BayNet: structure discovery with Bayesian Networks

Mikel Esnaola
mesnaola@creal.cat

Juan Ramón González
jrgonzalez@creal.cat

http://www.creal.cat/jrgonzalez/software.htm

June 22, 2011

1 Introduction

This document gives an introduction and overview of the R package BayNet, which provides search methods based on Bayesian Networks for discovering structural dependencies from data. The routines can be applied to discrete multinomial data without missing values.

2 Getting started

In order to get the best Bayesian Network plots, the Rgraphviz Bioconductor package has been used along with the SparseM package. The last of them can be found at the CRAN repositories and can be easily installed.

The Rgraphviz package needs an R independent program called graphviz. Linux users can easily install it because it can be found in most of the public repositories. For example, if we use a Debian based OS (as Ubuntu), we can install it running the command apt-get install graphviz, and if we use a Red Hat based OS (as Fedora or CentOS), we can install it running yum install graphviz.

If we use Windows, this installation is more difficult. The graphviz program can be downloaded from http://www.graphviz.org/Download_windows.php. In the *README* file of the Rgraphviz package it can be found a step by step explanation of the installation. Anyway, as the installation process is quite difficult, a normal plot function has been included.

3 Data

In order to illustrate how to use the BayNet package we will be using the coronary dataset, which is also included in the package. We can load this dataset using the following instruction.

```
> library(BayNet)
> data(coronary)
```

The coronary dataset contains information about 6 variables considered to be probable risk factors for coronary trombosis: *smoking, M. Work* (strenuous mental work), *P. Work* (strenuous physical work), *pressure* (systolic blood pressure), *proteins* (ratio of beta and alpha lipoproteins) and *family* (family anamnesis of coronary heart disease). All variables are categorical with two levels. For more information about the levels of each variable and their distribution:

```
> summary(coronary)
```

```
 Smoking   M. Work    P. Work   Pressure    Proteins  Family
 no :961   no :1130   no :927   <140:1054   <3:1061   neg:1581
 yes:880   yes: 711   yes:914   >140: 787   >3: 780   pos: 260
```

4 Greedy Search

The Greedy Search is a fast heuristic method. It starts with an initial random Bayesian Network and iteratively finds the best perturbation (the one that most improves the score) until all perturbations lead to a worse Bayesian Network. The function that performs this algorithm is greedySearch. The arguments of this function are data, Niter (number of algorithm iterations to perform) and verbose (should additional information be displayed?, default is FALSE).

```
> gr <- greedySearch(data=coronary,Niter=5)
```

After fitting the model we can obtain different information about the best (or other) Bayesian Network using the generic R functions print, summary and plot.

```
> gr
```

```
Greedy Search result for 5 initial graphs

*****************************************

Best Bayesian Network, with score = -6679.88:

Model:
[Family][M. Work|Family][P. Work|M. Work]
[Proteins|M. Work][Pressure|M. Work:P. Work:Proteins]
[Smoking|M. Work:P. Work:Pressure:Proteins:Family]



         Smoking M. Work P. Work Pressure Proteins Family
Smoking        0       0       0        0        0      0
M. Work        1       0       1        1        1      0
P. Work        1       0       0        1        0      0
Pressure       1       0       0        0        0      0
Proteins       1       0       0        1        0      0
Family         1       1       0        0        0      0
```

```
****************************************

Scores:

     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-6714.460 -6693.634 -6685.607 -6691.089 -6681.861 -6679.880
```

The print method shows information about the number of iterations performed by the algorithm, the best Bayesian Network's model[1], score and adjacency matrix[2]; and a common summary of the scores of all iterations' results.

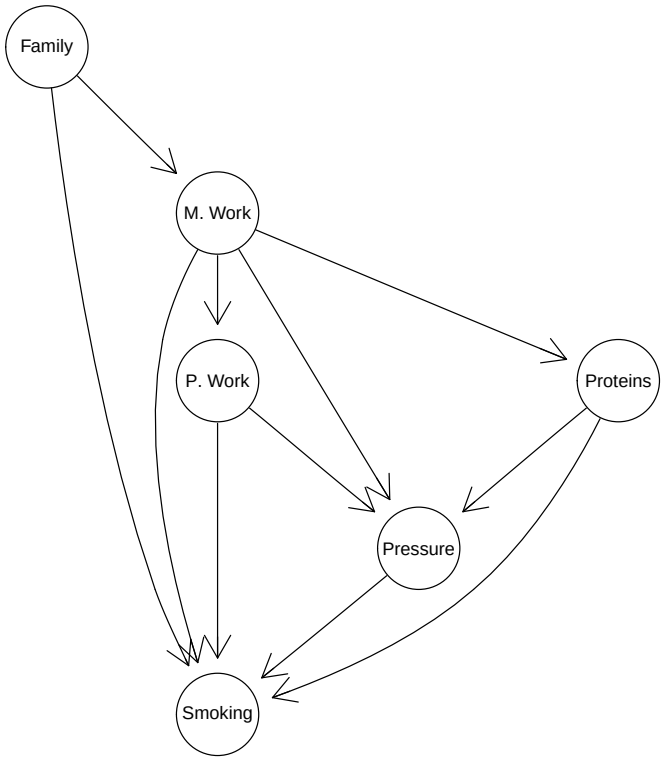
```
> summary(gr)

Greedy Search object with 5 Bayesian Networks

Best 5 BNs:
1:  Score = -6679.88
    Model: [Family][M. Work|Family][P. Work|M. Work]
[Proteins|M. Work][Pressure|M. Work:P. Work:Proteins]
[Smoking|M. Work:P. Work:Pressure:Proteins:Family]

2:  Score = -6681.861
    Model: [Family][M. Work|Family][P. Work|M. Work]
[Proteins|M. Work][Pressure|M. Work:P. Work:Proteins]
[Smoking|M. Work:P. Work:Pressure:Proteins:Family]

3:  Score = -6685.607
    Model: [Family][M. Work|Family][P. Work|M. Work]
[Proteins|M. Work][Pressure|M. Work:P. Work:Proteins]
[Smoking|M. Work:P. Work:Pressure:Proteins:Family]

4:  Score = -6693.634
    Model: [Family][M. Work|Family][P. Work|M. Work]
[Proteins|M. Work][Pressure|M. Work:P. Work:Proteins]
[Smoking|M. Work:P. Work:Pressure:Proteins:Family]

5:  Score = -6714.46
    Model: [Family][M. Work|Family][P. Work|M. Work]
[Smoking|M. Work:P. Work:Family][Pressure|Smoking:M. Work:P. Work]
[Proteins|Smoking:M. Work:P. Work:Pressure:Family]
```

The summary shows the best networks (up to 10) obtained by the Greedy Search algorithm along with their scores and models.

[1]The interpretation of the model is the following: each local node is represented between brackets with its parents. If a node X has no parents then it appears alone $[X]$, if a node X has n parents $Y_1, Y_2, \ldots, Y_n$ then it appears first X and then its parents $[X : Y_1, Y_2, \ldots, Y_n]$.

[2]The adjacency matrix is a binary matrix. Each row represents the sons of the corresponding node. Likewise, each column represents the parents of the corresponding node. For instance, looking at the first row (Smoking) we can see that this variable has no sons in the best Bayesian Network found by Greedy Search. Looking at the first column we can also see that all variables are parents of Smoking variable.

```
> gr.best <- bestBN(gr)
> plot(gr.best)
```



5 Structure MCMC

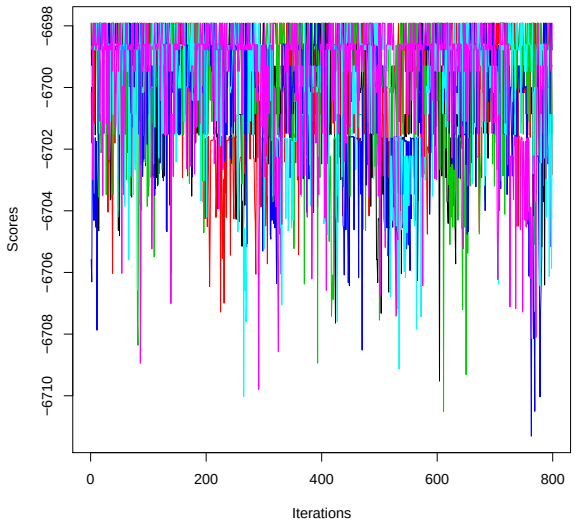
The Structure MCMC is a Metropolis-Hastings random walk in the Bayesian Network space. It allows to obtain the posterior distribution of the Bayesian Networks given the data, but when the number of variables is big (more than 14-15) it can have convergence problems. Once we obtain the posterior distribution we can extract the best Bayesian Network (or another one) as we did previously, or we can average over the posterior distribution.

The function to perform this algorithm is the structureMCMC. Its arguments are data, K (maximum number of parents per node, default is 3), Nchain (number of chains, i.e., number of independent algorithm runs, default is 1), Niter (number of iterations per chain, default is 10^5), thin (sampling interval, default is Niter/1000), burn (number of initial discarded iterations, default is $2 \cdot$Niter/1000) and verbose (should information be displayed?, default is FALSE).

```
> st <- structureMCMC(data=coronary,Nchain=6,Niter=1e5)
```

First of all, we can check convergence by plotting every chain's scores.

```
> plot(st)
```



The last graphic represents the progress of each chain's score. It allows us to check the mixing rate of all chains. As previously, we can also extract the Bayesian Network we wish, or directly get the best one.

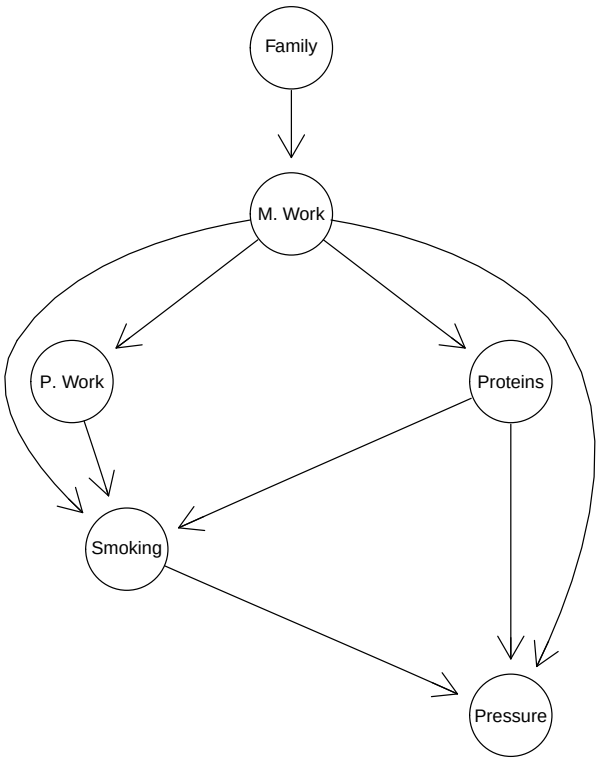
```
> st.best <- bestBN(st)
> st.best
```

```
Bayesian Network from StChain of 6 nodes with 9 edges
Score = -6697.915

Model: [Family][M. Work|Family][P. Work|M. Work]
[Proteins|M. Work][Smoking|M. Work:P. Work:Proteins]
[Pressure|Smoking:M. Work:Proteins]
```

	Smoking	M. Work	P. Work	Pressure	Proteins	Family
Smoking	0	0	0	1	0	0
M. Work	1	0	1	1	1	0
P. Work	1	0	0	0	0	0
Pressure	0	0	0	0	0	0
Proteins	1	0	0	1	0	0
Family	0	1	0	0	0	0

If we wanted to extract a particular Bayesian Network from a certain chain we can do it using the extractBN function, which has arguments chain (result of one of the search methods), bn (an integer representing which Bayesian Network we want to extract) adn c (an integer representing the chain from which we want to extract the Bayesian Network).

```
> plot(st.best)
```



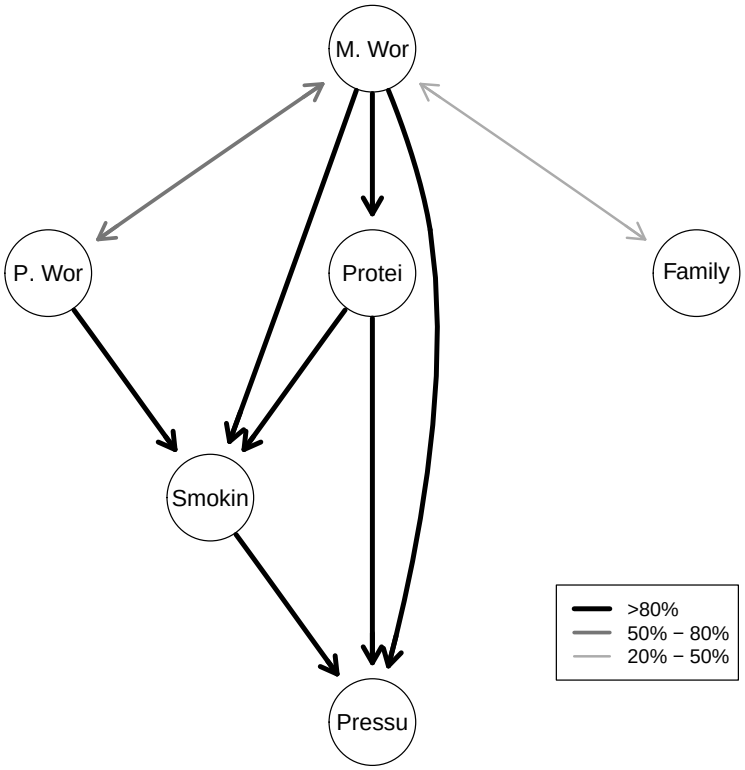
Anyway, using a single estimate of the whole posterior distribution is a huge loss of information. If we only use the best Bayesian Network and there are another ones with similar scores it is much better to average over all of them. This can be done by the means of Bayesian Model Averaging. This technique allows us to extract much more information and get as result the probability of each possible edge. To do so we can use the average function, which has arguments chain (result of a MCMC search method), p (a number between 0 and 100 representing the percentage of the posterior distribution we want to incorporate in the average) and k (an integer representing which chain we want to average from)

```
> av <- average(chain=st,p=50,k=1)
> av

Average of 'StChain':
         Smoking M. Work P. Work Pressure Proteins Family
Smoking     0.00    0.00    0.00     1.00     0.00   0.00
M. Work     1.00    0.00    0.70     1.00     0.88   0.32
P. Work     1.00    0.30    0.00     0.01     0.00   0.00
Pressure    0.00    0.00    0.00     0.00     0.00   0.00
Proteins    1.00    0.12    0.00     0.99     0.00   0.00
Family      0.00    0.68    0.00     0.00     0.00   0.00
```

```
Summary of the 13 not-null edges
         Edges
80%-100%     7
50%-80%      2
20%-50%      2
0%-20%       2
```

The print method shows the adjacency matrix with each possible edge's probability after averaging. It also shows a brief summary of the distribution of the not-null edges. There is an specific plot method for the averaged Bayesian Networks. It shows the strength of every edge, where the strength is its frequency in the selected subset of the posterior distribution.

```
> plot(av)
```



6 Order MCMC

The Order MCMC performs also a Metropolis-Hastings random walk, but instead of searching in the Bayesian Network space, it searchs in the topological order space. This algorithm allows more variables than the Structure MCMC, but its results are not Bayesian Networks. As a result, another process is needed after obtaining the posterior distribution of orders.

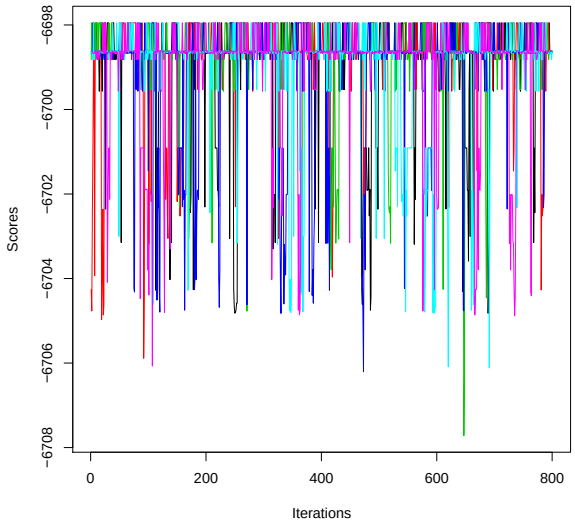
This algorithm can be executed using the orderMCMC function, which has arguments data, K (maximum number of parents per node, default is 3), Nchain (number of chains, i.e., number of independent algorithm runs, default is 1), Niter (number of iterations per chain, default is 10^4), thin (sampling interval, default is Niter/100), burn (number of initial discarded iterations, default is $2 \cdot$ Niter/10) and verbose (should information be displayed?, default is FALSE).

```
> or <- orderMCMC(data=coronary,Nchain=6,Niter=1e4)

> or

Order MCMC chains:

Number of chains: 6
Number of orders per chain: 800

************************************************

Best order for each chain:
-Chain1:
[Smoking][M. Work][P. Work][Pressure][Proteins]
[Family]
-Chain2:
[Smoking][M. Work][P. Work][Pressure][Proteins]
[Family]
-Chain3:
[Smoking][M. Work][P. Work][Pressure][Proteins]
[Family]
-Chain4:
[Smoking][M. Work][P. Work][Pressure][Proteins]
[Family]
-Chain5:
[Smoking][M. Work][P. Work][Pressure][Proteins]
[Family]
-Chain6:
[Smoking][M. Work][P. Work][Pressure][Proteins]
[Family]

************************************************

Summary of the scores for each chain:
-Chain1:
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-6704.822 -6698.673 -6698.621 -6698.956 -6698.617 -6697.948
```

```
-Chain2:
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-6705.888 -6698.673 -6698.621 -6698.804 -6698.617 -6697.948
-Chain3:
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-6707.719 -6698.673 -6698.621 -6698.768 -6698.617 -6697.948
-Chain4:
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-6706.208 -6698.673 -6698.621 -6699.025 -6698.617 -6697.948
-Chain5:
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-6706.115 -6698.673 -6698.621 -6699.002 -6698.617 -6697.948
-Chain6:
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-6706.070 -6698.673 -6698.621 -6698.976 -6698.617 -6697.948
```

The print's result is similar to the one obtained with the Structure MCMC, but this time the best orders for each chain are listed. It is also convenient to see if the chains have mixed in order to check convergence.

```
> plot(or)
```



Once we obtain the posterior distribution of orders we can extract the best one and sample random graphs from it. We can do this using the BNsampler function, which has arguments order (the order to sample from), ngraphs (integer representing the number of graphs to be sampled) and K (maximum numer of parents allowed for the sampled graphs, default is the same number as the one used in the orderMCMC process).

```
> or.best <- bestOrder(or)
> or.best
```

```
Order of 6 nodes
Score = -6697.948

Model: [Smoking][M. Work][P. Work][Pressure][Proteins]
[Family]
```

```
> or.best.s <- BNsampler(order=or.best, ngraphs=10)
> or.best.s
```

```
10 sampled BNs

Original order:
[Smoking][M. Work][P. Work][Pressure][Proteins]
[Family]

Best BN with score = -6697.915
Model: [Family][M. Work|Family][P. Work|M. Work]
[Proteins|M. Work][Smoking|M. Work:P. Work:Proteins]
[Pressure|Smoking:M. Work:Proteins]
```

	Smoking	M. Work	P. Work	Pressure	Proteins	Family
Smoking	0	0	0	1	0	0
M. Work	1	0	1	1	1	0
P. Work	1	0	0	0	0	0
Pressure	0	0	0	0	0	0
Proteins	1	0	0	1	0	0
Family	0	1	0	0	0	0

```
Summary of the scores:
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-6701.487 -6699.300 -6697.915 -6698.688 -6697.915 -6697.915
```

We can extract the best (or another) Bayesian Network of the sampled graphs.

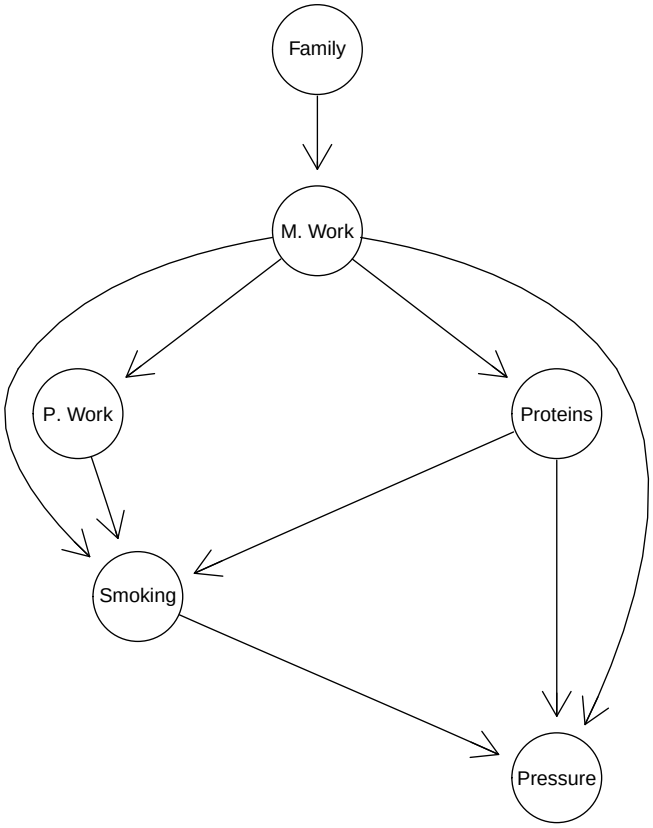
```
> or.best.s.best <- bestBN(or.best.s)
> or.best.s.best
```

```
Bayesian Network from OrChain of 6 nodes with 9 edges
Score = -6697.915

Model: [Family][M. Work|Family][P. Work|M. Work]
[Proteins|M. Work][Smoking|M. Work:P. Work:Proteins]
[Pressure|Smoking:M. Work:Proteins]
```

	Smoking	M. Work	P. Work	Pressure	Proteins	Family
Smoking	0	0	0	1	0	0
M. Work	1	0	1	1	1	0
P. Work	1	0	0	0	0	0
Pressure	0	0	0	0	0	0

```
Proteins       1       0       0       1       0       0
Family         0       1       0       0       0       0
```

```
> plot(or.best.s.best)
```



Another way of passing from orders to graphs is averaging directly to obtain the probability of each edge.

```
> or.av <- average(or,p=50,k=1)
> or.av
```

```
Average of 'OrChain':
         Smoking M. Work P. Work Pressure Proteins Family
Smoking     0.00    0.00    0.00     1.00     0.00   0.00
M. Work     1.00    0.00    1.00     1.00     1.00   0.51
P. Work     1.00    0.00    0.00     0.20     0.01   0.01
Pressure    0.00    0.00    0.00     0.00     0.00   0.00
Proteins    1.00    0.00    0.01     0.80     0.00   0.01
Family      0.00    0.44    0.02     0.00     0.04   0.00
```

```
Summary of the 17 not-null edges
          Edges
80%-100%      8
50%-80%       1
20%-50%       2
0%-20%        6

> plot(or.av)
```