dag=liver$dag, data=liver$data,
                          node.class=liver$node.class,
                          propagate = TRUE)
klds <- ComputeKLDs(tree=tree.init.p, var0="Nr1i3",
                     vars=setdiff(tree.init.p@node, "Nr1i3"),
                     seq=seq(-3,3,0.5))
head(klds)
## End(Not run)
```

---

ElimTreeInitialize  Initialize the elimination tree

Description

Initialize the elimination tree with the local models

Usage

```r
ElimTreeInitialize(tree, dag, model, node.sets, node.class)
```

Arguments

- `tree`: a graphNEL object of the elimination tree
- `dag`: a graphNEL object of the Bayesian network
- `model`: a list of local models built from `LocalModelCompile` function
- `node.sets`: a list of cluster sets obtained from `ClusterTreeCompile` function
- `node.class`: a named vector of logical values, TRUE if node is discrete, FALSE if otherwise

Details

Initialize the elimination tree with the local models

Value

- `ClusterTree` object with the local models incorporated

Author(s)

- Han Yu
emission

References


See Also

The functions `ClusterTreeCompile` and `LocalModelCompile` provide necessary objects to obtain `ClusterTree` object by initializing the elimination tree through this function.

Examples

data(liver)
cst <- ClusterTreeCompile(dag=liver$dag, node.class=liver$node.class)
models <- LocalModelCompile(data=liver$data, dag=liver$dag, node.class=liver$node.class)
tree.init <- ElimTreeInitialize(tree=cst$tree.graph,
                                dag=cst$dag,
                                model=models,
                                node.sets=cst$cluster.sets,
                                node.class=cst$node.class)

emission A ClusterTree Example of Emission Model

Description

A propagated `ClusterTree` object named emission. This model contains nine variables, including three discrete: Filter State (Fs), Waste Type (W), Burning Regimen (B) and six continuous variables: Metals in Waste (Min), Metals Emission (Mout), Filter Efficiency (E), Dust Emission (D), CO2 Concentration in Emission (C), Light Penetrability (L).

Usage

data(emission)

Format

The data set contains a propagated `ClusterTree` object emission ready for evidence absorption and making queries.

References

emission1000  A simulated data from the Emission example

Description
Simulated data from the Emission example (also known as the Waste Incinerator example)

Usage
data(emission1000)

Format
The data set emission1000 contains two objects:

data  a data.frame object of 1000 observations and 3 discrete variables and 6 continuous variables.
dag  a graphNEL object specifying the network structure.

References

FactorQuery  Queries of discrete variable distributions

Description
Obtain the joint, marginal, and conditional distributions of discrete variables

Usage
FactorQuery(tree, vars = c(), mode = c("joint", "conditional", "list"))

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tree</td>
<td>a ClusterTree object</td>
</tr>
<tr>
<td>vars</td>
<td>the variables to be queried</td>
</tr>
<tr>
<td>mode</td>
<td>type of desired distribution</td>
</tr>
</tbody>
</table>
GetValue

Details

Query the joint distribution of any combination of discrete variables when mode is "joint", or conditional distribution of a discrete variable. The mode "list" return a list of variable combinations, such that joint distributions of any subset of them are ready for extraction. Queries outside this list are also supported but may take longer computing time. This function will also return marginal distribution if only one variable is queried.

Value
data.frame object specifying a joint or conditional distribution.

Author(s)
Han Yu

References


Examples

data(chest)
dag <- chest$dag
node.class <- rep(TRUE, length(dag@nodes))
names(node.class) <- dag@nodes
tree.init.p <- Initializer(dag=dag, data=chest$data,
node.class=node.class,
propagate=TRUE)

# joint distribution
FactorQuery(tree=tree.init.p, vars=c("tub", "xray", "dysp", "asia"), mode="joint")

# conditional distribution
FactorQuery(tree=tree.init.p, vars=c("xray"), mode="conditional")

GetValue

Possible values of a discrete variable

Description

Obtain all the possible values of a discrete variable.
Initializer

Usage

GetValue(tree, var, message = TRUE)

Arguments

tree       a ClusterTree object
var        the variables to be queried
message    type of desired distribution

Value

a vector of the possible values of discrete variable. If the variable is continuous, the returned value will be NULL.

Author(s)

Han Yu

Examples

data(toytree)
GetValue(toytree, "HDL")

Description

Initialize a ClusterTree object

Usage

Initializer(dag, data, node.class, propagate = TRUE)

Arguments

dag         a graphNEL object of the Bayesian network
data        a data.frame object
node.class  a named vector of logical values, TRUE if node is discrete, FALSE if otherwise
propagate   logical TRUE if the discrete part of the ClusterTree to be propagated

Details

A wrapper function to initialize a ClusterTree object. It combines the functions of ClusterTreeCompile, LocalModelCompile, ElimTreeInitialize and Propagate, thus initialize the ClusterTree object in a single step.
**Value**

*ClusterTree* object

**Author(s)**

Han Yu

**References**


**See Also**

*ClusterTreeCompile, LocalModelCompile, ElimTreeInitialize, Propagate*

**Examples**

```r
data(liver)
tree.init.p <- Initializer(dag=liver$dag, data=liver$data,
  node.class=liver$node.class,
  propagate = TRUE)
```

---

**Description**

Liver QTL data was obtained from a F2 inner-cross between inbred MRL/MpJ and SM/J strains of mice.

**Usage**

```r
data(liver)
```

**Format**

The data set `liver` contains three objects: the data, a learned Bayesian network structure and vector specifying node type. The fields are described as follows:

- **data** a *data.frame* object that contains 280 samples (rows) and 15 variables: genotype data (genotype states at 5 SNP markers) and phenotype data (HDL levels and normalized expression values of 10 genes). Three of these phenotypes are dichotomized, including `Cyp2b10`, `Spgl1` and HDL. Genotypes and dichotomized phenotypes are of class *factor* and continuous phenotypes are of class *numeric*.

- **dag** a *graphNEL* object, which is the network structure learned by *qtlnet* package.

- **node.class** a named vector of logical values indicating whether each node is discrete.
LocalModelCompile

Description

Compile the local models

Usage

LocalModelCompile(data, dag = NULL, node.class = NULL)

Arguments

data a data.frame object or a qtlnet object
dag NULL if data is qtlnet object, or a graphNEL object of conditional Gaussian Bayesian network if data is data.frame.
node.class NULL if data is qtlnet object, or a vector of logical values named by node names, TRUE for discrete, FALSE for continuous variables if data is data.frame.

Details

This function compiles the local models, including the conditional probability tables for discrete variables, and linear predictor potentials for continuous variables. The qtlnet and qtl package need to be installed if data is a qtlnet object.

Value

pots a list of discrete potentials (conditional probability tables) for each discrete variable.
bags a list of sets of continuous potentials (lppotentials), each set for a continuous variables.

Author(s)

Han Yu

References


References

**Marginals**

**See Also**

`ElimTreeInitialize`

**Examples**

```r
data(liver)
models <- LocalModelCompile(data=liver$data, dag=liver$dag, node.class=liver$node.class)
```

---

**Marginals**

Obtain marginal distributions

**Description**

Get the marginal distributions of multiple variables

**Usage**

`Marginals(tree, vars)`

**Arguments**

- `tree` a `ClusterTree` object
- `vars` a vector of variables for query of marginal distributions

**Details**

Get the marginal distributions of multiple variables. The function `Marginals` returns a list of marginal distributions. The marginal distribution of a discrete variable is a named vector of probabilities. Meanwhile, the marginal distributions of continuous variables in a CG-BN model are mixtures of Gaussian distributions. To fully represent this information, the marginal of a continuous variable is represented by a `data.frame` with three columns to specify parameters for each Gaussian distribution in the mixture, which are:

- `mean` the mean value of a Gaussian distribution.
- `sd` the standard deviation of a Gaussian distribution.
- `n` the number of Gaussian mixtures

**Value**

- `marginals` a list of marginal distributions
- `types` a named vector indicating the types of the variables whose marginals are queried: TRUE for discrete, FALSE for continuous.
**Author(s)**
Han Yu

**References**


**See Also**
*PlotMarginals* for visualization of the marginal distributions, *SummaryMarginals* for summarization of the marginal distributions of continuous variables.

**Examples**

```r
data(liver)
tree.init.p <- Initializer(dag=liver$dag, data=liver$data,
                         node.class=liver$node.class,
                         propagate = TRUE)
tree.post <- AbsorbEvidence(tree.init.p, c("Nr1i3", "chr1_42.65"), list(1,"1"))
marg <- Marginals(tree.post, c("HDL", "Ppap2a"))
marg$ marginals$HDL
head(marg$ marginals$Ppap2a)
```

---

**PlotCGBN**

*Plot the Bayesian network*

**Description**
Plot and compare two Bayesian networks with different evidence(s) absorbed and propagated.

**Usage**

```r
PlotCGBN(
    tree.1,
    tree.2,
    fontsize = NULL,
    pbar = FALSE,
    plotting = TRUE,
    epsilon = 10^-6
)
```
Arguments

- `tree.1` (a `ClusterTree`)
- `tree.2` (a `ClusterTree`)
- `fontsize` (font size for the node labels)
- `pbar` (logical(1) whether to show progress bar)
- `plotting` (logical(1) whether to output plot)
- `epsilon` (numeric(1) the KL divergence is undefined if certain states of a discrete variable have probabilities of 0. In this case, a small positive number epsilon is assigned as their probabilities for calculating the divergence. The probabilities of other states are shrunk proportionally to ensure they sum up to 1.)

Details

Network visualization of the node-specific differences between Bayesian Networks with the same topology, but evidence that has been absorbed and propagated. The change of marginal distribution of each node is measured by signed and symmetric Kullback-Leibler divergence. The sign indicates the direction of change, with `tree.1` considered as the baseline. The magnitude of the change is reflected by the value. Nodes that are white are d-separated from the evidence. This function requires `Rgraphviz` package.

Value

- a plot of Bayesian network
- a vector of signed symmetric Kullback-Leibler divergence

Author(s)

Han Yu

References


Examples

```r
## Not run:
library("Rgraphviz")
data(toytree)
tree.post <- AbsorbEvidence(toytree, c("Nr1i3"), list(1))
PlotCGBN(tree.1=toytree, tree.2=tree.post)
## End(Not run)
```
PlotMarginals

Plot the marginal distributions.

Usage

PlotMarginals(marginals, groups = NULL)

Arguments

marginals the marginal distributions returned by Marginals for plotting
groups names of the marginals to be shown on plots

Details

Plot the marginal distributions. Marginals of discrete variables are plotted as bar plots, while those of continuous variables as density plots.

Author(s)

Han Yu

References


See Also

Marginals

Examples

data(toytree)
marg <- Marginals(toytree, c("Neu1", "Nr1i3", "chr1_42.65", "Spgl1"))
PlotMarginals(marginals=marg, groups=NULL)
**PlotTree**

---

*Plot the cluster tree*

---

**Description**

Plot the structure of a `ClusterTree` object

**Usage**

```
PlotTree(tree, color = "gray90")
```

**Arguments**

- `tree` a `ClusterTree` object
- `color` nodes color

**Details**

Plot the structure of a `ClusterTree` object, with the nodes labeled by corresponding elimination node. The circles represent continuous clusters, while the boxes represent discrete clusters. This function requires `Rgraphviz` package.

**Author(s)**

Han Yu

**References**


**Examples**

```r
## Not run:
library("Rgraphviz")
data(toytree)
PlotTree(toytree)

## End(Not run)
```
Propagate

Propagate the cluster tree

Description

This function propagates the discrete compartment of a ClusterTree object.

Usage

Propagate(tree, targets = NA)

Arguments

- tree: an initialized ClusterTree object
- targets: the cluster involved in evidence propagation, usually set by default

Details

The discrete compartment must be propagated to get the joint distributions of discrete variables in each discrete clusters. A ClusterTree object must be propagated before absorbing evidence and making queries.

Value

a ClusterTree object

References


Examples

data(liver)
tree.init <- Initializer(dag=liver$dag, data=liver$data,
                      node.class=liver$node.class,
                      propagate = FALSE)

propagate = FALSE)
tree.init <- Propagate(tree.init)
tree.init@propagated
qtlnet_to_graphNEL

Convert qtlnet to graphNEL object

Description

Extract network structure from qtlnet object and convert to graphNEL object

Usage

qtlnet_to_graphNEL(data)

Arguments

data a qtlnet object

Details

This function extracts network structure from qtlnet object and convert to graphNEL object. The example data can be downloaded from <https://github.com/hyu-ub/BayesNetBP>.

Value

graphNEL a graphNEL object.

Author(s)

Han Yu

Examples

## Not run:
load(liverqtl.rda)
qtlnet_to_graphNEL(liverqtl$qtlnet.fit)

## End(Not run)
runBayesNetApp  

Launch the BayesNetBP Shiny App

Description
Launch the BayesNetBP Shiny App

Usage
runBayesNetApp(launch.browser = TRUE)

Arguments
launch.browser  logical(1) whether launch the App in browser

Details
The function runBayesNetApp launches the Shiny App accompanied with this package. The app loads the toytree example by default and allows users to load customized ClusterTree object. In order to use this feature, a ClusterTree object should be built, propagated and named tree.init.p, and then saved as a .RDATA file. This file can be read in by the app.

The console of BayesNetBP Shiny App comprises three panels. The first part controls the model loading, visualization and subnetwork selection. The Fit function fits the entire graph in the window. The Fit Selected function fits the selected subnetwork to the window. The user can subset the network for visualization. The Expand function can trace the one hop neighbor of selected nodes in a stepwise manner. After selecting desired node sets, the user can subset the graph by the Subset function.

The second panel is used for absorption of fixed and hard evidences. The users can add multiple pieces of evidence to a list and absorb them into the model simultaneously. Marginals of other nodes can be queried as density or bar plots by node types. If a set of evidence has been absorbed, the marginals both before and after absorption will be returned to facilitate comparison. To query the marginals, the user can select the node of interest in the graph, and then click Marginal of Selected. The Shift in Marginals function computes the signed and symmetric Kullback-Liebler divergence for all applicable nodes in the network, and colors the nodes by their divergence and change in directions.

The function for systematic assessment of variable marginal shifts is provided in the third panel. It allows user to specify which node to absorb the spectrum of evidence in the select menu and click Select Observed, and to select whose divergence to be calculated by selecting the node in the menu and then clicking Add to Plot. Alternatively, the user can use Add All function to select all applicable nodes into the plotting list. The result is visualized in an interactive plot. The Min, Max and Step controls the range of values of the evidence to be absorbed.
Sampler

Author(s)
Han Yu

References

Examples

```r
## Not run:
# load or install required packages to run App
library("shiny")
library("googleVis")
library("devtools")
devtools::install_github("cytoscape/cyjShiny")
library("cyjShiny")
# run the App in browser
runBayesNetApp(launch.browser=TRUE)

## End(Not run)
```
SummaryMarginals

References


Examples

data(toytree)
Sampler(tree = toytree, n = 10)

SummaryMarginals (Summary a continuous marginal distribution)

Description

This function summary the marginal distributions of continuous variables by outputing the mean, standard deviation, and number of subpopulations

Usage

SummaryMarginals(marginals)

Arguments

marginals the marginal distributions obtained from Marginals function

Value

a data.frame object containing information about the marginal distributions for continuous variables. The marginal distributions of continuous variables in a CG-BN model are mixtures of Gaussian distributions. Therefore, besides the mean and standard deviation, the object has an additional column to specify the number of Gaussian mixtures.

mean the mean value of a Gaussian distribution.

sd the standard deviation of a Gaussian distribution.

n the number of Gaussian distributions in the mixture.

References


toytree

See Also

Marginals

Examples

data(liver)
tree.init.p <- Initializer(dag=liver$dag, data=liver$data,
node.class=liver$node.class,
propagate = TRUE)
marg <- Marginals(tree.init.p, c("HDL", "Ppap2a", "Neu1"))
SummaryMarginals(marginals=marg)

toytree

A ClusterTree Example of Liver Model

Description

A propagated ClusterTree object named toytree, obtained from liver QTL data.

Usage

data(toytree)

Format

The data set contains a propagated ClusterTree object toytree, which is ready for evidence absorption and making queries.

yeast

Saccharomyces Cerevisiae eQTL data from Kruglak et. al. (2005)

Description

eQTL data from 112 F1 segregants from a cross between BY4716 and RM11-1a strains of Saccharomyces Cerevisiae.

Usage

data(yeast)

Format

The data set yeast is a data frame of 112 observations of 50 variables: genotype data (genotype states at 12 SNP markers) and phenotype data (normalized and discretized expression values of 38 genes). Both genotypes and phenotypes are of class factor.
Details

The yeast dataset is a subset of the widely studied yeast expression dataset comprising of 112 F1 segregants from a cross between BY4716 and RM11-1a strains of Saccharomyces Cerevisiae. The original dataset consists of expression values reported as log2(sample/ BY reference) for 6216 genes. The data can be accessed in Gene Expression Omnibus (GEO) by accession number (GSE1990). After linkage analysis and filtering based on location and significance of QTL, a final set of 38 genes and their corresponding 12 SNP markers were identified and included in the yeast dataset. The gene expression values are discretized around the median and have two states, 1 (above or equal to median) and -1 (below median). There are two genotype states: 1 or 2. Thus the final dataset is a data frame of 112 observations (genotype) of 12 variables (SNP markers) and normalized gene expression of 38 variables (genes).

References


Index

AbsorbEvidence, 2
bn_to_graphNEL, 3
chest, 4
ClusterTree, 2, 3, 7–10, 12, 13, 15, 17, 19, 20, 22, 23, 25
ClusterTree-class, 4
ClusterTreeCompile, 5, 8, 9, 12, 13
ComputeKLDs, 6
ElimTreeInitialize, 6, 8, 12, 13, 15
emission, 9
emission1000, 10
FactorQuery, 10
GetValue, 11
Initializer, 12
liver, 13, 25
LocalModelCompile, 8, 9, 12, 13, 14
Marginals, 15, 18, 24, 25
PlotCGBN, 16
PlotMarginals, 16, 18
PlotTree, 19
Propagate, 12, 13, 20
qtlnet_to_graphNEL, 21
runBayesNetApp, 22
Sampler, 23
SummaryMarginals, 16, 24
toytree, 25
yeast, 25