Journal of Statistical Software

June 2020, Volume 94, Issue 3. *doi: 10.18637/jss.v094.i03*

BayesNetBP: An R Package for Probabilistic Reasoning in Bayesian Networks

Han Yu
Roswell Park Comprehensive Cancer Center

Janhavi Moharil
HTG Molecular Diagnostics

Rachael Hageman Blair
University at Buffalo

Abstract

The **BayesNetBP** package has been developed for probabilistic reasoning and visualization in Bayesian networks with nodes that are purely discrete, continuous or mixed (discrete and continuous). Probabilistic reasoning enables a user to absorb information into a Bayesian network and make queries about how the probabilities within the network change in light of new information. The package was developed in the R programming language and is freely available from the Comprehensive R Archive Network. A **shiny** app with **Cytoscape** widgets provides an interactive interface for evidence absorption, queries, and visualizations.

Keywords: Bayesian network, belief propagation, R, **shiny**, gene networks, visualization.

1. Introduction

Probabilistic graphical models (PGMs) provide a general framework for representing relationships between a set of random variables (Koller and Friedman 2009). In a PGM, nodes in the graph (aka network) represent random variables, and the edges between them encode information about the dependencies (directed) and associations (undirected) between them. Information gained from PGMs may provide novel insights into complex relationships. In broad terms, there are two layers of information that can be gained from PGMs: (1) through the structure of the network, and (2) by performing probabilistic reasoning (e.g., by computing marginal, conditional or joint distributions) given a known or estimated network. This work focuses on the latter, on a subclass of directed PGMs known as Bayesian networks (BNs; Pearl 2014; Koller and Friedman 2009).

In a BN, the directed edges of the network encode the conditional independencies between variables. Thus, the structure of the network conveys important information about the direct and indirect dependencies between variables. The BN paradigm also enables probabilistic queries within the network (Koller and Friedman 2009). Example queries include the probability of variables being assigned particular values, conditional probabilities of latent variables given values of observable values, or the probability of an outcome variable after causal evidence is taken into account. The process of performing probabilistic queries within a network is broadly known as *probabilistic reasoning*. Queries are performed by propagating new evidence (information) through the network using a method known as *belief propagation*. Probabilistic reasoning has been utilized in many application areas that benefit from reasoning as a means to better inform decision making (Koller and Friedman 2009), e.g., predicting traffic flow (Castillo, Menéndez, and Sánchez-Cambronero 2008; Sun, Zhang, and Yu 2006), clinical decision support (Kahn Jr, Roberts, Shaffer, and Haddawy 1997; Pradhan, Provan, Middleton, and Henrion 1994; Sesen, Nicholson, Banares-Alcantara, Kadir, and Brady 2013) and the prediction of gene network perturbations (Moharil, May, Gaile, and Blair 2016).

Probabilistic reasoning with belief propagation is performed *post hoc* to structural and parameter learning tasks. Our emphasis is on this post-hoc analysis and visualizations in BNs. In fact, methods for structural and parameter learning are separate, and developed independently, from the inference algorithms used for probabilistic reasoning. For this reason, the R (R Core Team 2020) package **BayesNetBP** (Yu 2020), available from the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=BayesNetBP`, integrates with the many existing packages and algorithms for structural and parameter learning (Højsgaard, Edwards, and Lauritzen 2012), and we showcase different examples in our applications. Belief propagation algorithms require different modifications depending on the type of nodes in the network, e.g., discrete, continuous and a mixture thereof. Package **BayesNetBP** provides the flexibility to accommodate these common BN representations.

The **BayesNetBP** package has the following major advantages. (1) To the authors' knowledge, **BayesNetBP** is the first open source package to facilitate exact probabilistic reasoning in the three most common types of BNs (discrete multinomial, continuous Gaussian and conditional Gaussian Bayesian networks (CG-BNs)). (2) **BayesNetBP** is the first R package that enables exact probabilistic reasoning in CG-BNs without commercial dependencies. (3) The package provides novel visualizations for probabilistic reasoning in the network. (4) The inference tools for probabilistic reasoning connect seamlessly with existing graphical modeling tools in R for structural learning. (5) The package is also supported through a **shiny** (Chang, Cheng, Allaire, Xie, and McPherson 2020) app that is coupled with **Cytoscape** visualizations (Shannon *et al.* 2003) designed to be accessible to the non-technical user.

This paper is organized as follows. In Section 2, we place the **BayesNetBP** package in the context of existing graphical modeling tools in R. In Section 3 we provide a brief description of the theory of BNs (Section 3.1), motivate probabilistic reasoning and demonstrate select features of the package (Section 3.2), provide an overview of the belief propagation procedures implemented in **BayesNetBP** (Section 3.3), and describe measures used to quantify probabilistic changes for the comparisons and visualizations of networks (Section 3.4). Section 4 provides examples in statistical genetics for CG-BNs (Section 4.1) and discrete BNs (Section 4.2). In Section 4.3, we describe the **shiny** app. Finally, conclusions and future directions are outlined in Section 5.

2. Existing tools and implementation

The general workflow of the **BayesNetBP** is illustrated in Figure 1. Note that network structure is the input from which **BayesNetBP** provides post-hoc probabilistic reasoning and visualizations. **BayesNetBP** can be seamlessly integrated with other tools in R that provide structural learning (Højsgaard *et al.* 2012; Scutari and Denis 2014). For example, **bnlearn** (Scutari 2010, 2017), **catnet** (Balov and Salzman 2020), **deal** (Boettcher and Dethlefsen 2003), **pcalg** (Kalisch, Mächler, Colombo, Maathuis, and Bühlmann 2012) and **RHugin** (Konis 2017), all provide structural learning for different types of BNs that can serve as input to **BayesNetBP**. The network input can also be specified by a user based on prior knowledge. **BayesNetBP** is compatible with networks that are discrete (multinomial), continuous (Gaussian) or a mixture of discrete and continuous variables (CG-BN).

In the R programming language, there are only two packages that can perform exact inference for probabilistic reasoning in BNs, **RHugin** and **gRain** (Højsgaard 2012). **RHugin** can be used

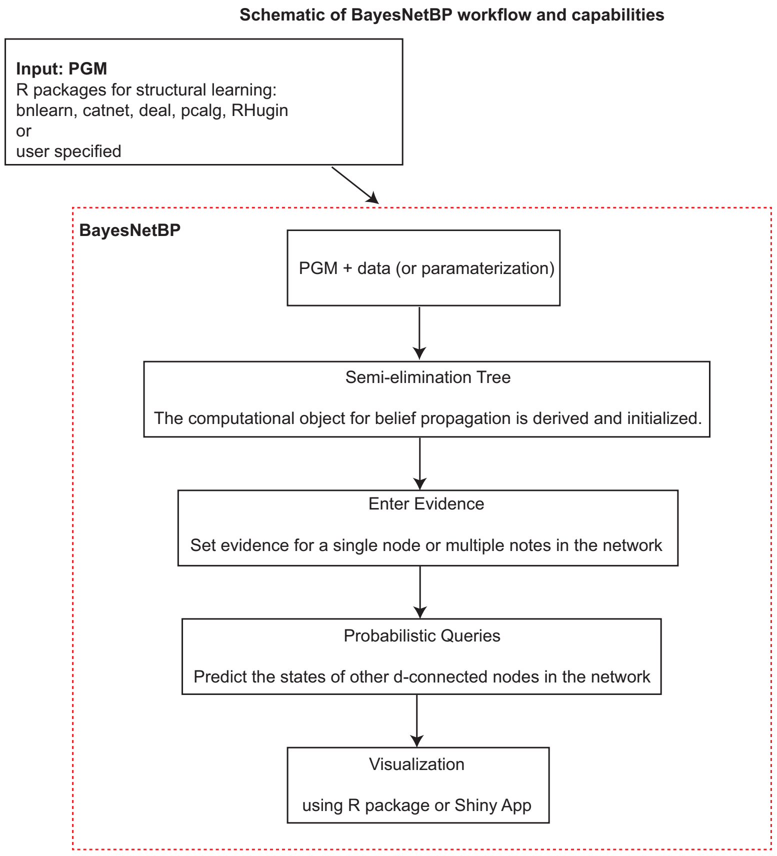

Schematic of BayesNetBP workflow and capabilities
Input: PGM
R packages for structural learning:
bnlearn, catnet, deal, pcalg, RHugin
or
user specified
BayesNetBP
PGM + data (or paramaterization)
Semi-elimination Tree
The computational object for belief propagation is derived and initialized.
Enter Evidence
Set evidence for a single node or multiple notes in the network
Probabilistic Queries
Predict the states of other d-connected nodes in the network
Visualization
using R package or Shiny App

Figure 1: The **BayesNetBP** workflow. The program integrates with existing tools for structural learning available in R, or the network can be user-specified based on prior knowledge. Parameters are estimated for the local conditional distributions using data, or are user-defined. A semi-elimination tree is inferred and initialized, which is the computational object for belief propagation. The user can enter evidence for a node(s) in the network, and perform probabilistic queries to predict the state of other nodes in the network. These queries can be visualized using the R package or in a **shiny** app.

for structural learning and exact inference for BNs, but relies on the commercial software, **Hugin** (**Hugin** Expert A/S 2017). A free demo version called **huginlite** can be used in connection with **RHugin**, but the models are limited to smaller networks and data (50 states and 500 cases) for the demo version. The **gRain** package can handle large datasets and networks, but it only supports probabilistic reasoning in purely discrete networks. **BayesNetBP** also supports inferences that **gRain** can do on discrete BNs, but can also preform probabilistic reasoning on continuous BNs and CG-BNs. Package **BayesNetBP** fills a major gap in the graphical modeling tools available in R. To our knowledge, **BayesNetBP** is the only fully open source R package to support exact inference in CG-BNs. **BayesNetBP** also provides tools for quantification and visualization of distributional changes.

The R package **ramidst** (Salmeron *et al.* 2016) links to the **AMIDST** (Masegosa *et al.* 2019) toolbox for scalable, but only approximate inference algorithms for BNs written in Java (Gosling, Joy, Steele, and Bracha 2000). The package was recently removed from the CRAN repository for check issues, but the Java toolbox remains functional. Notably, the most probable explanation (MPE) and maximum a posteriori (MAP) inferences that are utilized only output information associated with the highest probabilities, and thus do not capture the uncertainty of the approximate inference. The package also implements approximate inference based on importance sampling for single target variables. In cases of massive networks, the **ramidst** package presents computational advantages over **BayesNetBP**. However, in moderate to large networks (several hundreds of nodes), **BayesNetBP** offers more precision, and a more comprehensive output.

3. Theory

3.1. Bayesian networks: Introduction and terminology

BayesNetBP exclusively targets BNs. We begin with a simplistic BN for an emission problem (Figure 2) that will be used to demonstrate terminology and motivate probabilistic reasoning with **BayesNetBP** throughout this section. The discrete variables are filter state (F_s), waste type (W), and burning regimen (B), and the continuous variables are metals in waste (M_i), metals emission (M_o), filter efficiency (E), dust emission (D), CO_2 concentration in the emission (C), and light penetrability (L). The network can be used to predict emission of heavy metals, diagnose the stability of burning regimen and the filter state, and so forth. A brief description is provided below, and additional details and model parameters are provided in Lauritzen (1992).

> The emission from a waste incinerator differs due to compositional differences in incoming waste. Another factor is the waste burning regimen which can be monitored by measuring the concentration of CO_2 emissions. The filter efficiency depends on the technical state of the electrofilter and the amount and composition of waste. The emission of heavy metals depends both on the concentration of metals in the incoming waste and the emission of dust particulates in general. The emission of dust is monitored by measuring the penetrability of light.

BNs are directed acyclic graphs (DAGs), a network with only directed edges and no cycles. For example, $D \to W$ is not an allowable edge in the emissions BN because it would create

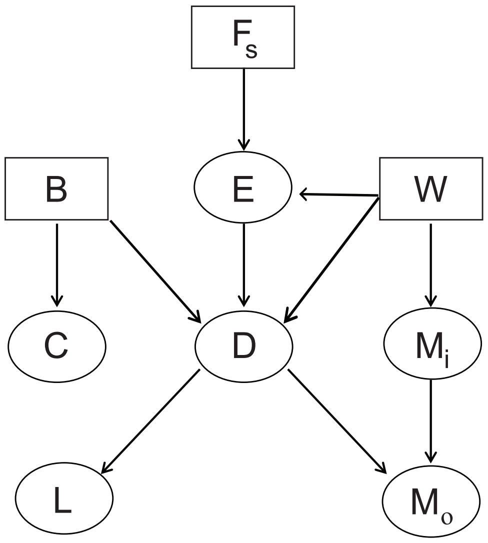


Figure 2: BN for the emission problem. The discrete variables are filter state (F_s = $\{\text{intact}, \text{defect}\}$), waste type ($W$ = $\{\text{industrial}, \text{household}\}$), and burning regimen ($B$ = $\{\text{stable}, \text{unstable}\}$). The continuous variables are metals in waste (M_i), metals emission (M_o), filter efficiency (E), dust emission (D), CO_2 concentration in the emission (C), and light penetrability (L).

a cycle $D \rightleftarrows W$ (Figure 2). The relationships between nodes in a network are described using a standard familial terminology. *Parent nodes* are connected by directed edges to their downstream *child nodes* (e.g., $\{D, M_i\}$ are parents of M_0). We define a *local distribution* as the conditional probability distribution defined by conditioning a child node on its parent nodes. Specifically, the local distributions are defined for each node conditional on the set of parents, e.g., $f\,(M_0 \mid D, M_i) \sim N(\beta_0 + \beta_1 \cdot D + \beta_2 \cdot M_i, \sigma^2)$, where $\{\beta_0, \beta_1, \beta_2\}$ denote the estimated coefficients from regressing a child node on its parents. The full specification of local distributions for a BN is the *parameterization* of the network. For purely continuous or purely discrete BNs, these local distributions are often defined as multivariate Gaussian (Geiger and Heckerman 1994) or multinomial (Heckerman, Geiger, and Chickering 1995), respectively.

The structure of a BN encodes a set of conditional independencies between random variables, $X = (X_1, X_2, \ldots, X_n)$, which can be used to represent the joint distribution in compact factored form. BNs follow the *Markov assumption*, which states that each node is conditionally independent of its non-descendants (nodes that are not parents) in the network, given the value of its parent nodes, e.g, $f\,(M_0 \mid D, M_i, W) = f\,(M_0 \mid D, M_i)$. This assumption enables the joint distribution to be expressed as a product of conditional probabilities (Lauritzen 1996; Koller and Friedman 2009):

$$P\,(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P\,(X_i \mid \mathrm{pa}(X_i), \Theta_i)\,,$$

where $\mathrm{pa}(X_i)$ are the parent nodes of X_i, and Θ_i denotes the parameters of the local distribution. Notably, the BN paradigm is especially powerful in high-dimensions, where it may not be tractable to estimate the parameters necessary to represent the joint distribution directly.

D-separation (dependence separation) is an important property that ultimately governs how information can flow through a network for probabilistic reasoning (Spirtes, Glymour, and Scheines 2000; Pearl 2014). In a BN, if two nodes, X and Y, are d-separated relative to a set of variables Z, then they are independent conditional on Z in all probability distributions represented by the BN. In other words, X provides no additional information about Y if Z is known. A pair of nodes is *d-separated* if all paths (sets of connected edges) between them contain what is known as a *v-structure* (aka collider). For example, D and M_i are d-separated because M_0 forms a v-structure between them, formed by $D \rightarrow M_0$ and $M_i \rightarrow M_0$. However, when information is absorbed into a BN, the patterns of d-separation within the graph change, see Koller and Friedman (2009) for details. Nodes that are not d-separated are known as *d-connected*.

Conditional Gaussian Bayesian networks

Conditional Gaussian Bayesian networks (CG-BNs) include a mixture of discrete and continuous variables (Figure 2). Additional modeling assumptions on the structure of the BN are necessary in CG-BNs for the local distributions. Following Lauritzen (1996), we denote the set of discrete nodes as Δ, and the set of continuous nodes as Γ. For a continuous variable $Y = X_j \in \Gamma$, we define a local distribution as a conditional Gaussian regression (CG-regression), denoted by Λ, on the states of the discrete parents I of child node Y:

$$\Lambda(Y \mid I = i, Z = z) = N\left(\alpha(i) + \beta(i)^\top z, \sigma^2(i)\right),$$

where $\alpha(i)$ is the mean of the regression, β denotes the regression coefficients, $Z \in \Gamma$ are the the continuous parents of Y, and the variance, $\sigma^2(i)$ depends only on the discrete states of the parent nodes. Importantly, in a CG-BN, discrete nodes can be parents of continuous nodes, but not vice versa. Networks known as *chain graphs* also contain a mixture of discrete and continuous nodes, as well as directed and undirected edges (Cox and Wermuth 1993). Chain graphs do not require the assumptions made for a CG-BN and are outside of the scope of the **BayesNetBP** package.

Structure and parameter learning

BNs require up to two layers of learning for the network structure and parameters of the local distributions. The learning requirements depend on the data, user knowledge and the modeling objectives. For example, a user can construct the graph based on prior domain knowledge about the relationships between variables. In this scenario, only parameterization of the local distributions is required, which can also be either user-specified or directly inferred from the data.

In many cases, the network structure is not known and has to be learned. Identifying the BN that best explains the data is an NP-hard problem (Chickering, Heckerman, and Meek 2004). The **BayesNetBP** package does not provide structural learning, but can interface with many of the existing and popular learning packages described in Section 2 (Højsgaard *et al.* 2012; Scutari and Denis 2014). Structural learning algorithms can be broadly classified as constraint-based, score-based, or as a hybrid of these. Briefly, we describe the basics of three foundational approaches, from which more complex modification have been developed. Constraint-based methods are inspired by the inductive causation (IC) algorithm (Verma and Pearl 1991). The IC algorithm initially carries out sets of conditional independence tests to identify an *undirected graph*, a graph with undirected edges, which are then directed in a way that ensures that there are no directed cycles. Score-based methods rely on the definition of a measure that reflects how well the data supports the structure of the network. For example, the Bayesian information criterion (Yu, Smith, Wang, Hartemink, and Jarvis 2004) is a popular score that naturally favors sparser graphs in terms of the number of edges. Probabilistic scores enable the integration of user knowledge about node relationships into the structural learning process. This can be achieved by (1) constraining the structural learning procedure to search over networks that contain *known* relationships, or (2) setting a graphical prior, e.g. see Werhli and Husmeier (2007) and Mukherjee and Speed (2008). Hybrid approaches blend the constraint and score based methods. For example, the PC algorithm (named after its authors, Peter and Clark; Spirtes *et al.* 2000) depends on statistical independence tests for all pairs of variables to derive an undirected skeleton, v-structures in the graph are identified and remaining edges are directed at random to enforce a DAG structure. Several other hybrid structural learning algorithms have been developed, and can be implemented in different R packages, see Højsgaard *et al.* (2012) and Scutari and Denis (2014) for a more in depth description.

3.2. A simple motivating example

In order to motivate probabilistic reasoning with **BayeNetBP**, consider a simple emissions network (Figure 2). Suppose that we want to make informed decisions about operations and possible malfunctions. For example, we may want to query the CG-BN for the distribution of metals emission (M_o), filter efficiency (E), dust emission (D) after observing the values of waste type (W), CO_2 concentration in the emission (C), and light penetrability (L).

The **BayesNetBP** can be installed from CRAN.

```
R> install.packages("BayesNetBP")
```

To get started, we need to create a ‘`ClusterTree`’ object. The initialization requires as input a ‘`graphNEL`’ object, a data frame and a vector to indicate explicitly whether the variables are discrete. The `emission1000` data in **BayesNetBP** package includes the network structure and a simulated dataset based on the parameterization from Lauritzen (1992).

```
R> library("BayesNetBP")
R> data("emission1000", package = "BayesNetBP")
R> node.class <- rep(c(TRUE, FALSE), c(3, 6))
R> names(node.class) <- c("B", "Fs", "W", "L", "Mo", "D", "Mi", "C", "E")
R> node.class

    B    Fs     W     L    Mo     D    Mi     C     E 
 TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE 

R> tree.init <- Initializer(dag = emission1000$dag,
+    data = emission1000$data, node.class = node.class)
```

Suppose that we observe some new evidence (information) about variables in our network. Probabilistic reasoning is performing queries about probability distributions (marginal, conditional and joint) in the BN once this new evidence has been absorbed and propagated through the network. For example, suppose we observe that waste has been of industrial type (W = "industrial"), the light penetrability has been recently measured ($L = 1.1$), and the CO_2 concentration in the emission was also measured ($C = -0.9$). `AbsorbEvidence` can be used to absorb evidence into a '`ClusterTree`' object. Updated probability distributions for marginal, conditional and joint distributions can be obtained for the remaining variables using `FactorQuery`. For example, we can query the joint probability distribution of filter state and burning regime, before and after absorbing and propagating the new evidence.

```
R> tree.post <- AbsorbEvidence(tree.init, vars = c("W", "L", "C"),
+    values = list("industrial", 1.1, -0.9))
R> FactorQuery(tree.init, vars = c("Fs", "B"), mode = "joint")

         B     Fs     prob
1   stable defect 0.040138
2   stable intact 0.813862
3 unstable defect 0.006862
4 unstable intact 0.139138

R> FactorQuery(tree.post, vars = c("Fs", "B"), mode = "joint")

      Fs        B         prob
1 defect   stable 8.989126e-05
2 defect unstable 4.173244e-04
3 intact   stable 1.050902e-02
4 intact unstable 9.889838e-01
```

From this, we can see that under initial (typical) conditions, it is most likely that the filter is intact and the burning regime is stable, with the joint distribution $P(F_s = \text{intact}, \text{B} = \text{stable}) = 0.81$. However, once this new observed evidence is taken into account, the updated probabilities suggest and intact filter, but unstable burning regime, with $P(F_s = \text{intact}, \text{B} = \text{unstable}) = 0.99$. We can also look at the updated marginal probabilities to further assess the situation.

```
R> marg.pre <- Marginals(tree.init, c("E", "D", "Mi", "Mo"))
R> SummaryMarginals(marg.pre)

         Mean        SD n
E  -3.2585315 0.6913124 4
D   3.0267606 0.7482027 8
Mi -0.2197765 0.4584857 2
Mo  2.8068047 0.8382861 8

R> marg.post <- Marginals(tree.post, c("E", "D", "Mi", "Mo"))
R> SummaryMarginals(marg.post)

         Mean         SD n
E  -3.8982243 0.07903811 4
```

```
D   3.5698591 0.38444326 4
Mi  0.5035502 0.10328602 1
Mo  4.0687900 0.39975314 4
```

The updated marginal probabilities suggest that with an unstable burning regime it is also likely to have reduced filter efficiency and an increase in dust and metals. The belief propagation procedure in **BayesNetBP** facilitates probabilistic inquiries of this type, and the inference for the updated distributions is exact. Note that the same setting was used in Lauritzen (1992) and Lauritzen and Jensen (2001), and negligible differences are due to the fact that the local distributions were estimated using simulated data.

3.3. Belief propagation

BNs enable probabilistic reasoning (queries) within the network through a process known as belief propagation. Belief propagation algorithms for probabilistic reasoning in a BN typically involve operations along a computational object known as a *cluster tree*, an undirected graph with no cycles derived from the BN. The nodes in a cluster tree are *clusters*, C_i, from the BN such that $C_i \subseteq \{X_1, X_2, \dots X_n\}$. *Sepsets*, defined as $S_{i,j} = C_i \cap C_j$, are the edges in a cluster tree. We define *potentials* as non-negative functions that parameterize the BN and represent the local distributions. They are in the form of conditional probability tables for discrete variables (ψ), and CG regressions for continuous variables (Λ). The cluster tree serves as a computational object that absorbs evidence and passes messages (functions of cluster potentials) between clusters and ultimately regulates the updating of potentials.

BayesNetBP supports reasoning with *soft evidence* that specifies likelihoods, or *hard evidence* that specifies actual values of individual nodes. The package also supports probabilistic reasoning in discrete multinomial, Gaussian BNs and CG-BNs. However, reasoning in CG-BNs requires a specialized set of operations to accommodate both discrete and continuous variables. Lauritzen (1992) developed an approach to reasoning in CG-BNs that was later modified to alleviate numerical instabilities (Lauritzen and Jensen 2001). This propagation scheme is currently implemented in the commercial software **Hugin** (**Hugin** Expert A/S 2017). Cowell (2005) developed an approach to reasoning in CG-BNs that is simpler to implement, and can be generalized to accommodated purely discrete or continuous BNs. **BayesNetBP** implements this approach and its generalizations, and provides unique visualizations to facilitate understanding of the probabilistic reasoning results. **BayesNetBP** simultaneously accommodates discrete and continuous clusters by compartmentalizing the cluster tree and the corresponding operations accordingly, and then interfacing the computation between these two compartments.

Construction of the elimination tree

A special type of cluster tree, known as an *elimination tree*, is the main computational object of the **BayesNetBP** package. Following Cowell (2005), we briefly describe four basic steps to constructing an elimination tree from a BN.

1. **Moralization:** The DAG is *moralized* by eliminating the directionality of the edges in G and connecting all parent nodes that have a common child (Figure 3A).

2. **Triangulation based on elimination ordering:** *Elimination ordering* is derived such that for every directed edge $X \to Y$ in the network, Y comes before X in the ordering. Continuous nodes always precede discrete nodes in the elimination order, e.g., $\{L, M_0, D, M_i, C, E, B, F_s\}$ is an elimination ordering for Figure 3A. Triangularization depends on the elimination ordering, as well as *neighbors* in a graph, which are simply nodes that share an edge. Specifically, in the triangularization, an undirected edge is added between nodes X and Y in the moral graph if they share a common *neighbor* and Z, and Y appears later than X and Z in the elimination ordering (Figure 3B).

3. **Cluster sets:** For each node X_i, a cluster set C_i is formed, which contains all of its neighbors in the triangulated graph that appear later in the elimination ordering, and itself (e.g., Figure 3C). X_i is termed the *elimination node* of C_i and is the unique identifier of the cluster set.

4. **Elimination tree formation:** Let C_i and C_j be two cluster sets with more than one node and with elimination nodes X_i and X_j, respectively. A directed edge is added between C_i and C_j ($C_i \to C_j$) if the elimination node X_i appears first in the elimination ordering among $C_j \backslash \{X_j\}$. Certain discrete clusters can be merged to improved computational efficiency, as described in Algorithm 3.3 by Cowell (2005). The obtained cluster tree is called strong semi-elimination tree. This construction satisfies the running intersection property, which ensures that if a variable appears in more than one cluster set in the elimination tree, then this variable will appear in every cluster set in the path between them (Koller and Friedman 2009, Figure 3D).

In practice, the number of parents for a node is usually very small relative to n, so the above steps typically have time complexity of $O(n)$. Suppose the maximum number of parents for a node is m, then in the worst case Step 1 and 2 can have time complexity of $O(nm^2)$, while Step 4 can be up to $O(n^2)$ (Cowell 2005). Elimination trees built from CG-BNs have two separated compartments. Clusters in the discrete compartment only contain discrete nodes, while those in the continuous part can include both discrete and continuous nodes. The *boundary cluster* is the discrete cluster that neighbors the continuous compartment (e.g., $\{B, W, F_s\}$ in Figure 3D). A *strong root* is a cluster such that, when a sepset between two clusters is not purely discrete, the cluster furthest away from the root has only continuous nodes beyond the sepset (Lauritzen 1992). Importantly, due to the compartmentalization of discrete clusters, a strong root will always exist in a CG-BN since any cluster containing discrete nodes (e.g., $\{B, W, F_s\}$ in Figure 3D) naturally satisfies the strong root conditions (Lauritzen 1992). Otherwise, the cluster whose elimination node is at the end of the elimination order will serve as the root.

After construction, each node must be assigned to a cluster that contains the node itself and its parents (local family) in the BN. In a CG-BN, discrete nodes must be assigned to a cluster that contains the local family, but may not include any continuous variables. Importantly, this will always be satisfied due to the elimination ordering in a CG-BN, which enforces the elimination of continuous nodes before the discrete ones.

An overview of the belief propagation procedure

BayesNetBP couples a series of four algorithms described in Cowell (2005) with classic procedures of belief propagation for discrete networks (Koller and Friedman 2009). The algorithms

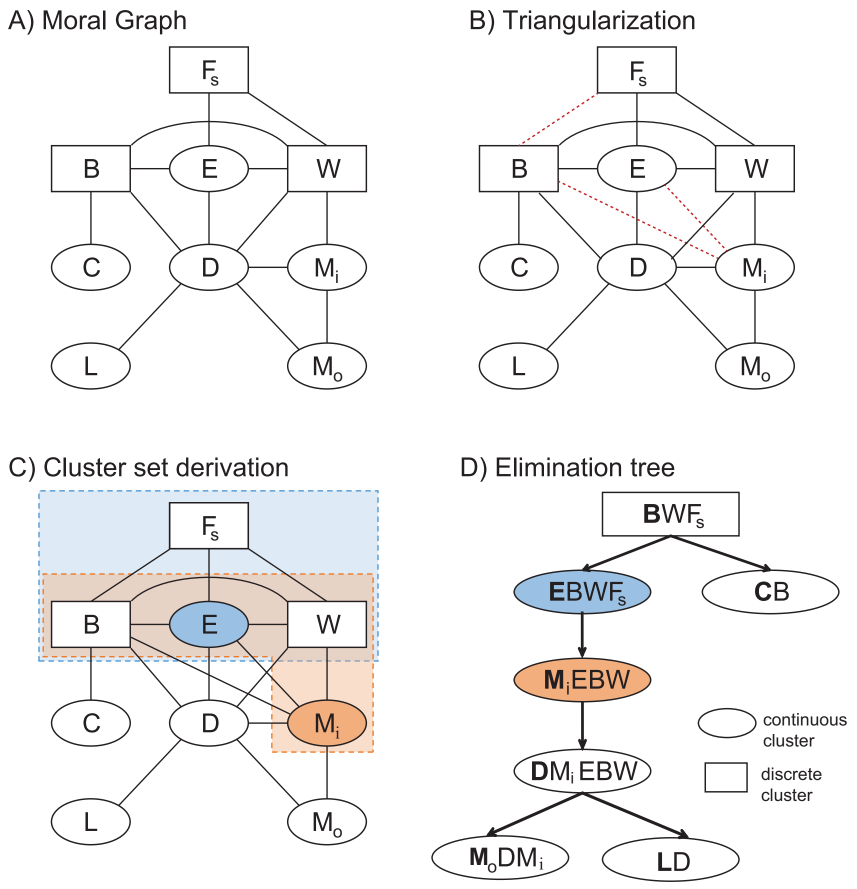


Figure 3: Construction of an elimination tree for the emissions BN of Figure 2. (A) The graph is moralized by dropping the directionality of edges in the BN, and connecting nodes with a common child, e.g., an undirected edge is added between D and M_i. (B) The moralized graph is triangulated by the elimination ordering $\{L, M_o, D, M_i, C, E, B, W, F_s\}$. (C) Cluster sets are derived. For example, the cluster set of E is formed by itself and all its neighbor nodes that appear after E in the elimination ordering, which are $\{B, W, F_s\}$ (blue). Likewise, the cluster set of M_i is formed by itself and $\{E, B, W\}$ (orange). (D) Formation of the elimination tree. For example, cluster sets with elimination nodes of E and M_i are connected. Note that among the members of cluster set M_i, E also appears first in the elimination order other than M_i itself. Clusters $\{F_s\}$ and $\{W, F_s\}$ are also merged to form the boundary cluster, $\{B, W, F_s\}$.

are used to allocate potentials, perform the initialization of the CG regressions of the tree, enter and propagate evidence into the tree through a series of *exchange* and *push* operations, and finally to evaluate the posterior marginals. Briefly, the exchange operation is an application of Bayes' theorem, which enables the representation of two CG regressions as two distributions, such that the joint distribution does not change. The push operation follows Lauritzen and Jensen (2001) and operates by pushing evidence absorbed into a continuous variable up the tree to the boundary cluster using a series of exchange operations. In **BayesNetBP**, tracking the push and exchange operations utilizes two different list structures for storing and organizing CG regressions as introduced in Cowell (2005).

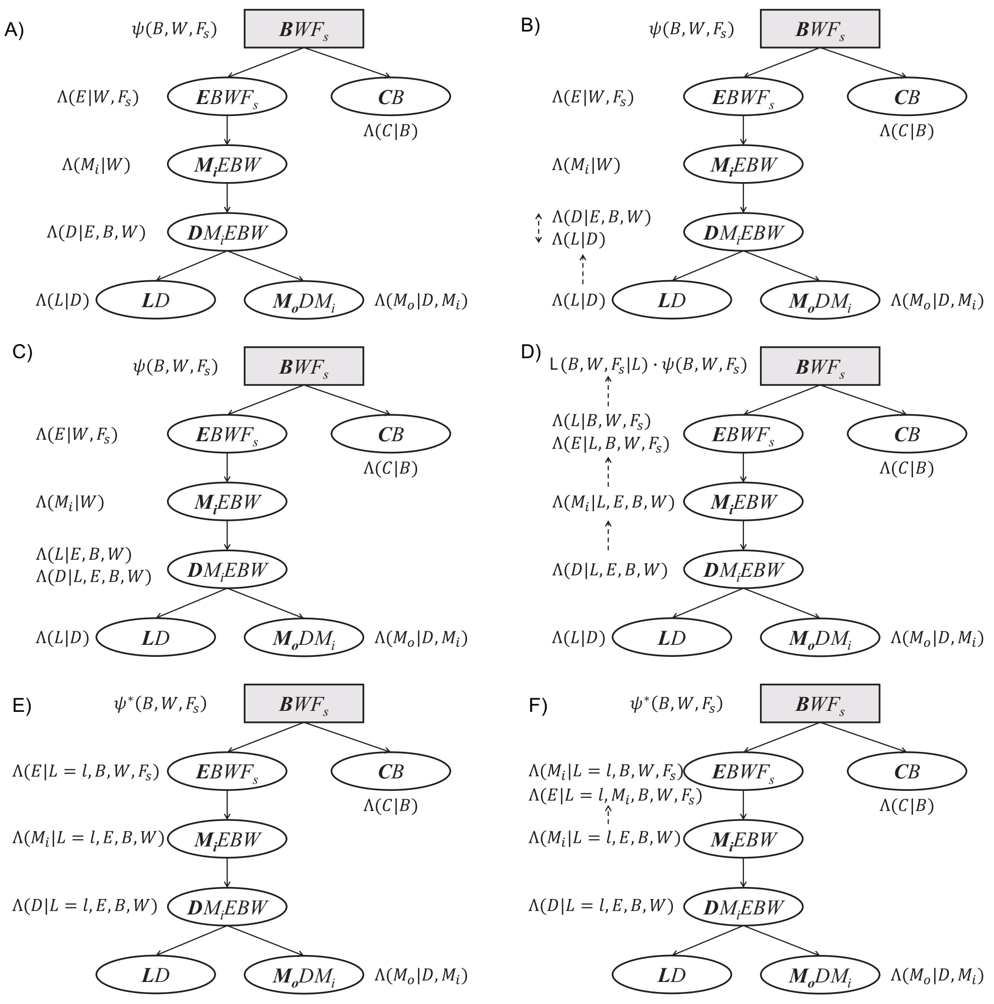


Figure 4: The belief propagation process after observing $L = l$. (A) The assignment of potentials is made, and (B–C) the potential for L becomes conditionally independent on the other continuous variables as it gets pushed up the tree. (D) At the boundary cluster, $\{B, W, F\}$, the potential of L depends only on discrete variables. (E) The potentials of discrete clusters are updated. (F) An example of how the marginal of M_i is obtained.

An overview of the process is described using the emissions example (Figure 4) and we refer the reader to Cowell (2005) for the comprehensive description of the four algorithms that facilitate belief propagation. During initialization, the CG regression of a continuous variable can always be assigned to the cluster where it is the elimination node for the cluster, while guaranteeing the potentials of its tails are all assigned upward of this cluster in the tree. The head and tail(s) of a CG regression are defined in the same way as in Lauritzen and Jensen (2001) with only one head variable. Specifically, a CG regression is defined as $\Lambda(\mathit{Head} \mid \mathit{Tail}_1, \mathit{Tail}_2, \ldots, A, B, \ldots)$, where the tails are the continuous nodes that the head variable is conditional on, while $A, B, \ldots$ are additional discrete variables. In our example, a possible assignment of CG regressions is shown in Figure 4A. The regression of L is assigned to the cluster $\{L, D\}$, whose elimination node is L.

To absorb evidence $L = l$ into the network, the potential of L will be pushed toward the boundary cluster, $\{B, W, F_s\}$. As the first step, the potential of L is pushed to the cluster set $\{D, M_i, E, B, W\}$ (Figure 4B), where the exchange operation is performed on the potentials $\Lambda(L \mid D)$ and $\Lambda(D \mid E, B, W)$. The exchange operation removes D from the tail of $\Lambda(L \mid D)$ using Bayes' theorem (Figure 4C). Through a series of push and exchange operations, the potential of L arrives at the boundary cluster and all continuous variables are removed from its tail (Figure 4D).

Interfacing the compartments

After this pushing completes, a *message* in the form of likelihood, will be passed from the continuous compartment to the discrete compartment. This message will then be propagated within that compartment using a standard sum-product algorithm (Koller and Friedman 2009). Interfacing the compartments is straightforward due to the fact that after the push operations are completed, the potential of observed variable only depends on a set of discrete variables. As described earlier, these discrete variables are guaranteed to be members of the involved boundary cluster (Figure 4D). Since the likelihood function is given as:

$$\begin{aligned} \mathcal{L}(B, W, F_s \mid L = l) &= f_{L|B,W,F}(l \mid B, W, F_s), \text{ and} \\ P(B, W, F_s \mid L = l) &\propto P(B, W, F_s) \cdot \mathcal{L}(B, W, F_s \mid L = l), \end{aligned}$$

the potential $\psi(B, W, F_s)$ can be updated as $\psi^*(B, W, F_s) = \psi(B, W, F_s)\mathcal{L}(B, W, F_s \mid L = l)$ (Figure 4E). When there are multiple discrete clusters, the joint distribution of all clusters will be obtained by the sum-product algorithm (Koller and Friedman 2009), with the evidence $L = l$ absorbed. Since all of the boundary clusters' updated joint distributions will then be available, they can be used to further update the marginals of the continuous variables after their potentials have been pushed to them.

Querying the marginal distribution of a continuous variable

After the propagation among discrete clusters completes, the marginal distributions of the continuous variables will be further updated through refreshed discrete potentials. The marginal distribution of a continuous variable is also obtained by pushing its potential upwards to the boundary cluster. When the pushing operations are complete, its distribution only depends on discrete factors. At this stage, the marginal can be readily obtained as a mixture of Gaussian distributions. For example, if we are interested in the marginal of M_i, its potential can be pushed from $\{M_i, E, B, W\}$ to $\{E, B, W, F_s\}$ (Figure 4F), and the marginal $f_{M_i|L}\left(m_i \mid L = l\right)$ can be computed as:

$$\sum_{B,W,F_s} f_{M_i|B,W,F_s}\left(m_i \mid L = l, B = b, W = w, F_s = f_s\right) P\left(B = b, W = w, F_s = f_s\right).$$

3.4. Network comparisons and visualizations

After the absorption and propagation of new evidence, nodes that are d-connected to a node where the evidence is entered, can be expected to exhibit changes in their distributions. Thus, assessing the node-specific changes in the network can be done systematically by examining the marginal distributions before and after evidence is absorbed and propagated (Moharil *et al.* 2016). **BayesNetBP** calculates a symmetric version of Kullback-Leibler divergence known as Jeffrey's information (JI; Jeffreys 1946) to quantify the change in the marginals for the continuous nodes. JI is a function of the parameters of the two distributions (before and after belief propagation). **BayesNetBP** utilizes a signed JI in order to visualize the magnitude and direction of effect. Alternative quantities can be used (e.g., fold change) and can be computed directly using information from the local distributions. Future releases will include additional options for quantifying distributional changes. **BayesNetBP** can also extract conditional and joint probabilities for all combinations of discrete variables.

4. Examples in statistical genetics

In this section, we present examples that are motivated by problems in statistical genetics (Benfey and Mitchell-Olds 2008; Rockman 2008). However, we emphasize that package **BayesNetBP** can be used in connection with any application. The data itself consists of a mixture of genotypes at SNP (single nucleotide polymorphism) markers, and phenotypes, which can be broadly defined as any complex trait, e.g., clinical traits or arising from array-based profiling (Jansen and Nap 2001). Over the past decade, a tremendous amount of research has been centered on the structural learning problem (Zhu *et al.* 2007, 2008; Li *et al.* 2006; Liu, de la Fuente, and Hoeschele 2008; Chaibub Neto, Ferrara, Attie, and Yandell 2008; Chaibub Neto, Keller, Attie, and Yandell 2010; Hageman, Leduc, Korstanje, Paigen, and Churchill 2011; Blair, Kliebenstein, and Churchill 2012; Chaibub Neto *et al.* 2013). Genotype-phenotype networks are often used to form hypothesis about candidate intermediates (e.g., genes and metabolites) to generate hypotheses and guide future experiments. Therefore, belief propagation offers a unique layer of information and insights from the quantification and visualization of probabilistic changes in the network.

In this application area, there are tools within the R programming language for quantitative trait loci (QTL) mapping, and network inference for structural learning, but not probabilistic reasoning (Table 1). **BayesNetBP** fills a gap by delivering accessible tools that will enable investigators to reason with genotype-phenotype networks based on their QTL mapping results. In this section, we showcase the flexibility of **BayesNetBP** for CG-BNs and discrete BNs and demonstrate interfacing with existing select packages for structural learning.

4.1. CG-BN example

Expression quantitative trait loci (eQTL; Jansen and Nap 2001) data from the livers from a MRL/MpJ × SM/J mouse intercross (Leduc *et al.* 2012) is used throughout the next examples. Each sample in the dataset has a gene expression profile, 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

Analysis task	R package
QTL mapping	**qtl** (Broman, Wu, Sen, and Churchill 2003; Broman and Sen 2009), **eqtl** (Khalili and Loudet 2012)
Network structural learning	**bnlearn**, **catnet**, **deal**, **qtlnet**, **RHugin**
Probabilistic reasoning	**BayesNetBP**, **gRain**, **RHugin**

Table 1: R packages for genotype-phenotype mapping and network analysis.

ontologies, were selected to be included in a BN (Alvord *et al.* 2007). This filtered data is included in the **BayesNetBP** package. In this section, examples are provided that demonstrate analyses and visualizations using **BayesNetBP** on a CG-BN with structure that was learned using **qtlnet** (Chaibub Neto *et al.* 2010). Within this network, we also dichotomized one of the nodes to demonstrate a CG-BN with two discrete layers.

Generating a cluster tree object

The **BayesNetBP** package can utilize a pre-specified network structure and a dataset to estimate the local distributions. The `LocalModelCompile` function can be used to compute local distributions. **qtlnet** (Chaibub Neto *et al.* 2010) is a package for structural learning of genotype-phenotype networks. The package produces a '`qtlnet`' object, which can be used input. More generally, a network of class '`graphNEL`' and a dataset can also be used as input. The BN structure and a vector indicating the node types needs to be extracted. Below, we extract the network from a '`qtlnet`' object which is contained in the file `liver.rda` that is available in the supplementary material.

```
R> library("qtlnet")
R> load("liverqtl.rda")
R> dag <- liverqtl$dag
R> liver <- liverqtl$data
R> node.class <- liverqtl$node.class
R> qtlnet.fit <- liverqtl$qtlnet.fit
```

Local distributions are computed using the `LocalModelCompile` function. The next step is to compile the cluster tree with `ClusterTreeCompile`. Since the input to `ClusterTreeCompile` is a '`graphNEL`' object, we demonstrate the conversion below for cluster tree compilation.

```
R> library("igraph")
R> library("qtl")
R> models.qtl <- LocalModelCompile(data = qtlnet.fit)
R> dag <- qtlnet_to_graphNEL(qtlnet.fit)
R> graph::nodes(dag) <- gsub("@", "_", graph::nodes(dag))
R> d.nodes <- names(models.qtl$pots)
R> c.nodes <- names(models.qtl$bags)
R> node.class <- rep(c(TRUE, FALSE), c(length(d.nodes), length(c.nodes)))
R> names(node.class) <- c(d.nodes, c.nodes)
R> cst <- ClusterTreeCompile(dag = dag, node.class = node.class)
```

This cluster tree object can be used for probabilistic reasoning using **BayesNetBP** in the R environment or imported into the **shiny** app.

Probabilistic reasoning with a CG-BN

Alternatively, we demonstrate the use of a '`graphNEL`' object to build the cluster tree. The `Initializer` function is used to generate and propagate a cluster tree. The input to this function is a DAG in the format of a '`graphNEL`' object, a data frame for the estimation of the local distributions and a vector specifying node types. In this example, the vector `node.class` indicates which nodes are discrete (`TRUE`) and continuous (`FALSE`).

```
R> data("liver", package = "BayesNetBP")
R> liver$node.class[1:5]

    HDL Pla2g4a   Nr1i3 Cyp2b10  Ppap2a
   TRUE   FALSE   FALSE    TRUE   FALSE

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

Note that the initialization of cluster tree comprises multiple steps. Instead of using the single wrapper function, `Initializer`, **BayesNetBP** also provides functions to implement these steps separately so that the user can obtain the intermediate outputs. This can be achieved using `ClusterTreeCompile` and `LocalModelCompile`, as shown in the previous example. The `ElimTreeInitialize` function initializes the cluster tree by integrating cluster tree structure with local distributions. In the following example, for illustrative purposes, the three phenotype variables (`HDL`, `Spgl1`, `Cyp2b10`) are dichotomized such that some of the discrete variables will have discrete parents, thereby resulting in two discrete clusters in the cluster tree.

```
R> data("liver", package = "BayesNetBP")
R> models <- LocalModelCompile(liver$data, liver$dag, liver$node.class)
R> cst <- ClusterTreeCompile(dag = liver$dag, node.class = liver$node.class)
R> tree.init <- ElimTreeInitialize(tree = cst$tree.graph, dag = cst$dag,
+    model = models, node.sets = cst$cluster.sets,
+    node.class = cst$node.class)
```

The cluster tree object is not ready for evidence absorption or making queries until it has been propagated. The function `Propagate` will perform the propagation within the discrete compartment so that the joint distribution tables of all clusters are computed.

```
R> tree.init@propagated

[1] FALSE

R> tree.init.p <- Propagate(tree.init)
R> tree.init.p@propagated

[1] TRUE
```

At this point the cluster tree is ready for probabilistic reasoning. Different types of evidence can be simultaneously entered for different variables in the network. The function `AbsorbEvidence` embeds hard or soft evidence about variables into the cluster tree object. For example, suppose that the continuous node `Nr1i3` is observed with value and `chr1@42.65` is set to state to 1 (hard evidence). Likelihood evidence (soft evidence) is entered for `Spgl1` (High: 0.9, Low: 0.2). After evidence absorption, this function propagates the cluster tree automatically, such that the returned object is ready for probabilistic queries or additional evidence absorption.

```
R> tree.post <- AbsorbEvidence(tree.init.p, c("Nr1i3", "chr1_42.65",
+    "Spgl1"), list(1, "1", c(High = 0.9, Low = 0.2)))
```

The marginal distributions of both continuous and discrete variables can be queried using the function `Marginals`. For continuous variables, the marginal is a mixture of Gaussian distributions. The output is a data frame with three columns of sub-population probabilities, means and variances. For a discrete variable, the marginal is a vector of state probabilities. The function `SummaryMarginals` computes means and standard deviations for continuous variables.

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

     High       Low 
0.2524564 0.7475436 

R> SummaryMarginals(marg)

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

R> 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
```

`FactorQuery` is a function that can provide the joint distribution of any combination of discrete variables, as well as their conditional distributions. For example, the joint distribution of `HDL` and `Cyp2b10`, and the conditional distribution of `HDL`, can be computed.

```
R> 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

R> FactorQuery(tree.post, "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
```

Observations can also be generated from a joint distribution of the BN using the `Sampler` function. Sampling results can also be used to approximately estimate the covariance and joint distributions of continuous variables.

```
R> set.seed(12345)
R> x <- Sampler(tree.post, n = 100)
R> head(x)

  Spgl1  HDL chr1_84.93 chr1_86.65 chr12_30.87 Cyp2b10 chr1_71.35       Neu1
1  High High          1          1           1    High          2  0.4771764
2  High High          1          1           2     Low          1 -1.9987354
3  High High          1          2           1    High          2 -0.1088504
4  High High          1          2           2    High          1 -1.6582488
5  High High          1          2           3    High          1  0.3342812
6  High High          1          3           2    High          1 -2.6548826
        Degs1     Ppap2a      Apoa2         Kdsr    Pla2g4a
1 -2.22942643 -0.8508257 -0.3710051  2.236634489  1.4462282
2 -0.35203008  0.7104870  0.2861025 -0.001581723 -0.9104425
3  0.28419039 -1.0847334  0.2551665  0.860638216  0.7459274
4  0.02938886  0.3400735  0.5378564  0.121909064  0.5980080
5 -0.26948601  2.2764720  0.1436680  0.259898526  1.8696464
6  0.65596537 -1.1265880 -0.4516221 -0.070646689  0.6319818

R> cov(x[c("Ppap2a", "Neu1", "Degs1")])

          Ppap2a      Neu1     Degs1
Ppap2a 1.2376771 0.3886075 0.3739735
Neu1   0.3886075 0.5748370 0.1246443
Degs1  0.3739735 0.1246443 1.0266060
```

BayesNetBP can be used to visualize marginal distributions of both discrete and continuous nodes with the `PlotMarginals` function. Marginals of continuous and discrete nodes are shown as density plots (Figure 5A) and bar plots (Figure 5B–C), respectively.

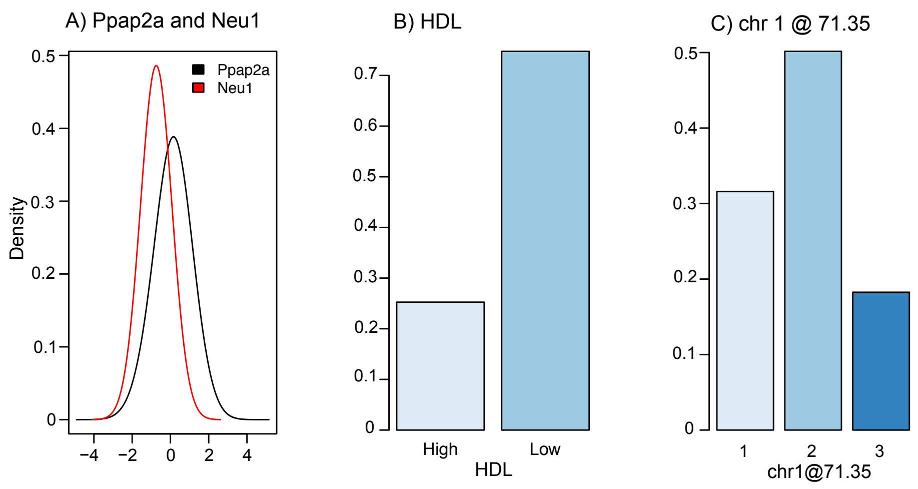


Figure 5: An example visualization of marginals for (A) continuous nodes (`Ppap2a` and `Neu1`) and discrete nodes (B) `HDL` and (C) `chr@71.35`.

```
R> PlotMarginals(marg)
```

`PlotCGBN` provides a visualization that quantifies changes in marginal distributions, as described in Section 4.3. The inputs are two cluster tree objects that are used to calculate, and return as output, the signed symmetric KL divergence between marginals for each node in the BN. `PlotCGBN` outputs the BN with nodes colored accordingly. An example is shown in Figure 6, where `Nr1i3` and `chr1@42.65` have absorbed and propagated hard evidence (orange), and the changes of the marginals for the other nodes are quantified. Color bars are used to capture the range of these changes for the continuous nodes (Figure 6), where red indicates an increase in mean (activation), and blue a decrease in mean (inhibited). Since there is no directionality for changes in discrete variables, only red is used to show the differences in marginal distributions.

```
R> library("Rgraphviz")
R> PlotCGBN(tree.init.p, tree.post)

          HDL         Spgl1       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 -0.6530478228 -0.0485877305  0.0075482375
        Apoa2          Kdsr       Pla2g4a
-0.0016797245  0.0023407076  0.0129152918
```

In the following example, `PlotCGBN` is utilized to demonstrate how the patterns of *d*-separation in the BN change depending on where evidence has been observed. `Spgl1` and `chr1@84.93` are not *d*-separated. Thus, after observing `Spgl1` (Figure 7A), the marginal of `chr1@84.93` will change (signed symmetric KL divergence is 0.0047). However, if `Cyp2b10` is also observed (gray), then `Spgl1` and `chr1@84.93` become *d*-separated, and further absorption of evidence on `Spgl1` will not change the distribution of `Cypb10` (symmetric KL divergence is 0; Figure 7B). Theses conditional independencies can be explored by absorbing `Cyp2b10` into the

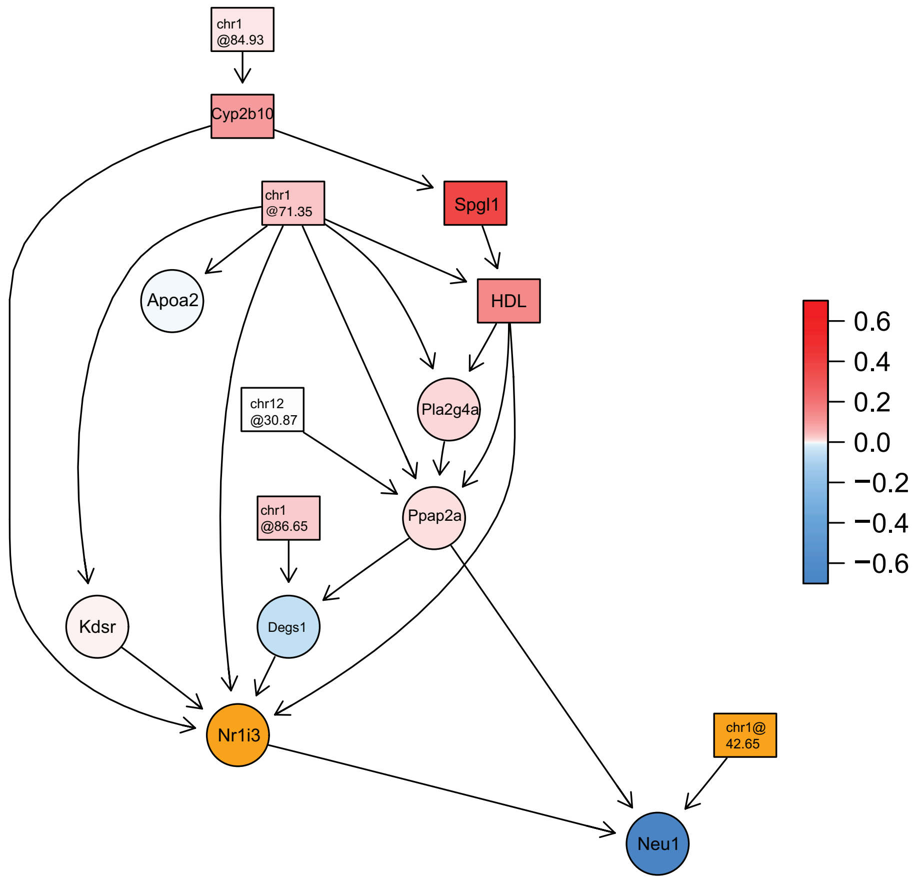


Figure 6: The signed symmetric KL divergence of *d*-connected nodes after the absorption and propagation of evidence for `Nr1i3` and `chr@42.65` (orange). Darker shades represent a larger shift in marginal distributions. For continuous variables, red indicates an increase in mean and blue indicates a decrease.

original cluster tree object to obtain `tree.1`, then absorbing `Spgl1` on top of this to obtain `tree.2`, and finally by comparing `tree.1` and `tree.2` using `PlotCGBN`.

```
R> tree.1 <- AbsorbEvidence(tree.init.p, c("Cyp2b10"), list("High"))
R> tree.2 <- AbsorbEvidence(tree.1, c("Spgl1"), list("High"))
R> tree.3 <- AbsorbEvidence(tree.init.p, c("Spgl1"), list("High"))
R> PlotCGBN(tree.init.p, tree.3)
```

```
         HDL      Cyp2b10  chr12_30.87   chr1_86.65   chr1_84.93   chr1_71.35
4.813077e-02 7.428658e-02 0.000000e+00 0.000000e+00 4.669436e-03 1.337859e-17
  chr1_42.65         Neu1        Nr1i3        Degs1       Ppap2a        Apoa2
4.054185e-17 1.127977e-03 1.242260e-02 1.639993e-04 4.198759e-03 0.000000e+00
        Kdsr      Pla2g4a
0.000000e+00 1.599930e-03
```

```
R> PlotCGBN(tree.1, tree.2)
```

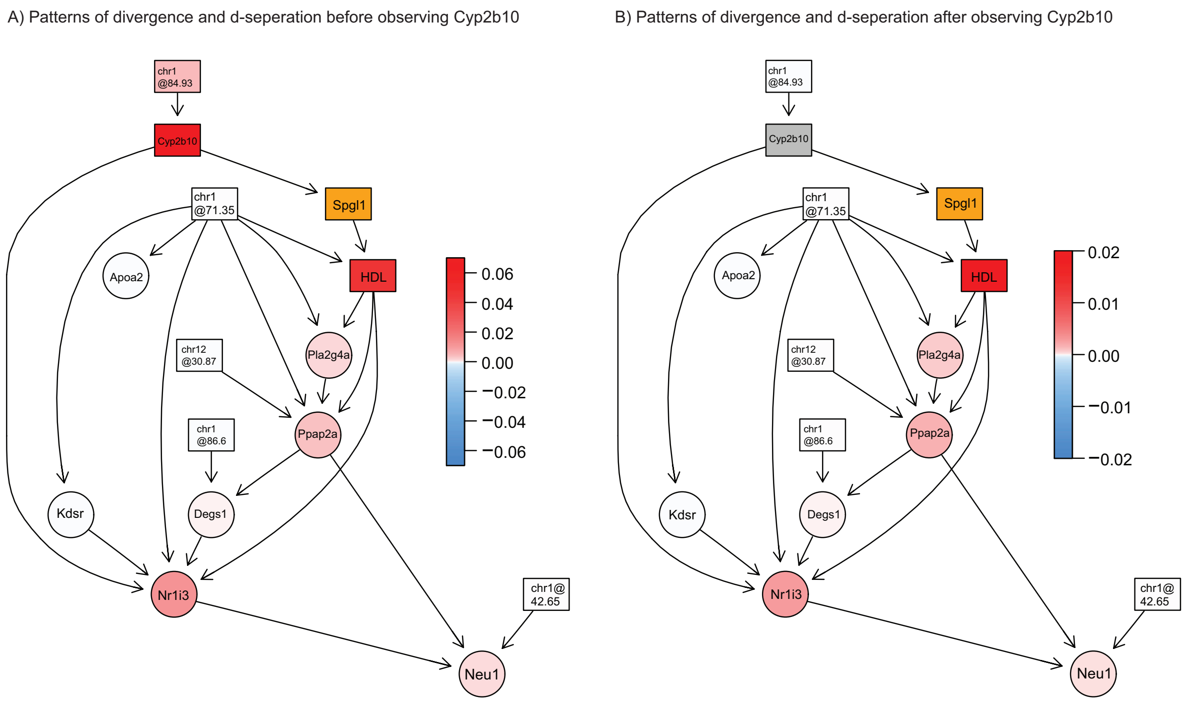


Figure 7: An example of how conditional dependencies change when new information is entered into the network. (A) When `Spgl1` is observed, a shift in the marginal will occur for `chr1@84.93`, because they are *d*-connected. (B) However, if `Cyp2b10` is also observed (gray), then there will not be change in `chr1@84.93`, because they have become *d*-separated.

```
         HDL  chr12_30.87   chr1_86.65   chr1_84.93   chr1_71.35   chr1_42.65
1.907479e-02 1.299197e-17 4.103860e-17 0.000000e+00 1.198082e-17 4.054185e-17
        Neu1        Nr1i3        Degs1       Ppap2a        Apoa2         Kdsr
2.475948e-04 2.619255e-03 6.421556e-05 1.672824e-03 0.000000e+00 0.000000e+00
     Pla2g4a
6.225181e-04
```

BayesNetBP also provides a systematic assessment of changes in marginals over a range of evidence. For example, evidence for node `Nr1i3` can be set to a range between −3 and 3. Evidence in this range is then absorbed and propagated in the BN, and `ComputeKLDs` can then be used to calculate the signed symmetric KL divergence for all *d*-connected nodes, which returns the results as a data frame for inspection of visualization (Figure 8).

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

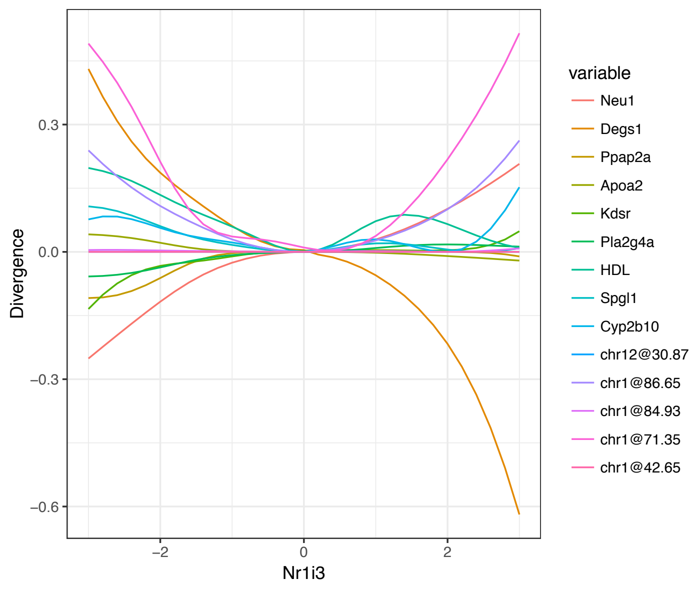


Figure 8: Profiles of signed and signed symmetric KL-divergence for other nodes as `Nr1i3` is observed at values ranging from −3 to 3.

4.2. Discrete BN example

In this example, we consider a subset of gene expression data from 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 log2(sample/reference) for 6,216 genes and was accessed from gene expression omnibus (GSE1990; Barrett *et al.* 2013). For network analysis, a subset of genes was identified after filtering, linkage analysis and regression modeling. Briefly, 901 expression values mapped to the YeastNet database (Kim *et al.* 2013). Linkage analysis was performed on these traits using the R package **qtl**. A set of 369 genes that had a significant QTL were used as predictors in an elastic net regression model (Zou and Hastie 2005) with COX10 as the response variable. The optimal shrinkage parameter was estimated as 0.086 using 10-fold cross validation. The resulting model shrunk all but 37 regression coefficients to zero. The corresponding 37 genes, COX10 and 12 SNP markers from significant QTL were included as variables for the BN.

For this example, gene expression values were dichotomized at the median. Thus, the data consists of discretized gene expression variables (phenotypes) and SNP variables that indicate with states for the parental strain of origin (genotypes; Brem and Kruglyak 2005). To demonstrate compatibility with existing packages, the structure of the BN is learned using a hill-climbing algorithm in the **bnlearn** package. The modeling assumptions that require genotypes to be upstream of phenotype can be directly encoded using the `blacklist` option.

```
R> library("bnlearn")
R> library("igraph")
R> data("yeast", package = "BayesNetBP")
R> node.names <- names(yeast)
R> geno <- node.names[1:12]
R> pheno <- node.names[13:50]
```

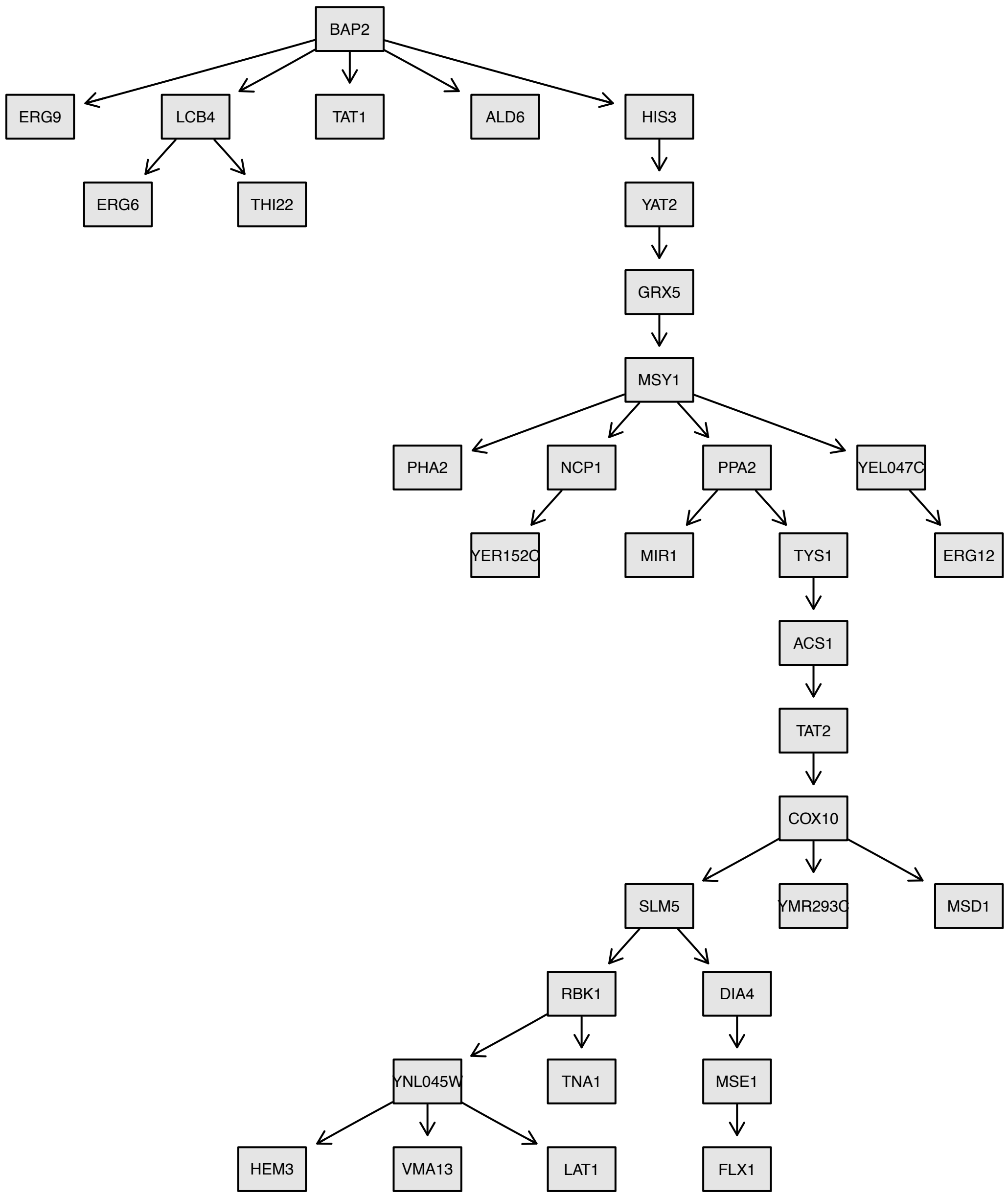


Figure 9: The structure of semi-elimination tree for the `yeast` network.

```
R> bl <- rbind(expand.grid(geno, geno), expand.grid(pheno, geno))
R> names(bl) <- c("from", "to")
R> dag.bn <- bnlearn::hc(yeast, blacklist = bl)
R> dag.graphNEL <- bn_to_graphNEL(dag.bn)
```

After the network structure is learned, the cluster tree can be built using the same procedure that was used for a CG-BN. Figure 9 shows the constructed semi-elimination tree, which is plotted using the `PlotTree` function.

```
R> node.names <- graph::nodes(dag.graphNEL)
```

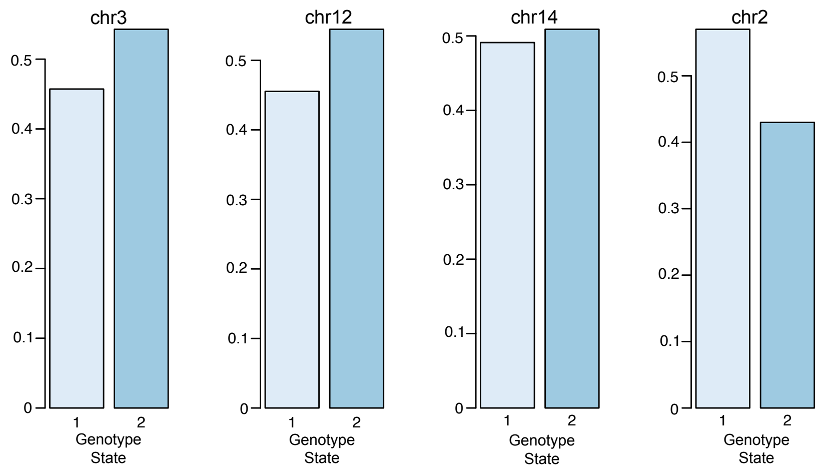


Figure 10: Select marginal distributions after evidence is entered and propagated.

```
R> node.class <- rep(TRUE, length(node.names))
R> names(node.class) <- node.names
R> tree.init.p <- Initializer(dag = dag.graphNEL, data = yeast,
+    node.class = node.class, propagate = TRUE)
R> 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 also be visualized using `PlotMarginals` (Figure 10).

```
R> tree.post <- AbsorbEvidence(tree.init.p, c("MSY1", "Qchr4"),
+    list("1", "2"))
R> marg.yeast <- Marginals(tree.post, node.names[2:5])
R> PlotMarginals(marg.yeast)
```

`PlotCGBN` provides a comparison of the marginals after evidence is entered and propagated through the cluster tree object (Figure 11).

```
R> div <- PlotCGBN(tree.init.p, tree.post)
```

Queries on joint and conditional distributions of factors can be output in tabular form.

```
R> FactorQuery(tree.post, node.names[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
```

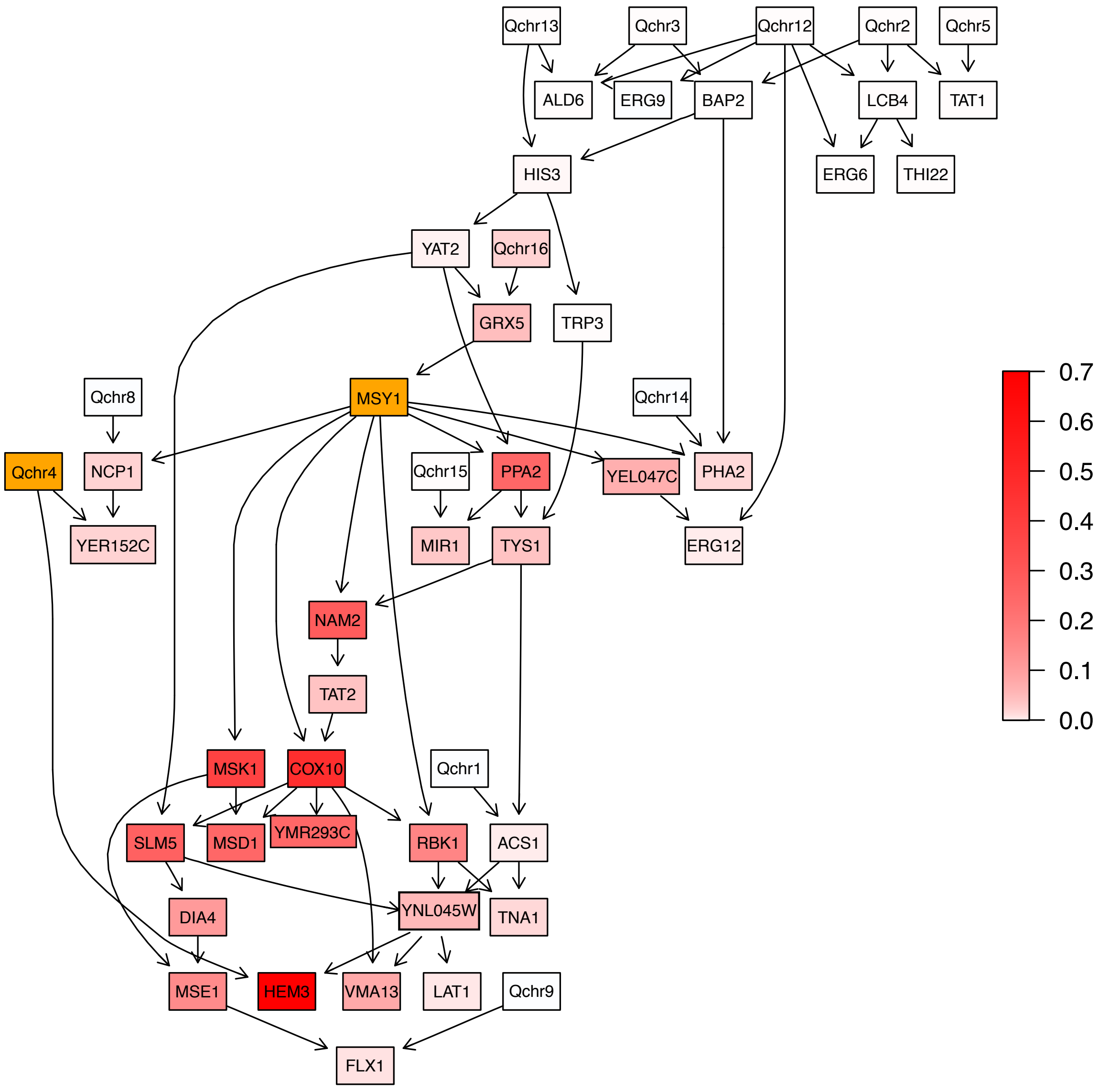
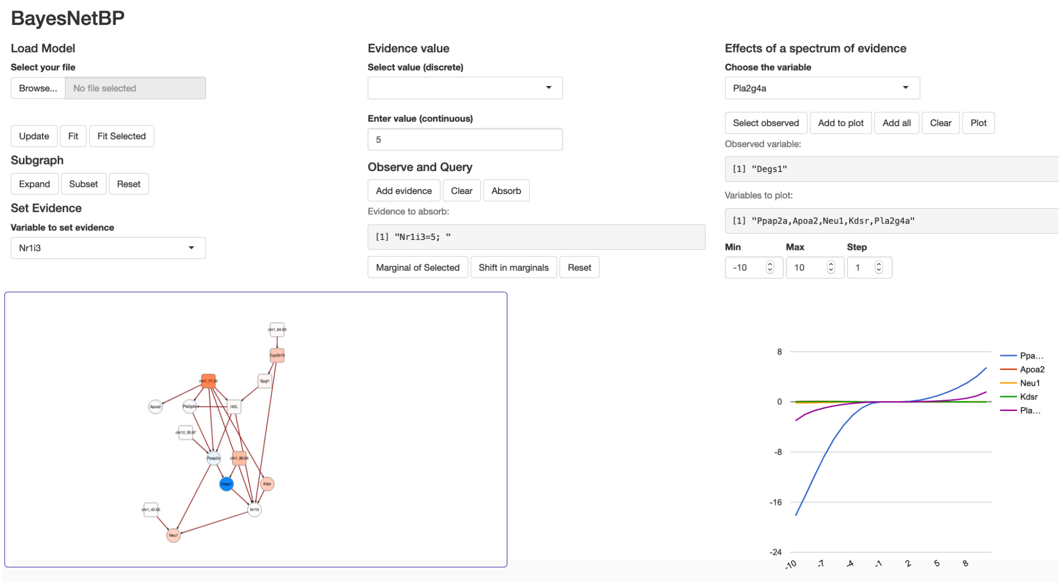


Figure 11: Visualization of the node-specific shift after evidences on `Qchr4` and `MSY1` were absorbed.

```
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
```

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


BayesNetBP

Load Model

Select your file: No file selected

Subgraph

Set Evidence

Variable to set evidence: Nr1i3

Evidence value

Select value (discrete):

Enter value (continuous): 5

Observe and Query

Evidence to absorb:

```
[1] "Nr1i3=5; "
```

Effects of a spectrum of evidence

Choose the variable: Pla2g4a

Observed variable:

```
[1] "Degs1"
```

Variables to plot:

```
[1] "Ppap2a,Apoa2,Neu1,Kdsr,Pla2g4a"
```

Min: -10 Max: 10 Step: 1

Figure 12: (A) A screenshot of the **shiny** console. The console allows the user to enter evidence, propagate and make queries. Users can also visualize changes in marginals over spectrums of evidence. (B) The interactive graphics, supported by **Cytoscape**, allow the investigator to explore the network, and (C) subset to regions.

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

4.3. shiny app

The **shiny** app console of package **BayesNetBP** consists of three panels (Figure 12). The first panel controls the loading of a '`clustertree`' object and network layouts. It also enables the user to subset the network in order to improve and customize visualization. The `Expand` function generates subgraphs around specified nodes. The second panel is used for the absorption of hard and soft evidence. The users can add multiple pieces of evidence to a list and absorb them into the cluster tree object simultaneously. The nodes that have evidence absorbed are colored orange. Marginals of the nodes can be queried and displayed as density or bar plots by node type. If a set of evidence has been absorbed, the marginals before and after absorption will be returned to facilitate comparison. The *Shift in Marginals* function computes the signed symmetric divergence for all *d*-connected nodes and colors them in the same manner as the function `PlotCGBN`. The function for systematic assessment of changes in node marginals is provided in the third panel. Users can specify the node for evidence absorption and the nodes whose signed symmetric KL divergence is to be calculated. The result is visualized in an interactive plot. A video introduction of the **shiny** app can be found at https://youtu.be/4qLt__E6h2c.

5. Conclusions

The **BayesNetBP** package was developed in the R programming language for probabilistic reasoning in BNs that are discrete, continuous or mixed (CG-BNs). To the authors' knowledge, this is the first non-commercial package in R that supports exact reasoning of this type. A limitation of the package is that intense computation may be required when the number of possible factor configurations of the discrete nodes is large. Moreover, the queries of joint distributions for discrete nodes that are members of a given cluster are more efficient compared to out of cluster queries, which require an additional run of message passing through a sub-tree. A list of efficient queries with respect to a node can be obtained through function `FactorQuery` by setting `mode` to `list`. Future developments will focus on improving the computational efficiency, implementation of additional measures of divergence and the extension of the package to perform loopy belief propagation in undirected graphs.

Acknowledgments

HY, JM and RHB were supported through the NSF DMS 1312250. HY and RHB were also supported through the NSF DMS 1557593. HY was also supported by NCI P30CA016056 and U24CA232979.

References

Alvord G, Roayaei J, Stephens R, Baseler MW, Lane HC, Lempicki RA (2007). "The DAVID Gene Functional Classification Tool: A Novel Biological Module-Centric Algorithm to Functionally Analyze Large Gene Lists." *Genome Biology*, **8**(9), 183. `doi:10.1186/gb-2007-8-9-r183`.

Balov N, Salzman P (2020). ***catnet**: Categorical Bayesian Network Inference.* R package version 1.15.7, URL `https://CRAN.R-project.org/package=catnet`.

Barrett T, Wilhite SE, Ledoux P, Evangelista C, Kim IF, Tomashevsky M, Marshall KA, Phillippy KH, Sherman PM, Holko M, Yefanov A, Lee H, Zhang N, Robertson C, Serova N, Davis S, Soboleva A (2013). "NCBI GEO: Archive for Functional Genomics Data Sets Update." *Nucleic Acids Research*, **41**(D1), D991–D995. `doi:10.1093/nar/gks1193`.

Benfey PN, Mitchell-Olds T (2008). "From Genotype to Phenotype: Systems Biology Meets Natural Variation." *Science*, **320**(5875), 495–497. `doi:10.1126/science.1153716`.

Blair RH, Kliebenstein DJ, Churchill GA (2012). "What Can Causal Networks Tell Us about Metabolic Pathways?" *PLoS Computational Biology*, **8**(4), e1002458. `doi:10.1371/journal.pcbi.1002458`.

Boettcher SG, Dethlefsen C (2003). "**deal**: A Package for Learning Bayesian Networks." *Journal of Statistical Software*, **8**(20), 1–40. `doi:10.18637/jss.v008.i20`.

Brem RB, Kruglyak L (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), 1572–1577. `doi:10.1073/pnas.0408709102`.

Broman KW, Sen S (2009). *A Guide to QTL Mapping with R/**qtl***. Springer-Verlag.

Broman KW, Wu H, Sen Ś, Churchill GA (2003). "R/**qtl**: QTL Mapping in Experimental Crosses." *Bioinformatics*, **19**(7), 889–890. doi:10.1093/bioinformatics/btg112.

Castillo E, Menéndez JM, Sánchez-Cambronero S (2008). "Predicting Traffic Flow Using Bayesian Networks." *Transportation Research Part B: Methodological*, **42**(5), 482–509. doi:10.1016/j.trb.2007.10.003.

Chaibub Neto E, Broman AT, Keller MP, Attie AD, Zhang B, Zhu J, Yandell BS (2013). "Modeling Causality for Pairs of Phenotypes in System Genetics." *Genetics*, **193**(3), 1003–1013. doi:10.1534/genetics.112.147124.

Chaibub Neto E, Ferrara CT, Attie AD, Yandell BS (2008). "Inferring Causal Phenotype Networks From Segregating Populations." *Genetics*, **179**(2), 1089–1100. doi:10.1534/genetics.107.085167.

Chaibub Neto E, Keller MP, Attie AD, Yandell BS (2010). "Causal Graphical Models in Systems Genetics: a Unified Framework for Joint Inference of Causal Network and Genetic Architecture for Correlated Phenotypes." *The Annals of Applied Statistics*, **4**(1), 320–339. doi:10.1214/09-aoas288.

Chang W, Cheng J, Allaire J, Xie Y, McPherson J (2020). ***shiny**: Web Application Framework for R*. R package version 1.4.0.2, URL https://CRAN.R-project.org/package=shiny.

Chickering DM, Heckerman D, Meek C (2004). "Large-Sample Learning of Bayesian Networks in NP-Hard." *Journal of Machine Learning Research*, **5**, 1287–1330.

Cowell RG (2005). "Local Propagation in Conditional Gaussian Bayesian Networks." *Journal of Machine Learning Research*, **6**, 1517–1550.

Cox DR, Wermuth N (1993). "Linear Dependencies Represented by Chain Graphs." *Statistical Science*, **8**(3), 204–218. doi:10.1214/ss/1177010887.

Geiger D, Heckerman D (1994). "Learning Gaussian Networks." In *Proceedings of the Tenth International Conference on Uncertainty in Artificial Intelligence*, pp. 235–243. Morgan Kaufmann Publishers Inc.

Gosling J, Joy B, Steele G, Bracha G (2000). *The Java Language Specification*. Addison-Wesley Professional.

Hageman RS, Leduc MS, Korstanje R, Paigen B, Churchill GA (2011). "A Bayesian Framework for Inference of the Genotype–Phenotype Map for Segregating Populations." *Genetics*, **187**(4), 1163–1170. doi:10.1534/genetics.110.123273.

Heckerman D, Geiger D, Chickering DM (1995). "Learning Bayesian Networks: The Combination of Knowledge and Statistical Data." *Machine Learning*, **20**(3), 197–243. doi:10.1007/bf00994016.

Højsgaard S (2012). "Graphical Independence Networks with the **gRain** Package for R." *Journal of Statistical Software*, **46**(10), 1–26. doi:10.18637/jss.v046.i10.

Højsgaard S, Edwards D, Lauritzen S (2012). *Graphical Models with R*. Springer-Verlag.

Hugin Expert A/S (2017). ***Hugin** Software*. URL http://www.hugin.com/.

Jansen RC, Nap J (2001). "Genetical Genomics: The Added Value from Segregation." *Trends in Genetics*, **17**(7), 388–391. doi:10.1016/s0168-9525(01)02310-1.

Jeffreys H (1946). "An Invariant Form for the Prior Probability in Estimation Problems." *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, **186**(1007), 453–461. doi:10.1098/rspa.1946.0056.

Kahn Jr CE, Roberts LM, Shaffer KA, Haddawy P (1997). "Construction of a Bayesian Network for Mammographic Diagnosis of Breast Cancer." *Computers in Biology and Medicine*, **27**(1), 19–29. doi:10.1016/s0010-4825(96)00039-x.

Kalisch M, Mächler M, Colombo D, Maathuis MH, Bühlmann P (2012). "Causal Inference Using Graphical Models with the R Package **pcalg**." *Journal of Statistical Software*, **47**(11), 1–26. doi:10.18637/jss.v047.i11.

Khalili AA, Loudet O (2012). ***eqtl**: Tools for Analyzing eQTL Experiments: A Complementary to Karl Broman's **qtl** Package for Genome-Wide Analysis*. R package version 1.1-7, URL https://CRAN.R-project.org/package=eqtl.

Kim H, Shin J, Kim E, Kim H, Hwang S, Shim JE, Lee I (2013). "YeastNet V3: A Public Database of Data-Specific and Integrated Functional Gene Networks for *Saccharomyces Cerevisiae*." *Nucleic Acids Research*, **42**(D1), D731–D736. doi:10.1093/nar/gkt981.

Koller D, Friedman N (2009). *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press.

Konis K (2017). ***RHugin** Package*. URL http://rhugin.R-Forge.R-project.org/.

Lauritzen SL (1992). "Propagation of Probabilities, Means, and Variances in Mixed Graphical Association Models." *Journal of the American Statistical Association*, **87**(420), 1098–1108. doi:10.1080/01621459.1992.10476265.

Lauritzen SL (1996). *Graphical Models*. Oxford University Press.

Lauritzen SL, Jensen F (2001). "Stable Local Computation with Conditional Gaussian Distributions." *Statistics and Computing*, **11**(2), 191–203. doi:10.1023/a:1008935617754.

Leduc MS, Blair RH, Verdugo RA, Tsaih SW, Walsh K, Churchill GA, Paigen B (2012). "Using Bioinformatics and Systems Genetics to Dissect HDL Cholesterol Levels in an MRL/MpJ $\times$ SM/J Intercross." *Journal of Lipid Research*, **53**, M025833. doi:10.1194/jlr.m025833.

Li R, Tsaih SW, Shockley K, Stylianou IM, Wergedal J, Paigen B, Churchill GA (2006). "Structural Model Analysis of Multiple Quantitative Traits." *PLoS Genetics*, **2**(7), e114. doi:10.1371/journal.pgen.0020114.

Liu B, de la Fuente A, Hoeschele I (2008). "Gene Network Inference via Structural Equation Modeling in Genetical Genomics Experiments." *Genetics*, **178**(3), 1763–1776. doi:10.1534/genetics.107.080069.

Masegosa AR, Martínez AM, Ramos-López D, Cabañas R, Salmerón A, Langseth H, Nielsen TD, Madsen AL (2019). "**AMIDST**: A Java Toolbox for Scalable Probabilistic Machine Learning." *Knowledge-Based Systems*, **163**, 595–597. doi:10.1016/j.knosys.2018.09.019.

Moharil J, May P, Gaile DP, Blair RH (2016). "Belief Propagation in Genotype-Phenotype Networks." *Statistical Applications in Genetics and Molecular Biology*, **15**(1), 39–53. doi:10.1515/sagmb-2015-0058.

Mukherjee S, Speed TP (2008). "Network Inference Using Informative Priors." *Proceedings of the National Academy of Sciences of the United States of America*, **105**(38), 14313–14318. doi:10.1073/pnas.0802272105.

Pearl J (2014). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.

Pradhan M, Provan G, Middleton B, Henrion M (1994). "Knowledge Engineering for Large Belief Networks." In R Lopez de Mantaras, D Poole (eds.), *Uncertainty Proceedings 1994*, pp. 484–490. Morgan Kaufmann, San Francisco. doi:10.1016/B978-1-55860-332-5.50066-3.

R Core Team (2020). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Rockman MV (2008). "Reverse Engineering the Genotype-Phenotype Map with Natural Genetic Variation." *Nature*, **456**(7223), 738–744. doi:10.1038/nature07633.

Salmeron A, Ramos-Lopez D, Langseth H, Nielsen TD, Madsen AL, Masegosa A, Martinez AM, Cabanas R (2016). ***ramidst**: An Interface to the **AMIDST** Toolbox for Data Stream Processing*. R package version 0.1.0, URL https://CRAN.R-project.org/src/contrib/Archive/ramidst/.

Scutari M (2010). "Learning Bayesian Networks with the **bnlearn** R Package." *Journal of Statistical Software*, **35**(3), 1–22. doi:10.18637/jss.v035.i03.

Scutari M (2017). "Bayesian Network Constraint-Based Structure Learning Algorithms: Parallel and Optimized Implementations in the **bnlearn** R Package." *Journal of Statistical Software*, **77**(2), 1–20. doi:10.18637/jss.v077.i02.

Scutari M, Denis JB (2014). *Bayesian Networks: With Examples in R*. CRC Press. doi:10.1201/b17065.

Sesen MB, Nicholson AE, Banares-Alcantara R, Kadir T, Brady M (2013). "Bayesian Networks for Clinical Decision Support in Lung Cancer Care." *PLoS One*, **8**(12), e82349. doi:10.1371/journal.pone.0082349.

Shannon P, Markiel A, Ozier O, Baliga NS, Wang JT, Ramage D, Amin N, Schwikowski B, Ideker T (2003). "**Cytoscape**: A Software Environment for Integrated Models of Biomolecular Interaction Networks." *Genome Research*, **13**(11), 2498–2504. doi:10.1101/gr.1239303.

Spirtes P, Glymour CN, Scheines R (2000). *Causation, Prediction, and Search*, volume 81. The MIT Press. doi:10.7551/mitpress/1754.001.0001.

Sun S, Zhang C, Yu G (2006). "A Bayesian Network Approach to Traffic Flow Forecasting." *IEEE Transactions on Intelligent Transportation Systems*, **7**(1), 124–132. doi:10.1109/tits.2006.869623.

Verma TS, Pearl J (1991). "Equivalence and Synthesis of Causal Models." In *Proceedings of the Sixth Annual Conference on Uncertainty in Artificial Intelligence*, volume 6, pp. 255–270.

Werhli AV, Husmeier D (2007). "Reconstructing Gene Regulatory Networks with Bayesian Networks by Combining Expression Data with Multiple Sources of Prior Knowledge." *Statistical Applications in Genetics and Molecular Biology*, **6**(1), 15. doi:10.2202/1544-6115.1282.

Yu H (2020). ***BayesNetBP**: Bayesian Network Belief Propagation.* R package version 1.5.5, URL https://CRAN.R-project.org/package=BayesNetBP.

Yu J, Smith VA, Wang PP, Hartemink AJ, Jarvis ED (2004). "Advances to Bayesian Network Inference for Generating Causal Networks from Observational Biological Data." *Bioinformatics*, **20**(18), 3594–3603. doi:10.1093/bioinformatics/bth448.

Zhu J, Wiener MC, Zhang C, Fridman A, Minch E, Lum PY, Sachs JR, Schadt EE (2007). "Increasing the Power to Detect Causal Associations By Combining Genotypic and Expression Data in Segregating Populations." *PLoS Computational Biology*, **3**(4), e69. doi:10.1371/journal.pcbi.0030069.

Zhu J, Zhang B, Smith EN, Drees B, Brem RB, Kruglyak L, Bumgarner RE, Schadt EE (2008). "Integrating Large-Scale Functional Genomic Data to Dissect the Complexity of Yeast Regulatory Networks." *Nature Genetics*, **40**(7), 854–861. doi:10.1038/ng.167.

Zou H, Hastie T (2005). "Regularization and Variable Selection via the Elastic Net." *Journal of the Royal Statistical Society B*, **67**(2), 301–320. doi:10.1111/j.1467-9868.2005.00503.x.

Affiliation:

Rachael Hageman Blair
Department of Biostatistics
3435 Main Street
709 Kimball Tower
Buffalo, NY 14214, United States of America
E-mail: hageman@buffalo.edu

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/

June 2020, Volume 94, Issue 3 *Submitted:* 2017-05-12
doi:10.18637/jss.v094.i03 *Accepted:* 2019-02-11