Package 'geneNetBP'

May 6, 2016

Type Package

Title Belief Propagation in Genotype-Phenotype Networks

Version 2.0.0

Date 2016-05-05

Author Janhavi Moharil

Maintainer Janhavi Moharil <janhavim@buffalo.edu>

Depends R (>= 3.1.0),Rgraphviz (>= 2.8.1), graph (>= 1.42.0), bnlearn (>= 3.7.1), gRain (>= 1.2-3)

Suggests RHugin

Imports igraph, scales

Additional_repositories http://rhugin.r-forge.r-project.org/

Description Belief propagation methods in genotype-phenotype networks (Conditional Gaussian and Discrete Bayesian Networks) to propagate phenotypic evidence through the network.

License GPL-2

R topics documented:

geneNetBP-package *Belief Propagation in Genotype-Phenotype Networks*

Description

The geneNetBP package implements algorithms to infer Conditional Gaussian Bayesian Networks (CG-BN) from genotype-phenotype or Quantitative Trait Loci (QTL) data, absorb evidence in the network and quantify and visualize the changes in network beliefs. Inference of Discrete Bayesian networks is also supported.

Details

Package:	geneNetBP
Type:	Package
Version:	1.0.0
Date:	2016-04-22
License:	GPL-2

This package implements structure learning and conditional probability learning methods from the packages `RHugin`, **gRain** and **bnlearn** to Quantative Trait Loci (QTL) data and provides functions to query and visualize the network beliefs. The function `fit.gnbp` can be used to learn conditional gaussian bayesian networks from genotype-phenotype data while and `fit.dbn` allows infering discrete bayesian networks from categorical data. The main function for absorbing evidence and quantifying the changes in beliefs is `absorb.gnbp` for CG-BN and `absorb.dbn` for discrete networks. A generic `plot` method is available for visualizing network beliefs.

For belief propagation in CG-BN, the package makes extensive use of `RHugin` package that provides an R interface for the Hugin Decision Engine (commercial software for belief propagation). geneNetBP is compatible with demo versino of HuginLite. Note that the demo version of Hugin is restricted to 50 states and 500 cases.The package RHugin is not available on CRAN and can be obtained from R-Forge by accessing the link given below. Hugin Decision Engine should also be installed. Please visit the project homepage for installation instructions.

For discrete bayesian networks, algorithms from the package **bnlearn** are implemented for structural learning while belief propagation is implemented using **gRain**.

All the functions in the geneNetBP package are documented. For a complete list of functions, use the command ls(pos = "package:geneNetBP"). A vignette illustrating the use of geneNetBP is available.

Author(s)

Janhavi Moharil <janhavim@buffalo.edu>

Maintainer: Janhavi Moharil <janhavim@buffalo.edu>

References

Janhavi Moharil, Paul May, Daniel P. Gaile, Rachael Hageman Blair (2016). "Belief Propagation in Genotype-Phenotype Networks", *Stat Appl Genet Mol Biol*, 15(1):39-53. pdf

The geneNetBP Homepage: http://genenetbp.r-forge.r-project.org/

The RHugin Homepage: http://rhugin.r-forge.r-project.org

HUGIN EXPERT website: http://www.hugin.com

HUGIN API Reference Manual

---

absorb-methods *Absorb evidence and propagate beliefs in a genotype-phenotype network*

Description

Absorb a single piece or a spectrum of evidence for one or more phenotype nodes, propagate the beliefs, obtain the updated beliefs and quantify the effects.

Usage

```
## For Conditional Gaussian Bayesian Networks
## and Discrete Bayesian Networks (Implements RHugin)
absorb.gnbp(object, node, evidence)
## For Discrete Bayesian Networks (Implements gRain)
absorb.dbn(object, node, evidence)
```

Arguments

object	an object of class "gpfit" (output from fit.gnbp) for absorb.gnbp or an object of class "dbnfit" (Output from fit.dbn) for absorb.dbn.
node	a character vector specifying the names of the nodes for which the evidence is to be absorbed.
evidence	a matrix or a numeric vector of evidence. number of rows of the matrix or the length of the vector should be equal to the length of node.

Details

The function absorb.gnbp is compatible with the output produced by fit.gnbp. It absorbs evidence in both conditional gaussian bayesian networks or discrete bayesian networks inferred by RHugin and propagates beliefs by the PC algorithm implemented in the RHugin package. Jeffrey's Signed information is calculated to quantify the effects of the evidence absorption on the marginals. Note that the demo version of HuginLite is restricted to 50 states and 500 cases.

The function absorb.dbn is compatible with the output produced by fit.dbn. It absorbs evidence in only discrete bayesian networks that are inferred by bnlearn. Belief propagation is implemented using gRain package.

Value

absorb.gnbp returns an object of class "gnbp" while absorb.dbn returns an object of class "dbn". An object of class "gnbp" or "dbn" is a list containing the following components.

gp	an RHugin domain (for absorb.gnbp) or a grain object (for absorb.dbn) that is triangulated, compiled and with the latest absorbed evidence propagated.
gp_flag	type of network.
node	a character vector specifying the nodes for which evidence has been absorbed

marginal	a list of marginal probabilities for phenotypes (pheno) and genotypes (geno)
belief	a list of updated beliefs for phenotypes (pheno) and genotypes (geno)
JSI	a matrix of Jeffrey's signed information if network is Conditional Gaussian, otherwise NULL if network is Discrete Bayesian.
FC	a list of two. a matrix FC of fold changes and a matrix pheno_state of phenotype node beliefs - state with maxium probability. If network is Conditional Gaussian, a NULL value is returned.

The marginals, beliefs and JSI or FC are calculated for only *d*-connected nodes.

If a sequence of evidence is absorbed for a single node in a Conditional Gaussian network, a plot of JSI vs evidence is produced.

Author(s)

Janhavi Moharil <janhavim@buffalo.edu>

See Also

gen.evidence, plot.gnbp, plot.dbn

Examples

```
## load the mouse kidney eQTL dataset
data(mouse)

## get genotype and phenotype data
mousegeno<-mouse[,1:5]
mousepheno<-mouse[,6:19]

## Not run:
## Simple example : Fit a bayesian network to genotype-phenotype data
mouse.cgbn<-fit.gnbp(mousegeno,mousepheno,alpha=0.1)

## Absorb a single evidence for a single node
absorb.gnbp(mouse.cgbn,node="Tlr12",evidence=matrix(2.5))

## Absorb a sequence of evidence for a single node generated using \code{\link{gen.evidence}
mouse.cgbn<-fit.gnbp(mousegeno,mousepheno,alpha=0.1)
evidence<-gen.evidence(mouse.cgbn,node="Tlr12")
absorb.gnbp(mouse.cgbn,node="Tlr12",evidence=evidence)

##Absorb sequence of evidence for multiple nodes}
mouse.cgbn<-fit.gnbp(mousegeno,mousepheno,alpha=0.1)
evidence<-gen.evidence(mouse.cgbn,node=c("Ak2","Ptp4a2","Hmgcl"),std=2,std.equal=TRUE)
absorb.gnbp(mouse.cgbn,node=rownames(evidence),evidence=evidence)

## End(Not run)
```

`fit-methods` *Fit a Conditional Gaussian Bayesian Network or Discrete Bayesian Network to QTL data*

Description

Learn the structure of a genotype-phenotype network from quantitative trait loci (QTL) data and the conditional probability table for each node in the network.

Usage

```
## Fit a conditional gaussian or a discrete bayesian network using RHugin.
fit.gnbp(geno,pheno,constraints,learn="TRUE",graph,type ="cg",
                  alpha=0.001,tol=1e-04,maxit=0)
## Fit a discrete bayesian network using bnlearn.
fit.dbn(geno,pheno,graph,learn="TRUE",method="hc",whitelist,blacklist)
```

Arguments

`geno`	a data frame of column vectors of class factor (or one that can be coerced to that class) and non-empty column names.
`pheno`	a data frame of column vectors of class numeric for `fit.gnpb` if `type = "cg"` or class factor if `type = "db"` and for `fit.dbn`. Non-empty column names.
`constraints`	an optional list of constraints on the edges for specifying required and forbidden edges for `fit.dbn`. See details.
`learn`	a boolean value. If TRUE (default), the network structure will be learnt. If FALSE, only conditional probabilities will be learnt (a graph must be provided in this case.)
`graph`	graph structure of class "graphNEL" or a data frame with two columns of (labeled "from" and "to"), containing a set of edges to be included in the graph to be provided if learn == FALSE. See details.
`type`	specify the type of network for `fit.gnbp`. `"cg"` for `Conditional Gaussian` (default) and `"db"` for `Discrete Bayesian`.
`method`	a character string. The score-based or constraint-based algorithms available in the package **bnlearn**. Valid options are `"hc"`, `"tabu"`, `"gs"`, `"iamb"`, `"fast.iamb"`, `"inter.iamb"`, `"mmhc"`. See details below.
`whitelist`	a data frame with two columns of (labeled "from" and "to"), containing a set of edges to be included in the graph.
`blacklist`	a data frame with two columns (labeled "from" and "to"), containing a set of edges NOT to be included in the graph.
`alpha`	a single numeric value specifying the significance level (for use with RHugin). Default is 0.001.
`tol`	a positive numeric value (optional) specifying the tolerance for EM algorithm to learn conditional probability tables (for use with RHugin). Default value is 1e-04. See `learn.cpt` for details.
`maxit`	a positive integer value (optional) specifying the maximum number of iterations of EM algorithm to learn conditional probability tables (for use with RHugin). See `learn.cpt` for details.

Details

The function `fit.gnbp` fits a conditional gaussian bayesian network or a discrete bayesian network at the specified level of significance alpha, to genotype-phenotype (QTL) data by the PC algorithm implemented in the RHugin package. The conditional probability tables are learnt for each node in the domain by the EM algorithm implemented in the RHugin package.

Edges between the genotypes at SNP markers are not allowed and the genotypes are constrained to precede the phenotypes. The phenotypes should be either all numeric or all discrete. The function does not currently support mixture of discrete and continuous phenotypes. Additional domain knowledge in terms of edges should be provided as a list of constraints, the structure of which is described in detail in `learn.structure`. Briefly, the constraints argument is a list of two elements: directed and undirected. Each of these elements in turn should be a list with two elements: required and forbidden. The elements of required and forbidden must be a character vector of length two specifying the names of the nodes. See `learn.cpt` for details.

Note that this function works on Hugin domains. Since Hugin domains are external pointers and cannot be saved in R workspace, the RHugin package provides functions `read.rhd` and `write.rhd` for loading and saving the Hugin domains. See RHugin documentation for more information.

The function fit.dbn infers a discrete bayesian network structure from genotype-phenotype (QTL) categorical data by implementing score based and constraint based algorithms from the **bnlearn** package. The conditional probability tables are learnt for each node in the inferred network. The phenotypes should be ALL discrete variables. Additional domain knowledge in terms of edges should be provided as a `whitelist` and `blacklist`.Edges between the genotypes at SNP markers are not allowed and the genotypes are constrained to precede the phenotypes.

The supported algorithms from **bnlearn** are

1. Score-based: *Hill-Climbing* (`hc`,default), *Tabu Search* (`tabu`)
2. Constraint-based: *Grow-Shrink* (`gs`), *Incremental Association* (`iamb`), *Fast Incremental Association* (`fast.iamb`), *Interleaved Incremental Association* (`inter.iamb`)
3. Hybrid: *Max-Min Hill-Climbing* (`mmhc`).

The algorithm can be specified by `method`. Structure learning functions are implemented with their default parameters. If different parameter values are desired, it is recommended to learn the network structure independently using the bnlearn package.The inferred structure can be input as a graph object to `fit.dbn` and then set learn="FALSE".

Value

`fit.gnbp` returns an object of class "gpfit" containing the following components.

`gp`	a pointer to a compiled RHugin domain that is the inferred network structure and the conditional probability tables for each node in the network.
`marginal`	a list of marginal probabilities for phenotypes (`pheno`) and genotypes (`geno`)
`gp_nodes`	a data frame containing information about nodes for internal use with other functions.
`gp_flag`	a character string specifying the type of network : "cg" for `Conditional Gaussian` or "db" for `Discrete Bayesian`.

`fit.dbn` returns an object of class "dbnfit" containing the following components.

`dbn`	an object of class bn. See `bn-class` for details. This object contains the inferred network structure and the conditional probability tables for each node in the network.

marginal	a list of marginal probabilities for phenotypes (pheno) and genotypes (geno)
dbn_nodes	a data frame containing information about nodes for internal use with other functions.
dbn_flag	a character string specifying the type of network "dbn" for Discrete Bayesian.

Author(s)

Janhavi Moharil <janhavim@buffalo.edu>

See Also

plot.gpfit, plot.dbnfit, absorb.gnbp, For discrete bayesian networks : fit.dbn,absorb.dbn

Examples

```
## Not run:
## load the mouse kidney eQTL dataset
data(mouse)

## get genotype and phenotype data
mousegeno<-mouse[,1:5]
mousepheno<-mouse[,6:19]

## Simple example : Fit a bayesian network to genotype-phenotype data using the default values
fit.gnbp(mousegeno,mousepheno)

## Fit a bayesian network to genotype-phenotype data at a specified significance level
fit.gnbp(mousegeno,mousepheno,alpha = 0.1)

## load yeast dataset
data(yeast)

## get genotype and phenotype data
yeastgeno<-yeast[,1:12]
yeastpheno<-yeast[,13:50]

## Simple example : Fit a discrete bayesian network to genotype-phenotype data
fit.dbn(yeastgeno,yeastpheno)

## Fit a discrete bayesian network by Grow-Shrink method and plot it.
yeast.dbn.gs<-fit.dbn(yeastgeno,yeastpheno,method="gs")
plot(hdlgs)

## End(Not run)
```

---

gen.evidence *Generate a sequence of evidence for a continuous node in a conditional gaussian bayesian network.*

Description

The evidence for a node in an RHugin domain is generated as a linear sequence within the specified standard deviation from the marginal mean of the node. The evidence can be given as an input to absorb.gnbp

Usage

```
gen.evidence(gpfit, node, std = 2, length.out = 10, std.equal = TRUE)
```

Arguments

gpfit	an object of class "gpfit" obtained by using fit.gnbp
node	a character string specifying the name of a continuous node in the domain
std	a numeric value specifying the number of standard deviations of marginal distribution within which the evidence is generated. A numeric vector of length = number of nodes, must be specified when std.equal=FALSE.
length.out	a positive integer giving the desired length of the sequence.
std.equal	a logical value indicating whether same number of standard deviations should be used to generate evidence for all nodes. Default is TRUE.

Details

The evidence for a node in an RHugin domain is generated as a linear sequence within the specified standard deviation from the marginal mean of the node. The evidence can be given as an input to absorb.gnbp

Value

A matrix of evidence for each specified node

Author(s)

Janhavi Moharil<janhavim@buffalo.edu>

See Also

absorb.gnbp,fit.gnbp

Examples

```
##Fit a network
data(mouse)
## Not run:
mouse.cgbn<-fit.gnbp(mouse[,1:5],mouse[,6:14],alpha=0.1)

##Generate a sequence of evidence for a single node
evidence<-gen.evidence(mouse.cgbn,node="Tlr12",std=2,length.out=20)

## End(Not run)
```

hdl *Mus Musculus HDL QTL data from Leduc et. al. (2012)*

Description

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

Usage

```
data(hdl)
```

Format

The data set `hdl` is a data frame of 280 observations of 15 variables: genotype data (genotype states at 5 SNP markers) and phenotype data (HDL levels and normalized expression values of 10 genes).Genotypes are of class `factor` and phenotypes are of class `numeric`.

Details

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

Source

Leduc MS, Blair RH, Verdugo RA, Tsaih SW, Walsh K, Churchill GA, Paigen B.(2012). "Using bioinformatics and systems genetics to dissect HDL-cholesterol genetics in an MRL/MpJ x SM/J intercross." *J Lipid Res.*, 6, 1163-75.

mouse *Mus Musculus Kidney eQTL data from Hageman et. al. (2011)*

Description

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

Usage

```
data(mouse)
```

Format

The data set `mouse` is a data frame consisting of 173 observations of 19 variables: genotype data (genotype states at 5 SNP markers) and phenotype data (normalized expression data of 14 genes). Genotype data is of class `factor` and phenotype data is of class `numeric`.

Details

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

Source

Hageman,R.S., Leduc,M.S., Caputo,C.R., Tsaih,S.-W., Churchill, G.A., and Korstanje,R.(2011). "Uncovering genes and regulatory pathways related to urinary albumin excretion." *Journal of the American Society of Nephrology22*, 1, 73-81.

Examples

```
## Not run:
# load the data and fit a genotype-phenotype network
data(mouse)
fit.gnbp(mouse[,1:5],mouse[,6:19])

## End(Not run)
```

`plot methods` *Plot a Genotype-Phenotype Network*

Description

Plot methods for genotype-phenotype networks. `plot.gnbp` and `plot.dbn` can be used to plot networks in which evidence has been absorbed and propagated. The beliefs or Jeffrey's signed information for conditional gaussian and phenotypic states and Fold Changes for discrete bayesian networks is mapped onto the network. `plot.gpfit` and `plot.dbfit` are plot methods for objects of class "gpfit" and "dbnfit" respectively.

Usage

```
## S3 method for class 'gnbp'
plot(x, y="JSI",ncol = 1, col.palette,col.length = 100,
                    fontsize=14, fontcolor="black",...)
## S3 method for class 'dbn'
plot(x, y="state",ncol = 1, col.palette,col.length = 100,
                    fontsize=14, fontcolor="black",...)
## S3 method for class 'gpfit'
plot(x, fontsize=14, fontcolor="black",...)
## S3 method for class 'dbnfit'
plot(x, fontsize=14, fontcolor="black",...)
```

Arguments

`x`	An object of class "gnbp" (`plot.gnbp`), "dbn" (`plot.dbn`), "gpfit" (`plot.gpfit`) or "dbnfit" (`plot.dbnfit`)
`y`	A character string. Valid options are `"JSI"` (default) or `"belief"` for Conditional Gaussian network and `"FC"` or `"state"` for Discrete Bayesian networks. `plot.dbn` plots only discrete bayesian networks. Note that `y` will be ignored for `plot.dbnfit` and `gpfit`
`ncol`	a positive integer specifying the column number of JSI / belief / FC to plot. By default, the first column will be used.

`col.palette` A list of character strings. For `"JSI"`,`"belief"` and `"FC"` a list of 6 elements specifying colors for colormap.All 6 elements should be character strings specifying the colour for `pos_high`= high end of gradient of positive values (default = "red" (JSI, belief), "darkmagenta",(FC)) `pos_low`=low end of gradient of positive values (default = "wheat1" (JSI, belief), "palegoldenrod",(FC)) `neg_high`=high end of gradient of positive values (default = "cyan" (JSI, belief), "palegoldenrod",(FC)) `neg_low`=low end of gradient of positive values (default = "blue" (JSI, belief), "gold2",(FC)) `dsep_col`= *d*-separated nodes (default = "white") `qtl_col`= discrete nodes (QTLs) (default = "grey") `node_abs_col`= nodes for which evidence has been absorbed (default = "palegreen2")

For `"state"`, a list of 4 elements specifying colors for colormap should be specified. All 4 elements should be character strings specifying the colour for `col_nodes`- a vector of colors for phenotype states should be specified. The length of the vector should be equal to the maximum number of phenotype states possible. `dsep_col`= *d*-separated nodes (default = "white") `qtl_col`= discrete nodes (QTLs) (default = "grey") `node_abs_col`= nodes for which evidence has been absorbed (default = "palegreen2")

`col.length` a positive integer (default = 100) specifying the resolution of the colormap (number of colors) in case of `"belief"`, `"JSI"` and `"FC"`. For `"state"`, this argument will be ignored.

`fontsize` a single numeric value. fontsize for node labels

`fontcolor` fontcolor for node labels

`...` further arguments to the function `plot`. These will be ignored

Details

`plot.gpfit` and `plot.dbnfit` are generic plot methods for objects of class"gpfit" and "dbnfit" respectively that are output from the fit-methods. These are networks in which evidence has not been absorbed and propagated.

`plot.gnbp` and `plot.dbn` are generic plot methods for objects of class "gnbp" and "dbn" that are outputs from the absorb methods. These functions plot the genotype-phenotype networks in which evidence has been absorbed and propagated and maps the quantitative system wide effects on the network. Both conditional gaussian and discrete bayesian networks are supported. Users can specify the colormap options such as end colors for the positive and negative gradients and the resolution of the colormap. The default node shapes are "ellipse" for the phenotype nodes and "box" for genotype nodes.The *d*-separated nodes are white while the colored nodes are *d*-connected, with the color indicating the strength and direction of change. The node for which evidence is absorbed is colored green (default color).

Value

x is invisibly returned

Author(s)

Janhavi Moharil<janhavim@buffalo.edu>

See Also

`absorb.gnbp`, `absorb.dbn`, `fit.gnbp`, `fit.dbn`

Examples

```
## Fit, absorb and plot a genotype-phenotype network
data(mouse)
## Not run:
mouse.cgbn<-fit.gnbp(mouse[,1:5],mouse[,6:19],alpha=0.1)
plot(mouse.cgbn)
mouse.cgbn.abs<-absorb.gnbp(network,node="Tlr12",evidence=matrix(-0.99))
plot(mouse.cgbn.abs)

## End(Not run)
```

---

toy *Toy example dataset*

Description

Toy example, Simulated data set

Usage

```
data(toy)
```

Format

The data set `toy` is a data frame of 500 observations of 9 variables: genotype data (genotype states at 3 SNP markers) and phenotype data (6 variables).Genotypes are of class `factor` and phenotypes are of class `numeric`.

Details

The simulated dataset consists of 3 genotypes each with 2 states and 6 phenotypes.

Examples

```
## Not run:
# load the data and fit a genotype-phenotype network
data(toy)
fit.gnbp(toy[,1:3],toy[,4:9])
## End(Not run)
```

---

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).\

Source

Brem RB, Kruglyak L. The landscape of genetic complexity across 5,700 gene expression traits in yeast. Proc Natl Acad Sci U S A 2005 Feb 1;102(5):1572-7.\

Brem RB, Storey JD, Whittle J, Kruglyak L. Genetic interactions between polymorphisms that affect gene expression in yeast. Nature 2005 Aug 4;436(7051):701-3.\

Examples

```
## Not run:
# load the data and fit a genotype-phenotype network
data(yeast)
fit.dbn(yeast[,1:12],yeast[,13:50])

## End(Not run)
```

Index