Proceedings of the 3rd International Workshop on Distributed Statistical Computing (DSC 2003)
March 20–22, Vienna, Austria *ISSN 1609-395X*
Kurt Hornik, Friedrich Leisch & Achim Zeileis (eds.)
http://www.ci.tuwien.ac.at/Conferences/DSC-2003/

Learning Bayesian Networks with R

Susanne G. Bøttcher **Claus Dethlefsen**

Abstract

`deal` is a software package freely available for use with R. It includes several methods for analysing data using Bayesian networks with variables of discrete and/or continuous types but restricted to conditionally Gaussian networks. Construction of priors for network parameters is supported and their parameters can be learned from data using conjugate updating. The network score is used as a metric to learn the structure of the network and forms the basis of a heuristic search strategy. `deal` has an interface to Hugin.

1 Introduction

A Bayesian network is a graphical model that encodes the joint probability distribution for a set of random variables. Bayesian networks are treated in *e.g.* Cowell, Dawid, Lauritzen, and Spiegelhalter (1999) and have found application within many fields, see Lauritzen (2003) for a recent overview.

Here we consider Bayesian networks with mixed variables, *i.e.* the random variables in a network can be of both discrete and continuous types. A method for learning the parameters and structure of such Bayesian networks has recently been described by Bøttcher (2001). We have developed a package called `deal`, written in R (Ihaka and Gentleman, 1996), which provides these methods for learning Bayesian networks. In particular, the package includes procedures for defining priors, estimating parameters, calculating network scores, performing heuristic search as well as simulating data sets with a given dependency structure. The package can be downloaded from the Comprehensive R Archive Network (CRAN) `http://cran.R-project.org/` and may be used freely for non-commercial purposes.

In Section 2 we define Bayesian networks for mixed variables. To learn a Bayesian network, the user needs to supply a training data set and represent any prior knowledge available as a Bayesian network. Section 3 discusses how to specify a Bayesian network in terms of a directed acyclic graph and the local probability distributions. `deal` uses the prior Bayesian network to deduce prior distributions

for all parameters in the model. Then, this is combined with the training data to yield posterior distributions of the parameters. The parameter learning procedure is treated in Section 4. Section 5 describes how to learn the structure of the network. A network score is calculated and a search strategy is employed to find the network with the highest score. This network gives the best representation of data and we call it the *posterior network*. Section 6 describes how to transfer the posterior network to Hugin (`http://www.hugin.com`). The Hugin graphical user interface (GUI) can then be used for further inference in the posterior network.

2 Bayesian networks

Let $D = (V, E)$ be a Directed Acyclic Graph (DAG), where V is a finite set of nodes and E is a finite set of directed edges (arrows) between the nodes. The DAG defines the structure of the Bayesian network. To each node $v \in V$ in the graph corresponds a random variable X_v. The set of variables associated with the graph D is then $X = (X_v)_{v \in V}$. Often, we do not distinguish between a variable X_v and the corresponding node v. To each node v with parents $\mathrm{pa}(v)$, a local probability distribution, $p(x_v|x_{\mathrm{pa}(v)})$ is attached. The set of local probability distributions for all variables in the network is $\mathcal{P}$. A Bayesian network for a set of random variables X is then the pair $(D, \mathcal{P})$.

The possible lack of directed edges in D encodes conditional independencies between the random variables X through the factorization of the joint probability distribution,

$$p(x) = \prod_{v \in V} p\big(x_v|x_{\mathrm{pa}(v)}\big).$$

Here, we allow Bayesian networks with both discrete and continuous variables, as treated in Lauritzen (1992), so the set of nodes V is given by $V = \Delta \cup \Gamma$, where Δ and Γ are the sets of discrete and continuous nodes, respectively. The set of variables X can then be denoted $X = (X_v)_{v \in V} = (I, Y) = ((I_\delta)_{\delta \in \Delta}, (Y_\gamma)_{\gamma \in \Gamma})$, where I and Y are the sets of discrete and continuous variables, respectively. For a discrete variable, δ, we let $\mathcal{I}_\delta$ denote the set of levels.

To ensure availability of exact local computation methods, we do not allow discrete variables to have continuous parents. The joint probability distribution then factorizes into a discrete part and a mixed part, so

$$p(x) = p(i, y) = \prod_{\delta \in \Delta} p\big(i_\delta|i_{\mathrm{pa}(\delta)}\big) \prod_{\gamma \in \Gamma} p\big(y_\gamma|i_{\mathrm{pa}(\gamma)}, y_{\mathrm{pa}(\gamma)}\big).$$

3 Specification of a Bayesian network

In `deal`, a Bayesian network is represented as an object of class `network`. The network object has several attributes, added or changed by methods described in the following sections. A network is generated from a dataframe (here `ksl`), where the discrete variables are specified as factors,

```
ksl.nw <- network(ksl)
```

and default it is set to the empty network (the network without any arrows). If the option `specifygraph` is set to `TRUE`, a point and click graphical interface allows the user to insert and delete arrows until the requested DAG is obtained.

The primary attribute of a network is the list of nodes, in the example: `ksl.nw$nodes`. Each entry in the list is an object of class `node` representing a node in the graph, which includes information associated with the node. Several methods for the network class operate by applying an appropriate method for one or more nodes in the list of nodes.

3.1 Specification of the probability distributions

The joint distribution of the random variables in a network in `deal` is a CG distribution.

For discrete nodes, this means that the local probability distributions are unrestricted discrete distributions. We parameterize this as

$$\theta_{i_\delta|i_{\mathrm{pa}(\delta)}} = p\big(i_\delta|i_{\mathrm{pa}(\delta)}, \theta_{\delta|i_{\mathrm{pa}(\delta)}}\big),$$

where $\theta_{\delta|i_{\mathrm{pa}(\delta)}} = (\theta_{i_\delta|i_{\mathrm{pa}(\delta)}})_{i_\delta \in \mathcal{I}_\delta}$. The parameters fulfil $\sum_{i_\delta \in \mathcal{I}_\delta} \theta_{i_\delta|i_{\mathrm{pa}(\delta)}} = 1$ and $0 \leq \theta_{i_\delta|i_{\mathrm{pa}(\delta)}} \leq 1$.

For continuous nodes, the local probability distributions are Gaussian linear regressions on the continuous parents with parameters depending on the configuration of the discrete parents. We parameterize this as

$$\theta_{\gamma|i_{\mathrm{pa}(\gamma)}} = \big(m_{\gamma|i_{\mathrm{pa}(\gamma)}}, \beta_{\gamma|i_{\mathrm{pa}(\gamma)}}, \sigma^2_{\gamma|i_{\mathrm{pa}(\gamma)}}\big),$$

so that

$$\big(Y_\gamma|i_{\mathrm{pa}(\gamma)}, y_{\mathrm{pa}(\gamma)}, \theta_{\gamma|i_{\mathrm{pa}(\gamma)}}\big) \sim \mathcal{N}\big(m_{\gamma|i_{\mathrm{pa}(\gamma)}} + y_{\mathrm{pa}(\gamma)}\beta_{\gamma|i_{\mathrm{pa}(\gamma)}}, \sigma^2_{\gamma|i_{\mathrm{pa}(\gamma)}}\big).$$

A suggestion for the local probability distributions is generated and attached to each node as the attribute `prob`. The suggestion can then be edited afterwards.

For a discrete variable δ, the suggested local probability distribution $p(i_\delta|i_{\mathrm{pa}(\delta)})$ is taken to be uniform over the levels for each parent configuration, *i.e.*

$$p(i_\delta|i_{\mathrm{pa}(\delta)}) = 1/\mathcal{I}_\delta.$$

Define $z_{\mathrm{pa}(\gamma)} = (1, y_{\mathrm{pa}(\gamma)})$ and let $\eta_{\gamma|i_{\mathrm{pa}(\gamma)}} = (m_{\gamma|i_{\mathrm{pa}(\gamma)}}, \beta_{\gamma|i_{\mathrm{pa}(\gamma)}})$, where $m_{\gamma|i_{\mathrm{pa}(\gamma)}}$ is the intercept and $\beta_{\gamma|i_{\mathrm{pa}(\gamma)}}$ is the vector of coefficients. For a continuous variable γ, the suggested local probability distribution $\mathcal{N}(z_{\mathrm{pa}(\gamma)}\eta_{\gamma|i_{\mathrm{pa}(\gamma)}}, \sigma^2_{\gamma|i_{\mathrm{pa}(\gamma)}})$ is determined as a regression on the continuous parents for each configuration of the discrete parents.

3.2 The joint distribution

We now show how the joint probability distribution of a network can be calculated from the local probability distributions.

For the discrete part of the network, the joint probability distribution is found as

$$p(i) = \prod_{\delta \in \Delta} p\big(i_\delta|i_{\mathrm{pa}(\delta)}\big).$$

For continuous variables, the joint distribution $\mathcal{N}(M_i, \Sigma_i)$ is determined for each configuration of the discrete variables by applying a sequential algorithm developed in Shachter and Kenley (1989).

In `deal`, we can assess these quantities by

```
ksl.j <- jointprior(ksl.nw)
```

and inspect the attributes `jointmu`, containing M_i, `jointsigma`, containing Σ_i, and `jointalpha`. The discrete part, $p(i)$, is not returned directly but may be deduced from `ksl.j$jointalpha` by division by `sum(ksl.j$jointalpha)`.

4 Parameter learning

To estimate the parameters in the network, we use the Bayesian approach. We encode our uncertainty about parameters θ in a prior distribution $p(\theta)$, use data d to update this distribution, and hereby obtain the posterior distribution $p(\theta|d)$ by using Bayes' theorem,

$$p(\theta|d) = \frac{p(d|\theta)p(\theta)}{p(d)}, \qquad \theta \in \Theta. \tag{1}$$

Here Θ is the parameter space, d is a random sample from the probability distribution $p(x|\theta)$ and $p(d|\theta)$ is the joint probability distribution of d, also called the likelihood of θ. We refer to this as *parameter learning* or just *learning*.

In `deal`, we assume that the parameters associated with one variable are independent of the parameters associated with the other variables and, in addition, that the parameters are independent for each configuration of the discrete parents, *i.e.*

$$p(\theta) = \prod_{\delta\in\Delta}\prod_{i_{\mathrm{pa}(\delta)}\in\mathcal{I}_{\mathrm{pa}(\delta)}} p(\theta_{\delta|i_{\mathrm{pa}(\delta)}}) \prod_{\gamma\in\Gamma}\prod_{i_{\mathrm{pa}(\gamma)}\in\mathcal{I}_{\mathrm{pa}(\gamma)}} p(\theta_{\gamma|i_{\mathrm{pa}(\gamma)}}), \tag{2}$$

We refer to (2) as *parameter independence*. Further, as we have assumed complete data, the parameters stay independent given data, see Bøttcher (2001). This means that we can learn the parameters of a node independently of the parameters of the other nodes, *i.e.* we update the *local parameter prior* $p(\theta_{v|i_{\mathrm{pa}(v)}})$ for each node v and each configuration of the discrete parents.

As local prior parameter distributions we use the Dirichlet distribution for the discrete variables and the Gaussian inverse-Gamma distribution for the continuous variables. These distributions are conjugate to observations from the respective distributions and this ensures simple calculations of the posterior distributions.

In the next section we present an automated procedure for specifying the local parameter priors associated with any possible DAG. The procedure is called the *master prior procedure*. For the mixed case it is treated in Bøttcher (2001), for the purely discrete and the purely continuous cases it is treated in Heckerman, Geiger, and Chickering (1995) and Geiger and Heckerman (1994), respectively.

4.1 The master prior procedure

The idea in the master prior procedure is that from a given Bayesian network we can deduce parameter priors for any possible DAG. The user just has to specify the Bayesian network as he believes it to be. We call this network a *prior Bayesian network*.

1. Specify a prior Bayesian network, *i.e.* a prior DAG and prior local probability distributions. Calculate the joint prior distribution.

2. From this joint prior distribution, the marginal distribution of all parameters in the family consisting of the node and its parents can be determined. We call this the *master prior*.

3. The local parameter priors are now determined by conditioning in the master prior distribution.

This procedure ensures parameter independence. Further, it has the property that if a node has the same set of parents in two different networks, then the local parameter prior for this node will be the same in the two networks. Therefore, we only have to deduce the local parameter prior for a node given the same set of parents once. This property is called *parameter modularity*.

4.2 Master prior for discrete nodes

Let $\Psi = (\Psi_i)_{i\in\mathcal{I}}$ be the parameters for the joint distribution of the discrete variables. The joint prior parameter distribution is assumed to be a Dirichlet distribution

$$p(\Psi) \sim \mathcal{D}(\alpha),$$

with hyperparameters $\alpha = (\alpha_i)_{i\in\mathcal{I}}$. To specify this Dirichlet distribution, we need to specify these hyperparameters. Consider the following relation for the Dirichlet distribution,

$$p(i) = \mathbb{E}(\Psi_i) = \frac{\alpha_i}{N},$$

with $N = \sum_{i\in\mathcal{I}} \alpha_i$. Now we use the probabilities in the prior network as an estimate of $\mathbb{E}(\Psi_i)$, so we only need to determine N in order to calculate the parameters α_i.

We determine N by using the notion of an imaginary data base. We imagine that we have a data base of cases, from which we have updated the distribution of Ψ out of total ignorance. The *imaginary sample size* of this imaginary data base is thus N. It expresses how much confidence we have in the (in)dependencies expressed in the prior network, see Heckerman et al. (1995).

We use this joint distribution to deduce the master prior distribution of the family $A = \delta \cup \mathrm{pa}(\delta)$. Let

$$\alpha_{i_A} = \sum_{j:j_A=i_A} \alpha_j,$$

and let $\alpha_A = (\alpha_{i_A})_{i_A\in\mathcal{I}_A}$. Then the marginal distribution of Ψ_A is Dirichlet, $p(\Psi_A) \sim \mathcal{D}(\alpha_A)$. This is the master prior in the discrete case. The local parameter priors can now be found by conditioning in these master prior distributions.

4.3 Master prior for continuous nodes

Bøttcher (2001) derived this procedure in the mixed case. For a configuration i of the discrete variables we let $\nu_i = \rho_i = \alpha_i$, where α_i was determined in Section 4.2. Also, $\Phi_i = (\nu_i - 1)\Sigma_i$.

The joint parameter priors are assumed to be distributed as

$$\begin{aligned} p(M_i|\Sigma_i) &= \mathcal{N}\left(\mu_i, \frac{1}{\nu_i}\Sigma_i\right) \\ p(\Sigma_i) &= \mathcal{IW}(\rho_i, \Phi_i). \end{aligned}$$

We cannot use these distributions to derive priors for other networks, so instead we use the imaginary data base to derive local master priors.

Define the notation

$$\rho_{i_{A\cap\Delta}} = \sum_{j:j_{A\cap\Delta}=i_{A\cap\Delta}} \rho_j$$

and similarly for $\nu_{i_{A\cap\Delta}}$ and $\Phi_{i_{A\cap\Delta}}$. For the family $A = \gamma \cup \mathrm{pa}(\gamma)$, the local master prior is then found as

$$\begin{aligned} \Sigma_{A\cap\Gamma|i_{A\cap\Delta}} &\sim \mathcal{IW}(\rho_{i_{A\cap\Delta}}, \tilde{\Phi}_{A\cap\Gamma|i_{A\cap\Delta}}) \\ M_{A\cap\Gamma|i_{A\cap\Delta}}|\Sigma_{A\cap\Gamma|i_{A\cap\Delta}} &\sim \mathcal{N}\left(\bar{\mu}_{A\cap\Gamma|i_{A\cap\Delta}}, \frac{1}{\nu_{i_{A\cap\Delta}}}\Sigma_{A\cap\Gamma|i_{A\cap\Delta}}\right), \end{aligned}$$

where

$$\begin{aligned} \bar{\mu}_{i_{A\cap\Delta}} &= \frac{\sum_{j:j_{A\cap\Delta}=i_{A\cap\Delta}} \mu_j\nu_j}{\nu_{i_{A\cap\Delta}}} \\ \tilde{\Phi}_{A\cap\Gamma|i_{A\cap\Delta}} &= \Phi_{i_{A\cap\Delta}} + \sum_{j:j_{A\cap\Delta}=i_{A\cap\Delta}} \nu_j(\mu_j - \bar{\mu}_{i_{A\cap\Delta}})(\mu_j - \bar{\mu}_{i_{A\cap\Delta}})^\top. \end{aligned}$$

Again, the local parameter priors can be found by conditioning in these local master priors.

4.4 The learning procedure in `deal`

The parameters of the joint distribution of the variables in the network are determined by the function `jointprior()` with the size of the imaginary data base as optional argument. If the size is not specified, `deal` sets the size to a reasonably small value.

```
ksl.prior <- jointprior(ksl.nw)    ## auto set size of imaginary data base
ksl.prior <- jointprior(ksl.nw,12) ## set size of imaginary data base to 12
```

The parameters in the object `ksl.prior` may be assessed as the attributes `jointalpha`, `jointnu`, `jointrho` and `jointphi`.

The procedure `learn()` determines the master prior, local parameter priors and local parameter posteriors,

```
ksl.nw <- learn(ksl.nw,ksl,ksl.prior)$nw
```

The result is attached to each node as the attributes `condprior` and `condposterior`. These contain the parameters in the local prior distribution and the parameters in the local posterior distribution, respectively.

5 Learning the structure

In this section we will show how to learn the structure of the DAG from data. The section is based on Bøttcher (2001), Heckerman et al. (1995) and Geiger and Heckerman (1994).

As a measure of how well a DAG D represents the conditional independencies between the random variables, we use the relative probability

$$S(D) = p(D, d) = p(d|D)p(D),$$

and refer to it as a *network score*.

The network score factorizes into a discrete part and a mixed part as

$$S(D) = \prod_{\delta\in\Delta}\prod_{i_{\mathrm{pa}(\delta)}\in\mathcal{I}_{\mathrm{pa}(\delta)}} S_{\delta|i_{\mathrm{pa}(\delta)}}(D) \prod_{\gamma\in\Gamma}\prod_{i_{\mathrm{pa}(\gamma)}\in\mathcal{I}_{\mathrm{pa}(\gamma)}} S_{\gamma|i_{\mathrm{pa}(\gamma)}}(D).$$

Note that the network score is a product over terms involving only one node and its parents. This property is called *decomposability*. It can be shown that the network scores for two independence equivalent DAGs are equal. This property is called *likelihood equivalence* and it is a property of the master prior procedure.

In `deal` we use, for computational reasons, the logarithm of the network score. The log network score contribution of a node is evaluated whenever the node is learned and the log network score is updated and is stored in the `score` attribute of the network.

5.1 Model search

In principle, we could evaluate the network score for all possible DAGs. However, the number of possible DAGs grows more than exponentially with the number of nodes and if the number of random variables in a network is large, it is not computationally possible to calculate the network score for all the possible DAGs. For these situations a strategy for searching for DAGs with high score is needed. In `deal`, the search strategy *greedy search with random restarts*, see *e.g.* Heckerman et al. (1995), is implemented. As a way of comparing the network scores for two different DAGs, D and D^*, we use the posterior odds,

$$\frac{p(D|d)}{p(D^*|d)} = \frac{p(D,d)}{p(D^*,d)} = \frac{p(D)}{p(D^*)} \times \frac{p(d|D)}{p(d|D^*)},$$

where $p(D)/p(D^*)$ is the prior odds and $p(d|D)/p(d|D^*)$ is the Bayes factor. At the moment, the only option in `deal` for specifying prior distribution over DAGs is to let all DAGs be equally likely, so the prior odds are always equal to one. Therefore, we use the Bayes factor for comparing two different DAGs.

In greedy search we compare models that differ only by a single arrow, either added, removed or reversed. In these cases, the Bayes factor is especially simple, because of decomposability of the network score.

To manually assess the network score of a network (*e.g.* to use as initial network in a search), use

```
ksl.nw <- drawnetwork(ksl.nw,ksl,ksl.prior)$nw
```

In the `drawnetwork()` procedure, it is possible to mark (ban) some of the arrows. In the search, `deal` then disregards any DAG which contains any of these arrows, and this reduces the search space.

The automated search algorithmis implemented in the function `heuristic()`. The initial network is perturbed according to the parameter `degree` and the search is performed starting with the perturbed network. The process is restarted the number of times specified by the option `restart`. A network family of all visited networks is returned.

```
ksl.h <- heuristic(ksl.nw,ksl,ksl.prior,restart=10,degree=5)$nw
```

6 Hugin interface

A network object may be written to a file in the Hugin `.net` language. Hugin (`http://www.hugin.com`) is commercial software for inference in Bayesian networks. Hugin has the ability to learn networks with only discrete networks but cannot learn either purely continuous or mixed networks. `deal` may therefore be used for this purpose and the result can then be transferred to Hugin. The procedure `savenet()` saves a network to a file. For each node, we use point estimates of the parameters in the local probability distributions. The `readnet()` procedure reads the network structure from a file but does not, however, read the probability distributions. This is planned to be included in a future version of `deal`.

7 Example

In this section, we describe the analysis of the *ksl* data that has been used as illustration throughout the paper. The data set, included in Badsberg (1995), is from a study measuring health and social characteristics of representative samples of Danish 70-year old people, taken in 1967 and 1984. In total, 1083 cases have been recorded and each case contains observations on nine different variables, see Table 1.

Node index	Variable	Explanation
1	`Fev`	Forced ejection volume – lung function
2	`Kol`	Cholesterol
3	`Hyp`	Hypertension (no/yes)
4	`BMI`	Body Mass Index
5	`Smok`	Smoking (no/yes)
6	`Alc`	Alcohol consumption (seldom/frequently)
7	`Work`	Working (yes/no)
8	`Sex`	Gender (male/female)
9	`Year`	Survey year (1967/1984)

Table 1: Variables in the *ksl* data set. The variables `Fev`, `Kol`, `BMI` are continuous variables and the rest are discrete variables.

The purpose of our analysis is to find dependency relations between the variables. One interest is to determine which variables influence the presence or absence of hypertension. From a medical viewpoint, it is possible that hypertension is influenced by some of the continuous variables `Fev`, `Kol` and `BMI`. However, in `deal` we do not allow continuous parents of discrete nodes, so we cannot describe such a relation. A way to overcome this problem is to treat `Hyp` as a continuous variable, even though this is obviously not most natural. This is done in the analysis below. Further, the initial data analysis indicates a transformation of `BMI` into log(`BMI`). With these adjustments, the data set is ready for analysis in `deal`.

We have no prior knowledge about specific dependency relations, so for simplicity we use the empty DAG as the prior DAG and let the probability distribution of the discrete variables be uniform. The assessment of the probability distribution for the continuous variables is based on data, as described in Section 3.1.

```
ksl.nw    <- network(ksl)         # specify prior network
```

```
ksl.prior <- jointprior(ksl.nw)   # make joint prior distribution
```

We do not allow arrows into `Sex` and `Year`, as none of the other variables can influence these variables. So we create a ban list which is attached to the network. The ban list is a matrix with two columns. Each row contains the directed edge that is not allowed.

```
## ban arrows towards Sex and Year
banlist <- matrix(c(5,5,6,6,7,7,9,
                    8,9,8,9,8,9,8),ncol=2)
ksl.nw$banlist <- banlist
```

Finally, the parameters in the network are learned and structural learning is used with the prior DAG as starting point.

```
ksl.nw   <- learn(ksl.nw,ksl,ksl.prior)$nw
result   <- heuristic(ksl.nw,ksl,ksl.prior,restart=2,degree=10,trace=TRUE)
thebest  <- result$nw[[1]]
savenet(thebest, "ksl.net")
```

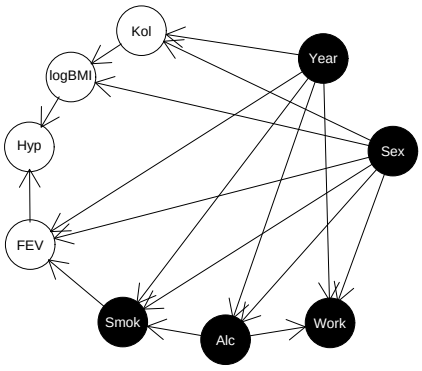

Figure 1: The network with the highest score, $\log(\texttt{score}) = -15957.91$.

The resulting network `thebest` is shown in Figure 1 and it is the network with the highest network score among those networks that have been tried through the search.

In the result we see for the discrete variables that `Alc`, `Smok` and `Work` depend directly on `Sex` and `Year`. In addition, `Smok` and `Work` also depend on `Alc`. These two arrows are, however, not causal arrows, as `Smok` $\leftarrow$ `Alc` $\rightarrow$ `Work` in the given DAG represents the same probability distribution as the relations `Smok` $\leftarrow$ `Alc` $\leftarrow$ `Work` and `Smok` $\rightarrow$ `Alc` $\rightarrow$ `Work`, *i.e.* the three DAGs are independence equivalent. `Year` and `Sex` are independent on all variables, as specified in the ban list. For the continuous

variables all the arrows are causal arrows. We see that `Fev` depends directly on `Year`, `Sex` and `Smok`. So given these variables, `Fev` is conditional independent on the rest of the variables. `Kol` depends directly on `Year` and `Sex`, and `logBMI` depends directly on `Kol` and `Sex`. Given `logBMI` and `Fev`, the variable `Hyp` is conditionally independent on the rest of the variables. So according to this study, hypertension can be determined by the body mass index and the lung function forced ejection volume. However, as `Hyp` is not continuous by nature, other analyses should be performed with `Hyp` as a discrete variable, *e.g.* a logistic regression with `Hyp` as a response and the remaning as explanatory variables. Such an analysis indicates that, in addition, `Sex` and `Smok` may influence `Hyp` but otherwise identifies `logBMI` as the main predictor.

8 Discussion and future work

`deal` is a tool box that adds functionality to R so that Bayesian networks may be used in conjunction with other statistical methods available in R for analysing data. In particular, `deal` is part of the gR project, which is a newly initiated workgroup with the aim of developing procedures in R for supporting data analysis with graphical models, see http://www.r-project.org/gR.

In addition to methods for analysing networks with either discrete or continuous variables, `deal` handles networks with mixed variables. `deal` has some limitations and we plan to extend the package with the procedures described below. Also, it is the intention that the procedures in `deal` will eventually be adjusted to the other procedures developed under the gR project. The methods in `deal` are only applicable on complete data sets and in the future we would like to incorporate procedures for handling data with missing values and networks with latent variables. The criteria for comparing the different network structures in `deal` is the BDe criteria. We intend to also incorporate the Bayesian Information Criteria (BIC) and Akaikes Information Criteria (AIC) and let it be up to the user to decide which criteria to use. Another possible extension of `deal` is to incorporate procedures for specifying mixed networks, where the variance in the mixed part of the network does not depend on the discrete parents, but the mean does. Finally, we are working on an implementation of the greedy equivalence search (GES) algorithm, see Chickering (2002), which is an algorithm for search between equivalence classes. Asymptotically, for the size of the database tending to infinity, this algorithm guarantees that the search terminates with the network with the highest network score.

Acknowledgements

The work has been supported by Novo Nordisk A/S.

References

J.H. Badsberg. *An Environment for Graphical Models.* PhD thesis, Aalborg University, 1995.

S.G. Bøttcher. Learning Bayesian networks with mixed variables. In *Proceedings of the Eighth International Workshop in Artificial Intelligence and Statistics*, 2001.

D.M. Chickering. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3:507–554, November 2002.

R.G. Cowell, A.P. Dawid, S.L. Lauritzen, and D.J. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Springer-Verlag, 1999.

D. Geiger and D. Heckerman. Learning Gaussian networks. Technical Report MSR-TR-94-10, Microsoft Research, 1994.

D. Heckerman, D. Geiger, and D. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 1995.

R. Ihaka and R. Gentleman. R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, 5:299–314, 1996.

S.L. Lauritzen. Propagation of probabilities, means and variances in mixed graphical association models. *Journal of the American Statistical Association*, 82:1082–1108, 1992.

S.L. Lauritzen. Some modern applications of graphical models. In P.J. Green, N.L. Hjort, and S. Richardson, editors, *Highly Structured Stochastic Systems*. Oxford University Press, 2003.

R.D. Shachter and C.R. Kenley. Gaussian influence diagrams. *Management Science*, 35:527–550, 1989.

Corresponding author

Claus Dethlefsen
Dept. of Mathematical Sciences
Aalborg University
Fr. Bajers Vej 7G
9220 Aalborg, Denmark
E-mail: dethlef@math.auc.dk