

# Inference in Bayesian Networks with R package **BayesNetBP**

*Han Yu, Rachael Hageman Blair*

*2017-06-30*

## Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Conditional Gaussian Bayesian network example</b>	<b>2</b>
2.1	Model initialization . . . . .	2
2.2	Evidence absorption and queries . . . . .	3
2.3	Visualization . . . . .	7
<b>3</b>	<b>Discrete Bayesian network example</b>	<b>10</b>
<b>4</b>	<b>Shiny app</b>	<b>14</b>
	<b>References</b>	<b>15</b>

## 1 Introduction

The **Bayesian Network Belief Propagation** (**BayesNetBP**) package was developed in the R programming language (<https://www.r-project.org/>), for probabilistic reasoning in Probabilistic Graphical Models (PGMs) known as Bayesian Networks (BNs). Implementation of the belief propagation is based on the work by Cowell (2005). If you would like to use **BayesNetBP** in your publications, please cite the package as

Han Yu, Moharil Janhavi, Rachael Hageman Blair. “BayesNetBP: An R package for probabilistic reasoning in Bayesian Networks”. *Submitted*.

Bayesian networks are a class of PGMs that convey directed dependencies between variables (nodes) in the network. These dependencies can be read from the graph as conditional probabilities and are factored terms in a compact representation of the joint probability distribution for the variables in the network Lauritzen (1996), Koller and Friedman (2009). For purely continuous or purely discrete BNs, these local model are often defined using Gaussian regressions and Conditional Probability Table (CPTs), respectively. Conditional Gaussian Bayesian Networks (CG-BNs) accommodate a mixture of discrete and continuous variables, where discrete nodes can be parents of continuous nodes, but not vice-versa.

Probabilistic reasoning enables a user to absorb information into a BN and make queries about how the probabilities within the network most likely change in light of new information. The inference in BNs exploits new evidence (information) about the states of a node (or nodes) to make probabilistic queries about other nodes in the network. A major strength of such probabilistic reasoning is that it can be performed with only partial information about the nodes in the network, but can generate comprehensive system-wide queries.

There are two packages developed in the R programming language that can be used for probabilistic reasoning, **RHugin** Konis (2017) and **gRain** Højsgaard (2012). **RHugin** can be used for network inference and belief propagation for CG-BNs, but relies on the commercial software Hugin Hugin Expert A/S (2017). While a free demo version of huginlite can be used in connection with **RHugin**, the reasoning and inference is limited to smaller networks and datasets (50 states and 500 cases) for the demo version. The **gRain** package can handle large datasets and networks, but it supports probabilistic reasoning only in purely discrete networks. On the other hand, **BayesNetBP**, not only supports probabilistic reasoning in purely discrete, purely continuous, and

CG-BNs, but also provides tools for quantification of distributional changes and visualization. Therefore, the **BayesNetBP** package fills a major gap in the graphical modeling tools available in R. The package is the first open source package to facilitate probabilistic reasoning and novel visualizations in all types of CG-BNs. In the following sections, we present examples that are motivated by problems in statistical genetics. However, we emphasize that the **BayesNetBP** can be used in connection with any application.

In general, the **BayesNetBP** package takes a DAG and a data set as input. It can accommodate DAGs learned from different packages in R such as **bnlearn** Scutari (2010), **RHugin** or a network that is described by an expert based on prior knowledge network structure. In the following sections, we will give a detailed introduction on how to use **BayesNetBP** to perform probabilistic reasoning in a CG-BN example. We will also demonstrate the application in purely discrete and continuous networks.

## 2 Conditional Gaussian Bayesian network example

The data set in this example is from the livers of MRL/MpJ  $\times$  SM/J mouse intercross, and consists of gene expression data, genotypes at SNP markers and High Density Lipoprotein (HDL) Leduc et al. (2012). Genes that share a QTL with HDL on chromosome 1 and also relate to enriched categories for lipid metabolism in KEGG and Gene Ontologies were selected Alvord et al. (2007). The filtered data used for the modeling can be found in the **BayesNetBP** package. Within this network, we also consider dichotomizing three of the nodes, which creates a second discrete layer in the CG-BN. This example demonstrates the functions for initialization, reasoning and visualizations in a CG-BN.

### 2.1 Model initialization

For initialization, a **graphNEL** object of DAG and a vector specifying node types are required to build the semi-elimination tree. In this example, the vector **node.class** indicates which nodes are discrete (**TRUE**) and continuous (**FALSE**). The **ClusterTreeCompile** function builds the graph of semi-elimination tree and get the cluster sets, which are the frame work of the final computational object. The **LocalModelCompile** function estimates the local models from a DAG and a data frame. The columns of the data frame must be named by corresponding node names. The local models computed by **LocalModelCompile** function are distributed into the semi-elimination tree through the **ElimTreeInitialize** function. After initialization, a **ClusterTree** object will be generated.

```
library("BayesNetBP")
data("liver")
liver$node.class
```

```
##      HDL      Pla2g4a      Nr1i3      Cyp2b10      Ppap2a      Kdsr
##      TRUE      FALSE      FALSE      TRUE      FALSE      FALSE
##      Degr1      Neu1      Spgl1      Apoa2      chr1_42.65      chr1_71.35
##      FALSE      FALSE      TRUE      FALSE      TRUE      TRUE
## chr1_84.93 chr1_86.65 chr12_30.87
##      TRUE      TRUE      TRUE
```

```
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)
```

The **ClusterTree** object is not ready for evidence absorption or making queries until its discrete compartment is propagated. The function **PropagateDBN** will perform the propagation within the discrete compartment of the clustertree object so that the joint distribution tables of all clusters are computed. Now the **ClusterTree** object is ready for evidence absorption and queries.

```
tree.init@propagated

## [1] FALSE
tree.init.p <- PropagateDBN(tree.init)
tree.init.p@propagated

## [1] TRUE
```

## 2.2 Evidence absorption and queries

A `GetValue` function can be used to check how the values for each discrete variable are coded. For example, the following code shows the locus `Chr1@42.65` has three states, while `Spg11` has two states *High* and *Low*. This information is helpful for evidence absorption.

```
GetValue(tree.init.p, "Nr1i3")

## The node is continuous.
## NULL
GetValue(tree.init.p, "chr1_42.65")

## [1] "1" "2" "3"
GetValue(tree.init.p, "Spg11")

## [1] "High" "Low"
```

The `AbsorbEvidence` function can handle evidence of a numeric value for continuous nodes. For a discrete node, the evidence can be either an observed state (hard evidence) or a likelihood (soft evidence). In the following example, there are three kinds of evidence: *Nr1i3* is observed with 1, `Chr1@42.65` is observed with state 1, and the likelihood of *Spg11* being *High* is 0.9, and being *Low* is 0.2. Those three pieces of information can enter the model simultaneously through the `AbsorbEvidence` function. The absorbed information in the model can also be accessed.

```
tree.post <- AbsorbEvidence(tree.init.p, c("Nr1i3", "chr1_42.65", "Spg11"),
  list(1, "1", c(High=0.9, Low=0.2)))
tree.post@absorbed.variables

## [1] "chr1_42.65" "Nr1i3"
tree.post@absorbed.values

## $chr1_42.65
## [1] "1"
##
## $Nr1i3
## [1] 1
tree.post@absorbed.soft.variables

## [1] "Spg11"
tree.post@absorbed.soft.values

## $Spg11
## High Low
## 0.9 0.2
```

The marginal distributions of both continuous and discrete variables before or after observing the above evidence can be queried with the function `Marginals`. The output of `Marginals` function returns a list of two elements. The first element `marginals` is a list of marginals, while the second element `types` is a vector indicating the type of nodes queried. For a continuous variable, the marginal is a mixture of Gaussian distributions, output as a data frame with three columns of sub-population probabilities, means and variances. For a discrete variable, the marginal is a named vector of probabilities. The function `SummaryMarginals` computes means and standard deviations for continuous variables as well as returns the number of sub-populations. In the following example, the marginals of both *Ppap2a* and *Neu1* after absorption are queried, and they comprise of 108 Gaussian sub-populations.

```
marg <- Marginals(tree.post, c("HDL", "Ppap2a", "Neu1", "chr1_71.35"))
marg$marginals$HDL
```

```
##           High           Low
## 0.2524564 0.7475436
```

```
SummaryMarginals(marg)
```

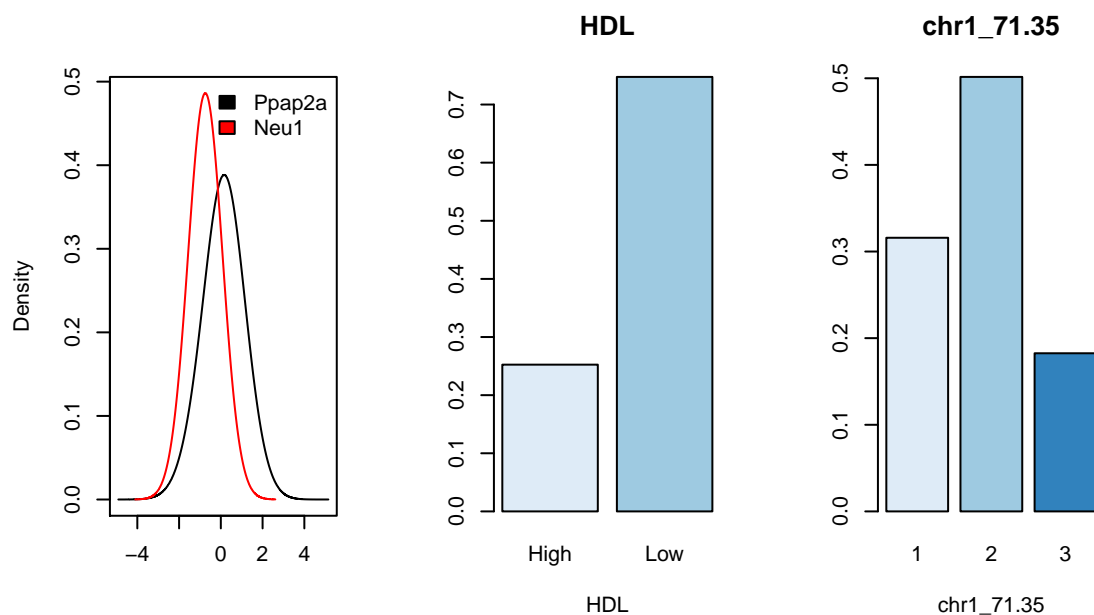
```
##           Mean          SD    n
## Ppap2a  0.1204055 1.0061905 108
## Neu1    -0.7446580 0.6710785 108
```

```
head(marg$marginals$Ppap2a)
```

```
##           prob          mu          sd
## 1 6.514238e-03  0.1466045 1.1614251
## 2 9.852202e-03 -0.6530988 0.8787452
## 3 9.704217e-04 -1.1213652 0.9937244
## 4 3.341952e-03  0.1476780 1.1888417
## 5 2.277200e-03 -0.6808841 0.8862256
## 6 2.432649e-05 -1.3578320 0.9972592
```

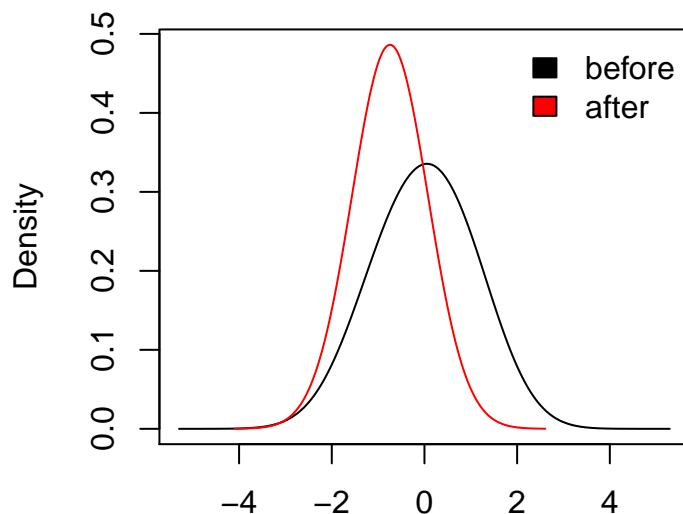
Marginal distributions of both discrete and continuous nodes can be visualized with the `PlotMarginals` function. The function outputs multiple marginal distributions simultaneously. Marginals of continuous nodes are shown as density plots, while those of discrete nodes as barplots.

```
PlotMarginals(marg)
```



The following codes compare the marginal distributions of Neu1 before and after absorbing the above evidence.

```
var <- "Neu1"
marg.2 <- Marginals(tree.post, var)
marg.1 <- Marginals(tree.init.p, var)
mgns <- list(marg.1$marginals[[1]], marg.2$marginals[[1]])
names(mgns) <- c(var, var)
types <- c(marg.1$types[1], marg.2$types[1])
marg <- list(marginals=mgns, types=types)
PlotMarginals(marg, groups=c("before", "after"))
```



The function `FactorQuery` can provide the joint distribution of any combination of factors, as well as conditional distributions of all discrete variables. In the following example, the joint distribution of *HDL* and *Cyp2b10*, and the conditional distribution of *HDL*, are computed. For a conditional distribution, the output is the distribution of the queried node conditioned on all its parents.

```
FactorQuery(tree.post, c("HDL", "Cyp2b10"), mode="joint")
```

```
##      HDL Cyp2b10      prob
## 1 High   High 0.18567653
## 2 High   Low  0.06677985
## 3 Low    High 0.52273666
## 4 Low    Low  0.22480697
```

```
FactorQuery(tree.post, c("HDL"), mode="conditional")
```

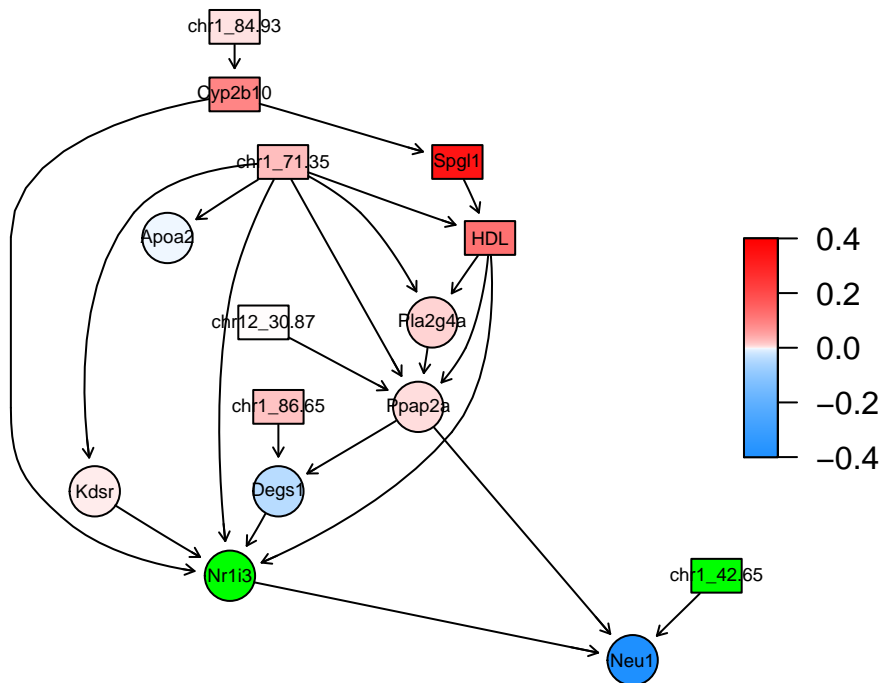
```
##      HDL Spgl1 chr1_71.35      prob
## 1 High   High          1 0.36725775
## 2 Low    High          1 0.63274225
## 3 High   High          2 0.20257483
## 4 Low    High          2 0.79742517
## 5 High   High          3 0.04563497
## 6 Low    High          3 0.95436503
## 7 High   Low           1 0.59305462
## 8 Low    Low           1 0.40694538
## 9 High   Low           2 0.42154566
## 10 Low   Low           2 0.57845434
## 11 High  Low           3 0.17304262
## 12 Low   Low           3 0.82695738
```

## 2.3 Visualization

The shift of marginals between two models can also be visualized using `PlotCGBN` function. This function takes input of two `ClusterTree` objects, computes the signed and symmetric KL divergence between marginals for each variable, and outputs a graph whose nodes are colored accordingly. The following figure shows an example in which *Nr1i3* and *Chr1@42.65* have absorbed hard evidence (green). The changes of the marginals for the other nodes are quantified using a signed and symmetric KL divergence. They are depicted on the network with a colorbar representaiton. Note that the discrete nodes will only change in one direction. For the continuous nodes, red indicates an increase in mean (activation), and blue a decrease in mean (inhibited). This function also returns the signed symmetric KL divergence for each node.

```
PlotCGBN(tree.init.p, tree.post, fontsize = 32)
```

##	HDL	Sppl1	Cyp2b10	chr12_30.87	chr1_86.65
##	0.1307547416	0.3534706656	0.0925031249	0.0001042556	0.0245952583
##	chr1_84.93	chr1_71.35	Neu1	Degs1	Ppap2a
##	0.0057804993	0.0288291851	-0.4275696912	-0.0397243702	0.0067383627
##	Apoa2	Kdsr	Pla2g4a		
##	-0.0015485380	0.0022800388	0.0124116400		



In the following example, the `PlotCGBN` function is used to demonstrate how conditional independencies within the network change based upon where evidence has been observed. The Bayesian network in the upper panel has an active trail between *Sppl1* and *Chr1@84.93*, so after observing *Sppl1*, the marginal of *chr1@84.93* will change (symmetric KL divergence is 0.0047). However, if *Cyp2b10* is also observed (gray in the lower panel), then *Sppl1* and *Chr1@84.93* become *d*-separated, and further absorption of evidence on *Sppl1* will not change the distribution of *Cypb10* (symmetric KL divergence is 0). The exploration of the conditional independencies described can be performed by simply absorbing *Cyp2b10* into the original model to get `tree.1`, and further absorb *Sppl1* into it to get `tree.2`, and finally compare `tree.1` and `tree.2` by `PlotCGBN`.

```

tree.1 <- AbsorbEvidence(tree.init.p, c("Cyp2b10"), list("High"))
tree.2 <- AbsorbEvidence(tree.1, c("Spg11"), list("High"))
tree.3 <- AbsorbEvidence(tree.init.p, c("Spg11"), list("High"))
PlotCGBN(tree.init.p, tree.3, fontsize = 32)

```

```

##          HDL      Cyp2b10  chr12_30.87   chr1_86.65   chr1_84.93
## 4.813077e-02 7.428658e-02 0.000000e+00 0.000000e+00 4.669436e-03
##   chr1_71.35   chr1_42.65           Neu1           Nr1i3           Degs1
## 1.337859e-17 4.054185e-17 9.954463e-04 1.021268e-02 1.563882e-04
##          Ppap2a          Apoa2          Kdsr          Pla2g4a
## 3.805036e-03 7.237347e-18 0.000000e+00 1.580037e-03

```

```

PlotCGBN(tree.1, tree.2, fontsize = 32)

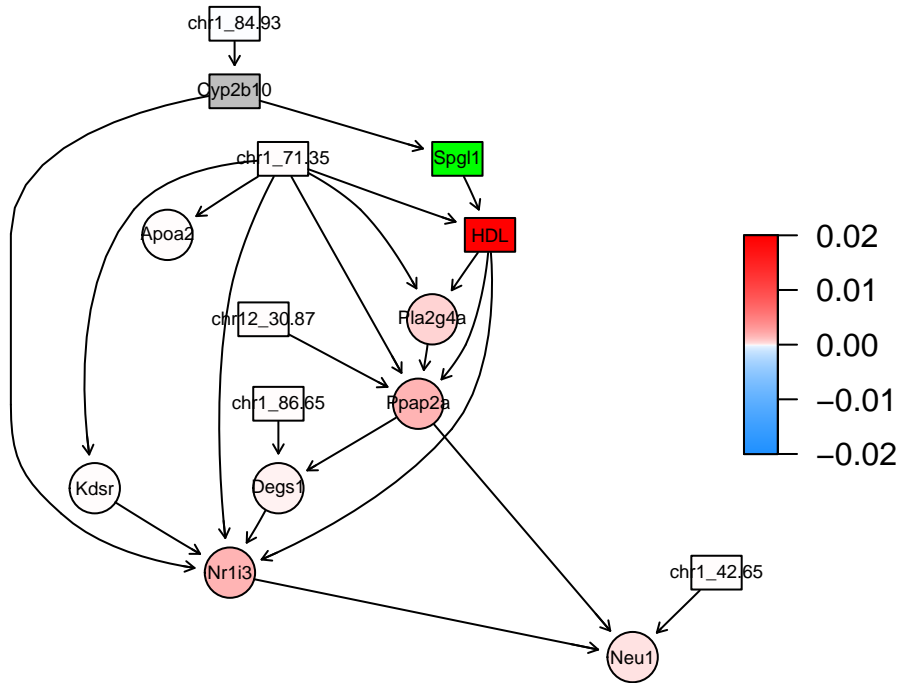
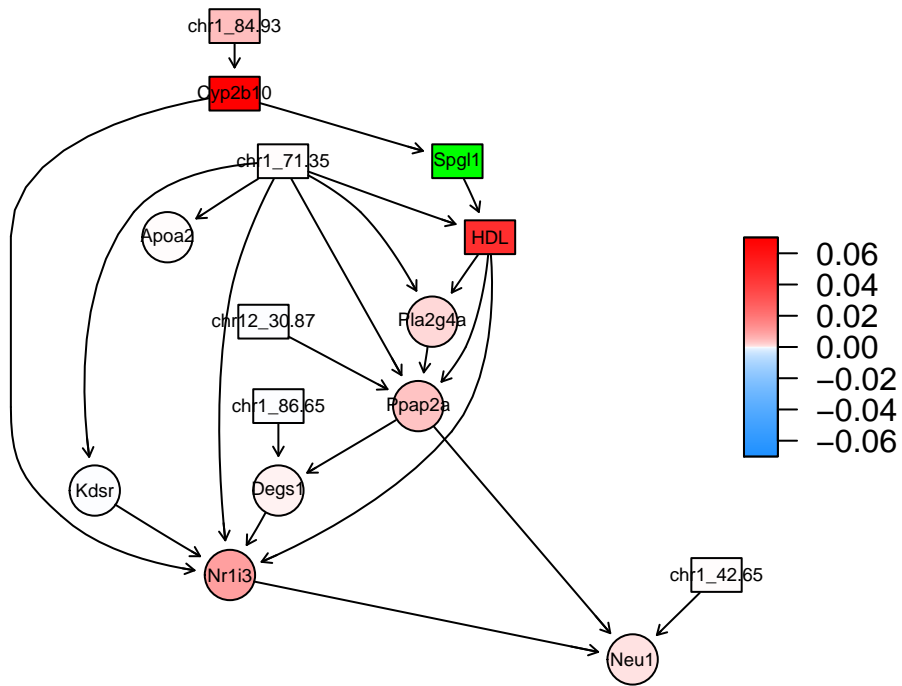
```

```

##          HDL  chr12_30.87   chr1_86.65   chr1_84.93   chr1_71.35
## 1.907479e-02 1.299197e-17 4.103860e-17 0.000000e+00 1.198082e-17
##   chr1_42.65           Neu1           Nr1i3           Degs1           Ppap2a
## 4.054185e-17 2.176000e-04 1.621936e-03 6.144299e-05 1.513669e-03
##          Apoa2          Kdsr          Pla2g4a
## 2.209711e-17 6.665141e-18 6.124072e-04

```



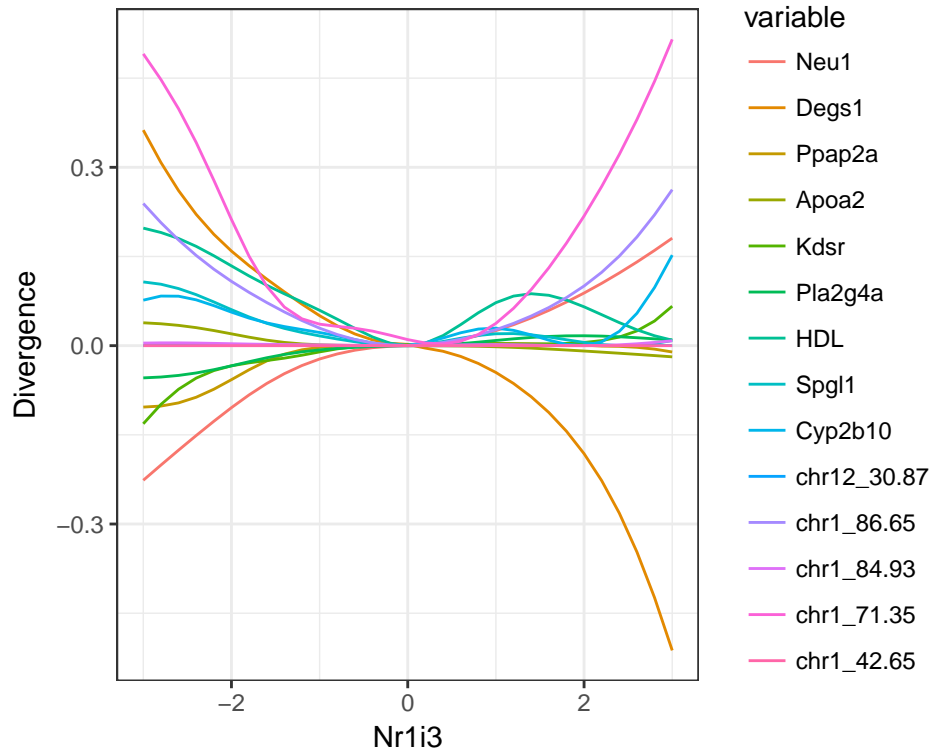


This package also facilitate a systematic assessment of variable marginal shifts over a spectrum of evidence through function `ComputeKLDs`. In the following example, evidences on *Nr1i3* ranging from -3 to 3 are absorbed into the original model, and the divergence of all other variables are computed and returned as a data frame. The output can be visualized by any graphical tools.

```

library("reshape2")
library("ggplot2")
klds <- ComputeKLDs(tree=tree.init.p, var0="Nr1i3",
  vars=setdiff(tree.init.p@node, "Nr1i3"),
  seq=seq(-3,3,0.2), pbar=FALSE)
klds.melt <- melt(klds, id="x")
ggplot(data=klds.melt, aes(x=x, y=value, group=variable, colour=variable)) +
  geom_line() +
  ylab("Divergence") +
  xlab("Nr1i3") +
  theme_bw() +
  theme(legend.key = element_blank())

```



### 3 Discrete Bayesian network example

In this example, we examine a completely discrete network. 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* Brem and Kruglyak (2005). The original dataset consists of expression values reported as  $\log_2(\text{sample}/\text{reference})$  for 6,216 genes. The full data can be accessed in Gene Expression Omnibus (GEO) - accession number GSE1990 Barrett et al. (2013). The subset of genes was identified after filtering, linkage analysis and model building. Briefly, 901 expression values mapped to the YeastNet database H. Kim et al. (2013). Linkage analysis was performed on these traits using R/qtl Broman and Sen (2009). 369 genes that had a significant QTL were used as predictors in an elastic net regression model with *COX10* as the response variable Zou and Hastie (2005). The optimal shrinkage parameter was identified as 0.086 using a 10-fold cross validation scheme. The resulting model shrunk all but 37 regression coefficients to zero. The 38 genes including *COX10* and their 12 SNP markers corresponding to their QTL were included as variables for the graphical model. This set of 38 genes and their corresponding 12 SNP markers were

identified and included in the yeast dataset.

In order to derive discrete nodes for network analysis, the gene expression values were dichotomized at the median. The discrete variables are binary genotype states that indicate the parental strain of origin. To demonstrate compatibility with existing packages, as a starting point, the structure of the BN is learned using a hill-climbing method in the `bnlearn` package. The modeling assumptions that require genotypes to be upstream of phenotype can be directly encoded using the `blacklist` option. The isolated nodes are removed from the network.

```
library("bnlearn")
library("igraph")
data(yeast)
node.names <- names(yeast)
geno <- node.names[1:12]
pheno <- node.names[13:50]

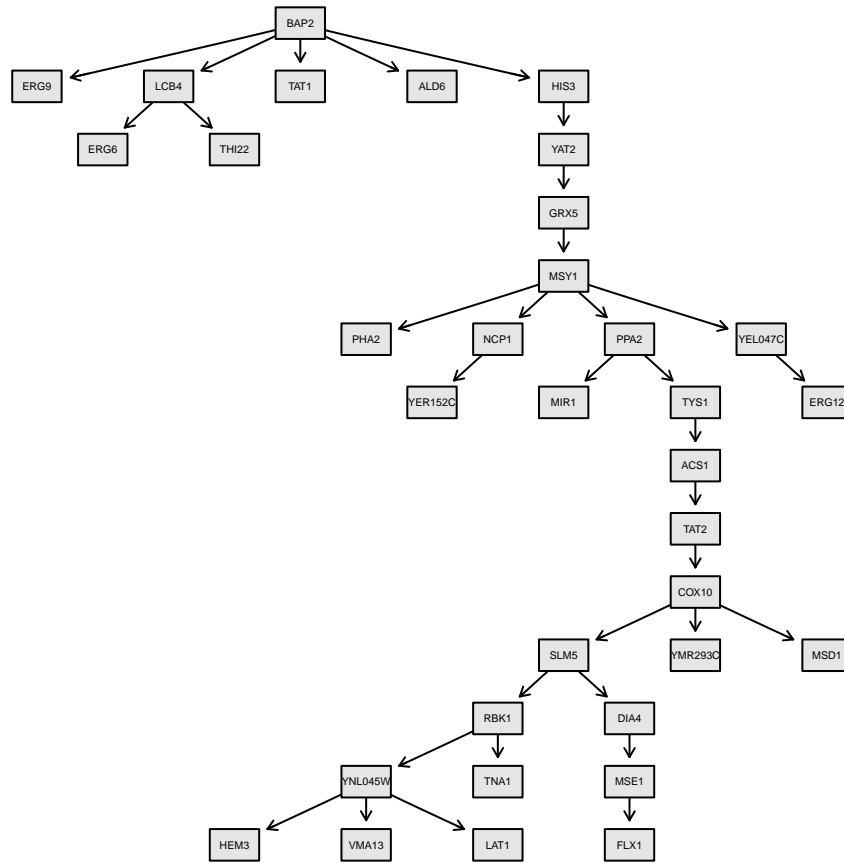
bl <- rbind(expand.grid(geno, geno), expand.grid(pheno, geno))
names(bl) <- c("from", "to")

fit <- bnlearn::hc(yeast, blacklist = bl)
dag <- as.graphNEL(fit)
dag.graph <- igraph.from.graphNEL(dag)

deg <- igraph::degree(dag.graph)
nodes <- names(deg)[deg>0]
dag.graph <- induced_subgraph(dag.graph, nodes)
dag <- igraph.to.graphNEL(dag.graph)
```

After the network structure being learnt, the models and cluster tree can be built in the same procedure as a CG-BN. BayesNetBP brings networks with purely discrete, continuous, or mixed (both discrete and continuous) types of nodes under the same framework, so they can be analyzed conveniently using the identical procedure. The following figure shows the structure of constructed semi-elimination tree, which is plotted using the `PlotTree` function. The nodes in the cluster tree plot are named by their corresponding elimination nodes.

```
node.class <- rep(TRUE, length(nodes))
names(node.class) <- nodes
cst <- ClusterTreeCompile(dag=dag, node.class=node.class)
models <- LocalModelCompile(data=yeast, dag=dag, node.class=node.class)
tree.init <- ElimTreeInitialize(tree=cst$tree.graph, dag=cst$dag, model=models,
  node.sets=cst$cluster.sets, node.class=cst$node.class)
tree.init.p <- PropagateDBN(tree.init)
PlotTree(tree.init.p)
```

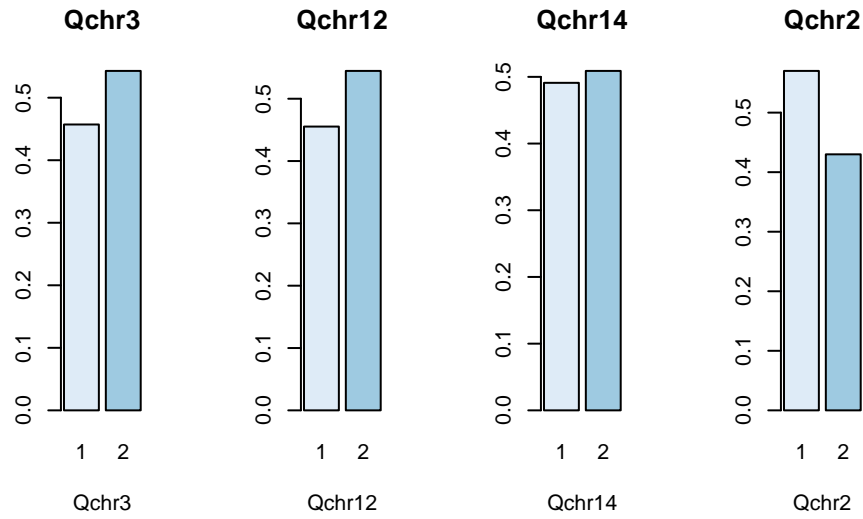


Evidence absorption and marginal computation can be performed in the exactly same manner as in the CG-BN example. The marginals can be visualized using `PlotMarginals`.

```

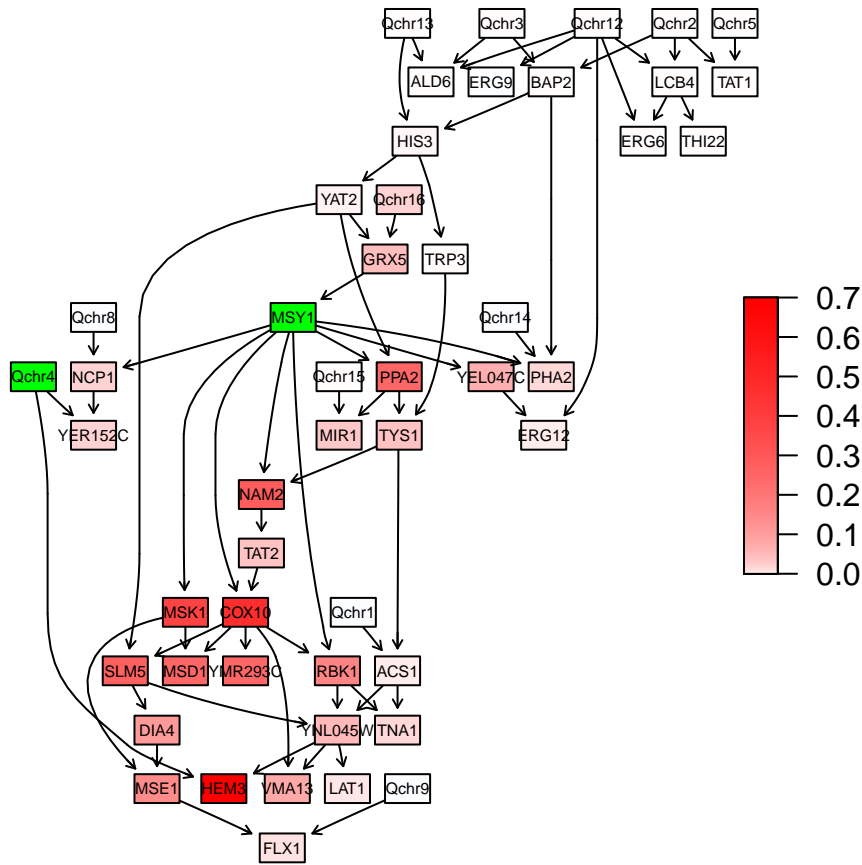
tree.post <- AbsorbEvidence(tree.init.p, c("MSY1", "Qchr4"), list("1", "2"))
marg.yeast <- Marginals(tree.post, nodes[2:5])
PlotMarginals(marg.yeast)

```



The PlotCGBN function can still be used to compare the marginals from two models.

```
div <- PlotCGBN(tree.init.p, tree.post, fontsize = 24)
```



Queries on joint and conditional distributions of factors is also straightforward and can be output in tabular

form.

```
FactorQuery(tree.post, nodes[2:5], mode="joint")
```

##	Qchr2	Qchr3	Qchr14	Qchr12	prob
## 1	1	1	1	1	0.05827458
## 2	1	1	1	2	0.06970097
## 3	1	1	2	1	0.06039366
## 4	1	1	2	2	0.07223555
## 5	1	2	1	1	0.06919652
## 6	1	2	1	2	0.08276447
## 7	1	2	2	1	0.07171276
## 8	1	2	2	2	0.08577408
## 9	2	1	1	1	0.04395494
## 10	2	1	1	2	0.05257356
## 11	2	1	2	1	0.04555331
## 12	2	1	2	2	0.05448533
## 13	2	2	1	1	0.05218683
## 14	2	2	1	2	0.06241955
## 15	2	2	2	1	0.05408454
## 16	2	2	2	2	0.06468935

```
FactorQuery(tree.post, "ERG9", mode="conditional")
```

##	ERG9	Qchr12	prob
## 1	-1	1	0.07843137
## 2	1	1	0.92156863
## 3	-1	2	0.85245902
## 4	1	2	0.14754098

## 4 Shiny app

The function `runBayesNetApp` invokes 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. In order to run the app, the packages `shiny`, `googleVis` and `r-cytoscape.js` need to be installed and loaded. The `r-cytoscape.js` is currently only available on GitHub repository and can be installed using the following code.

```
library("devtools")
devtools::install_github("cytoscape/r-cytoscape.js")
runBayesNetApp()
```

The console of BayesNetBP Shiny app comprises of three panels. The first part controls the model loading and network layouts. It also allows user to subset the network to facilitate visualization. The **Expand** function can trace the ancestors, descendants, or both, of a selected node in a stepwise manner. The expanded nodes will be colored orange. By clicking **Add to list**, the expanded nodes will be selected and shown purple. The user can then continue selecting additional nodes by using **Expand** and **Add to list** functions. After selecting desired node sets, the user can subset the graph by clicking **Subset**. The nodes in subsetted graph retain all properties before subsetting, including their colors and divergence. Other operations can also be performed on the reduced network.

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. The nodes with evidence absorbed will be colored green when the absorption is complete. Marginals of the 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 **Plot Marginals**. The **Shift in Marginals** function computes the signed and symmetric Kullback-Liebler divergence for all applicable nodes in the network, and colors the nodes in a similar manner as the function **PlotCGBN**.

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 a menu, and to select whose divergence to be calculated by firstly selecting the node on the graph and then clicking **Add to Plot List**. 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.

## References

- Alvord, Gregory, Jean Roayaei, Robert Stephens, Michael W Baseler, H Clifford Lane, and Richard A Lempicki. 2007. "The DAVID Gene Functional Classification Tool: A Novel Biological Module-Centric Algorithm to Functionally Analyze Large Gene Lists." *Genome Biol* 8 (9): 183.
- Barrett, Tanya, Stephen E Wilhite, Pierre Ledoux, Carlos Evangelista, Irene F Kim, Maxim Tomashevsky, Kimberly A Marshall, et al. 2013. "NCBI Geo: Archive for Functional Genomics Data Sets?update." *Nucleic Acids Research* 41 (D1). Oxford Univ Press: D991–D995.
- Brem, Rachel B, and Leonid Kruglyak. 2005. "The Landscape of Genetic Complexity Across 5,700 Gene Expression Traits in Yeast." *Proceedings of the National Academy of Sciences of the United States of America* 102 (5). National Acad Sciences: 1572–7.
- Broman, K. W., and S. Sen. 2009. *A Guide to QTL Mapping with R/Qtl*. Springer.
- Cowell, Robert G. 2005. "Local Propagation in Conditional Gaussian Bayesian Networks." *Journal of Machine Learning Research* 6 (Sep): 1517–50.
- Hugin Expert A/S. 2017. *Hugin Software*. <http://www.hugin.com/>.
- Højsgaard, Søren. 2012. "Graphical Independence Networks with the GRain Package for R." *Journal of Statistical Software* 46 (10): 1–26. <http://www.jstatsoft.org/v46/i10/>.
- Kim, Hanhae, Junha Shin, Eiru Kim, Hyojin Kim, Sohyun Hwang, Jung Eun Shim, and Insuk Lee. 2013. "YeastNet V3: A Public Database of Data-Specific and Integrated Functional Gene Networks for *Saccharomyces Cerevisiae*." *Nucleic Acids Research*. Oxford Univ Press, gkt981.
- Koller, Daphne, and Nir Friedman. 2009. *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press.
- Konis, K. 2017. *RHugin Package*. <http://rhugin.r-forge.r-project.org/>.
- Lauritzen, Steffen L. 1996. *Graphical Models*. Oxford University Press.
- Leduc, Magalie S, Rachael Hageman Blair, Ricardo A Verdugo, Shirng-Wern Tsaih, Kenneth Walsh, Gary A Churchill, and Beverly Paigen. 2012. "Using Bioinformatics and Systems Genetics to Dissect Hdl Cholesterol Levels in an Mrl/Mpj X Sm/J Intercross." *Journal of Lipid Research*. ASBMB, jlr–M025833.
- Scutari, Marco. 2010. "Learning Bayesian Networks With the bnlearn R Package." *Journal of Statistical Software* 35 (3): 1–22.
- Zou, Hui, and Trevor Hastie. 2005. "Regularization and Variable Selection via the Elastic Net." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 67 (2). Wiley Online Library: 301–20.