K2 Algorithm for learning Bayesian Networks

Alessandro Maria Capodaglio
Alessio Giorlandino
Carmen Martin Turrero

07/07/22

Bayesian Networks

Definition: A Bayesian Network is a DAG (**d**irected **a**cyclic **g**raph) characterized by :

- a structure B_S, in which every node represents a variable and each arc represents the probabilistic dependence between the involved nodes.
- a set of conditional probabilities B_P, i.e. for each node there exist a conditional probability function that relates it to its parents $\vec{\pi}_i$

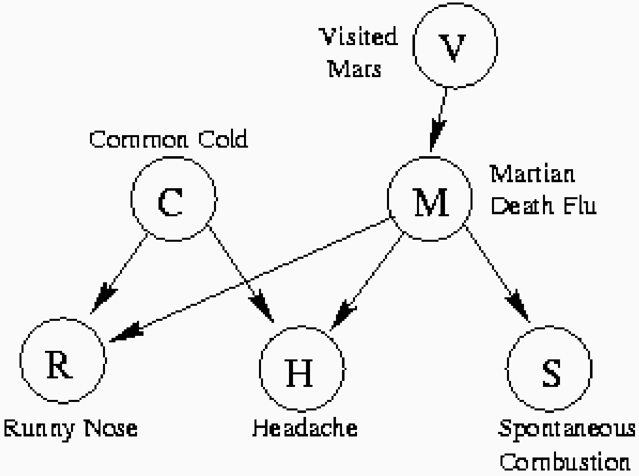
Main Feature

Factorization of joint probability over the parents: BN are capable of representing the joint probability of a set of variables $\{x_i\}_{i=1}^N$ over *any* (discrete) probability space Ω:

$$\forall\ P(x_1, ..., x_N),\ \exists \text{ BN } B(B_S, B_P) \text{ such that } \boxed{P(x_1, ..., x_N) = \prod_{i=1}^{N} P(x_i|\vec{\pi}_i)}$$

Goal: find the most probable BN structure B_S from a data-set.
Just the structure can give us an insight on the causal relationships between the variables.

Bayesian Networks - Simple Example



Bayesian Networks - Formalism

Notation:

- D : data-set of m cases
- Z : are the set of variables $\{x_i\}_{i=1}^{N}$
- $\vec{\pi}_i$: set of parents of node i

Approach:

For any two possible structures B_{Si}, B_{Sj} compute $\frac{P(B_{Si}|D)}{P(B_{Sj}|D)}$
$\implies$ we can compare all structures and keep the most probable.

Compute $P(B_S|D)$ - Assumptions

1. $\{X_i\}_{i=1}^{n}$ are all discrete variables;
2. Cases (i.e. records) are all independent and no cases have missing values;
3. Any B_P is equally likely given B_S.

Assumption 1

1. $\{X_i\}_{i=1}^n$ are all discrete variables:

$$\rightarrow \quad \mathbb{P}(B_S, D) = \int_{B_P} \boxed{\mathbb{P}(D|B_S, B_P)}\, f(B_P|B_S)\, \mathbb{P}(B_S)\, dB_P$$

- the integral is over *all possible belief networks with structure* B_S
- the boxed term is a *pmf* and not a *pdf* (because of the discrete hypothesis)

Assumptions

1. $\{X_i\}_{i=1}^n$ are all discrete variables;
2. **Cases (i.e. records) are all independent and no cases have missing values**;
3. Any B_P is equally likely given B_S.

Assumption 2

2. Cases are all independent

$$\rightarrow \quad D = \{C_n\}_{n=1}^{m}$$

$$\Rightarrow \mathbb{P}(B_S, D) = \int_{B_P} \prod_{n=1}^{m} \mathbb{P}(C_n|B_S, B_P)\, f(B_P|B_S)\, \mathbb{P}(B_S)\, dB_P$$

Assumptions

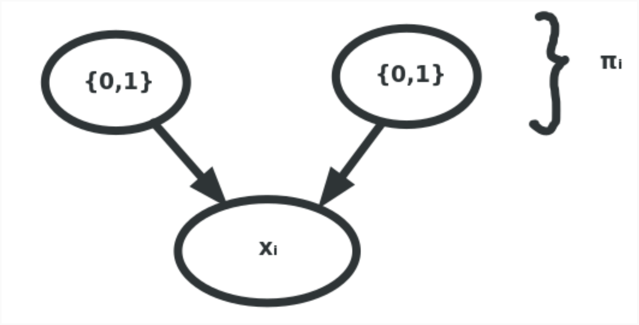
1. $\{X_i\}_{i=1}^{n}$ are all discrete variables;
2. Cases (i.e. records) are all independent and no cases have missing values;
3. **Any B_P is equally likely given B_S.**

Assumption 3

3. The *pdf* $f(B_P|B_S)$ is uniform, meaning that any B_P is equally likely given B_S

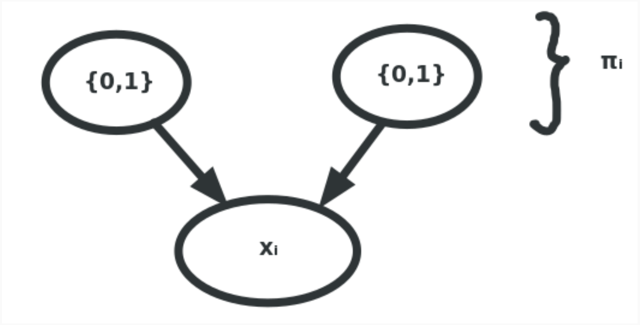
$$\rightarrow \quad \mathbb{P}(B_S|D) = \frac{1}{Z}\int_{B_P} \mathbb{P}(D|B_S, B_P)\, \mathbb{P}(B_S)\, dB_P$$

Notation



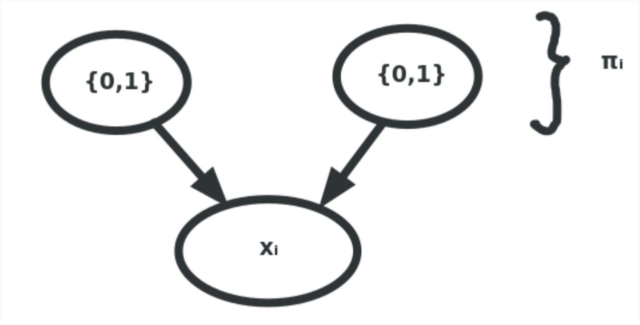
- Let $\vec{\pi}_i$ be the set of parents of x_i.
  $\vec{\pi}_i$ has q_i possible unique realizations $\rightarrow$ indicate as ω_{ij} the j_{th} unique state of $\vec{\pi}_i$

Notation



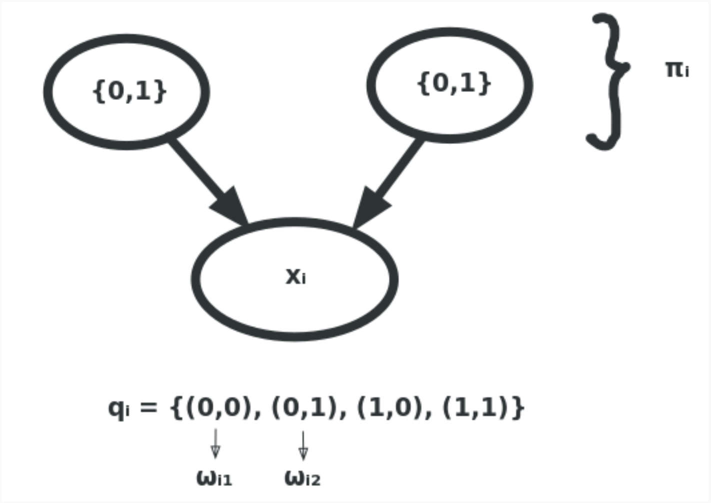
- Let $\vec{\pi}_i$ be the set of parents of x_i.
  $\vec{\pi}_i$ has q_i possible unique realizations $\rightarrow$ indicate as ω_{ij} the j_{th} unique state of $\vec{\pi}_i$
- Let x_i have support $\left\{\nu_{i1}, ..., \nu_{ir_i}\right\}$ with $i = 1, ..., n$

Notation



- Let $\vec{\pi_i}$ be the set of parents of x_i.
  $\vec{\pi_i}$ has q_i possible unique realizations $\rightarrow$ indicate as ω_{ij} the j_{th} unique state of $\vec{\pi_i}$
- Let x_i have support $\left\{\nu_{i1}, ..., \nu_{ir_i}\right\}$ with $i = 1, ..., n$
- Call N_{ijk} the number of cases in D in which x_i has value ν_{ik} and π_i has values ω_{ij}

Notation



Theorem

Statement: Let D be a dataset of m cases, $Z = \{x_i\}_{i=1}^{n}$ and B_S the BN structure associated to Z.
Given the previous assumptions it follows that:

$$\mathbb{P}(B_S, D) = \mathbb{P}(B_S) \prod_{i=1}^{n} \prod_{i}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}$$

where :

$$N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$$

Compute $\mathbb{P}(B_S|D)$

Note that: $\frac{P(B_{S_i}|D)}{P(B_{S_j}|D)} = \frac{P(B_{S_i},D)}{P(B_{S_j},D)} \implies$ we can compare all possible structures $B_{S_i},\ B_{S_j}$ and choose the most probable.

Problem: The set Q of possible structures for the set of variables Z has cardinality $|Q| \approx exp(|Z|)$

Idea: sample over $Y \subset Q$ with $|Y|$ big enough such that $\sum_{B_S \in Y} P(B_S, D) \simeq P(D)$, but yet $|Y| << |Q|$.

Exact Method

In principle there is no reason to think that a given structure is more probable than another, hence, it is sensible to assume that $\mathbb{P}(B_S) \sim cost = c$.

$$\Rightarrow \mathbb{P}(B_S, D) = c \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}!$$

$\rightarrow$ new goal:

$$\max_{B_S} \left[\mathbb{P}(B_S, D)\right] = c \prod_{i=1}^{n} \max_{\pi_i} \left[\prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}! \right]$$

From B_S to B_P

Once we have chosen a Dataset and we had found the most probable structure of the relative Bayesian network, we might be interested in calculating the expectations of the network conditional probabilities.
Network conditional probability: Call $\theta_{ijk} = P(x_i = \nu_{ik} | \vec{\pi}_i = w_{ij})$, i.e. the probability that the variable x_i has value ν_{ij} given that its parents are instantiated with value w_{ij}. It can be proven that:

$$\mathbb{E}[\theta_{ijk} | D, B_S, Assumptions] = \frac{N_{ijk} + 1}{N_{ij} + r_i} \xrightarrow{|D|>>1} \frac{N_{ijk}}{N_{ij}}$$

$$Var[\theta_{ijk} | D, B_S, Assumptions] = \frac{(N_{ijk} + 1)(N_{ij} + r_i - N_{ijk} - 1)}{(N_{ij} + r_i)^2 (N_{ij+r_i+1})} \rightarrow \frac{N_{ijk}(N_{ij} - N_{ijk})}{N_{ij}^2 N_{ij}}$$

K2 Algorithm - Idea

Strong assumption: There exist an ordering in the set of variable such that a given node can have as parents *only* nodes that precede it in that given order.

Consequences:

- the first node in the order has no parents
- we don't need to check all the possible combinations; for any node we can inspect only its precedents.

K2 Algorithm - Procedure

Recall that our goal is:

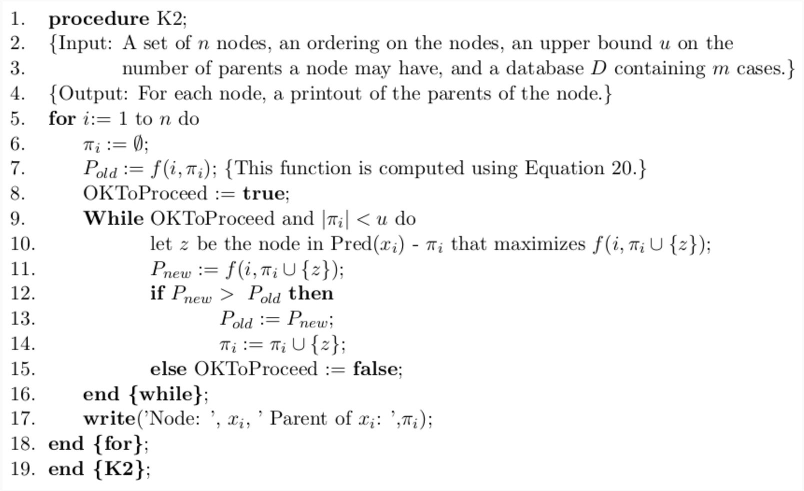
$$\max_{B_S}\left[\mathbb{P}(B_S, D)\right] = c\prod_{i=1}^{n}\max_{\pi_i}\left[\prod_{j=1}^{q_i}\frac{(r_i-1)!}{(N_{ij}+r_i-1)!}\prod_{k=1}^{r_i}N_{ijk}!\right]$$

Hence, for every node the network and for every combination of its parents (drawn only from its precedent nodes) we have to compute:

$$f(i,\vec{\pi}_i) = \prod_{j=1}^{q_i}\frac{(r_i-1)!}{(N_{ij}+r_i-1)!}\prod_{k=1}^{r_i}N_{ijk}!$$

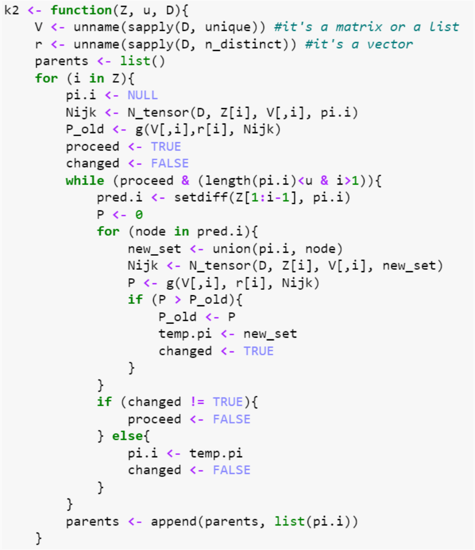
Result: Keep the $\vec{\pi}_i \ \forall\ i = 1, ..., N$ that maximizes $f(i,\vec{\pi}_i)$.

K2 Algorithm - Pseudocode



```
procedure K2;
{Input: A set of $n$ nodes, an ordering on the nodes, an upper bound $u$ on the
        number of parents a node may have, and a database $D$ containing $m$ cases.}
{Output: For each node, a printout of the parents of the node.}
for $i$:= 1 to $n$ do
    $\pi_i := \emptyset$;
    $P_{old} := f(i, \pi_i)$; {This function is computed using Equation 20.}
    OKToProceed := true;
    While OKToProceed and $|\pi_i| < u$ do
        let $z$ be the node in Pred($x_i$) - $\pi_i$ that maximizes $f(i, \pi_i \cup \{z\})$;
        $P_{new} := f(i, \pi_i \cup \{z\})$;
        if $P_{new} > P_{old}$ then
            $P_{old} := P_{new}$;
            $\pi_i := \pi_i \cup \{z\}$;
        else OKToProceed := false;
    end {while};
    write('Node: ', $x_i$, ' Parent of $x_i$: ',$\pi_i$);
end {for};
end {K2};
```

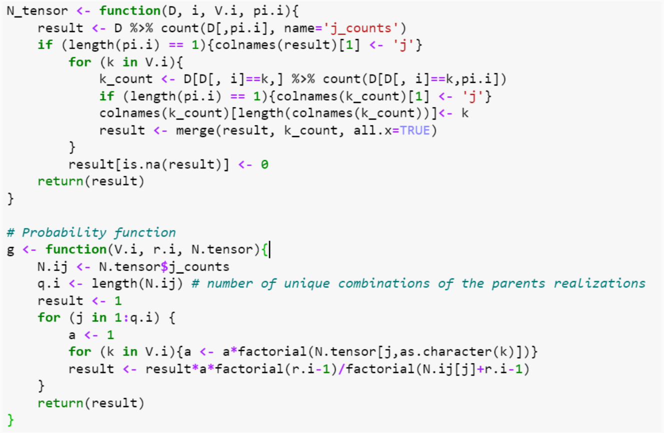
Our Implementation



```r
k2 <- function(Z, u, D){
    V <- unname(sapply(D, unique)) #it's a matrix or a list
    r <- unname(sapply(D, n_distinct)) #it's a vector
    parents <- list()
    for (i in Z){
        pi.i <- NULL
        Nijk <- N_tensor(D, Z[i], V[,i], pi.i)
        P_old <- g(V[,i],r[i], Nijk)
        proceed <- TRUE
        changed <- FALSE
        while (proceed & (length(pi.i)<u & i>1)){
            pred.i <- setdiff(Z[1:i-1], pi.i)
            P <- 0
            for (node in pred.i){
                new_set <- union(pi.i, node)
                Nijk <- N_tensor(D, Z[i], V[,i], new_set)
                P <- g(V[,i], r[i], Nijk)
                if (P > P_old){
                    P_old <- P
                    temp.pi <- new_set
                    changed <- TRUE
                }
            }
            if (changed != TRUE){
                proceed <- FALSE
            } else{
                pi.i <- temp.pi
                changed <- FALSE
            }
        }
        parents <- append(parents, list(pi.i))
    }
```

Figure 1: Main function.

Our Implementation



```r
N_tensor <- function(D, i, V.i, pi.i){
    result <- D %>% count(D[,pi.i], name='j_counts')
    if (length(pi.i) == 1){colnames(result)[1] <- 'j'}
        for (k in V.i){
            k_count <- D[D[, i]==k,] %>% count(D[D[, i]==k,pi.i])
            if (length(pi.i) == 1){colnames(k_count)[1] <- 'j'}
            colnames(k_count)[length(colnames(k_count))]<- k
            result <- merge(result, k_count, all.x=TRUE)
        }
        result[is.na(result)] <- 0
    return(result)
}

# Probability function
g <- function(V.i, r.i, N.tensor){
    N.ij <- N.tensor$j_counts
    q.i <- length(N.ij) # number of unique combinations of the parents realizations
    result <- 1
    for (j in 1:q.i) {
        a <- 1
        for (k in V.i){a <- a*factorial(N.tensor[j,as.character(k)])}
        result <- result*a*factorial(r.i-1)/factorial(N.ij[j]+r.i-1)
    }
    return(result)
}
```

Figure 2: Auxuliary functions.

Our Implementation - Big Datasets

Problem: The probability is built from factorials.

$$\mathbb{P}(B_S, D) = c \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}! = c \prod_{i=1}^{n} f(i, \pi_i)$$

Our Implementation - Big Datasets

Problem: The probability is built from factorials.

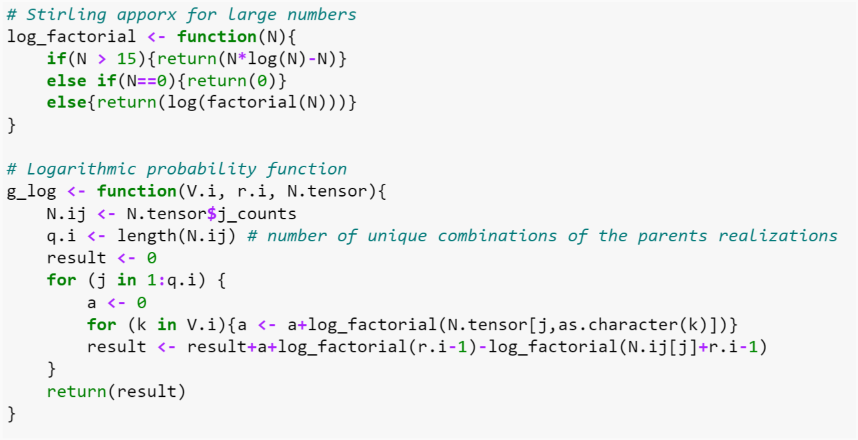
$$\mathbb{P}(B_S, D) = c \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}! = c \prod_{i=1}^{n} f(i, \pi_i)$$

Solution: Use the logarithm of the probability and apply Stirling's approximation.

$$\log f(i, \pi_i) = \sum_{j=1}^{q_i} \log\left[(r_i - 1)!\right] - \log\left[(N_{ij} + r_i - 1)!\right] + \sum_{k=1}^{r_i} \log\left[N_{ijk}!\right]$$

$$\log N! \approx N \log N - N$$

Our Implementation - Big Datasets



```r
# Stirling apporx for large numbers
log_factorial <- function(N){
    if(N > 15){return(N*log(N)-N)}
    else if(N==0){return(0)}
    else{return(log(factorial(N)))}
}

# Logarithmic probability function
g_log <- function(V.i, r.i, N.tensor){
    N.ij <- N.tensor$j_counts
    q.i <- length(N.ij) # number of unique combinations of the parents realizations
    result <- 0
    for (j in 1:q.i) {
        a <- 0
        for (k in V.i){a <- a+log_factorial(N.tensor[j,as.character(k)])}
        result <- result+a+log_factorial(r.i-1)-log_factorial(N.ij[j]+r.i-1)
    }
    return(result)
}
```

Figure 3: Logarithmic implementation functions.

Results - 3-node dataset

x1	x2	x3
1	0	0
1	1	1
0	0	1
1	1	1
0	0	0
0	1	1
1	1	1
0	0	0
1	1	1
0	0	0

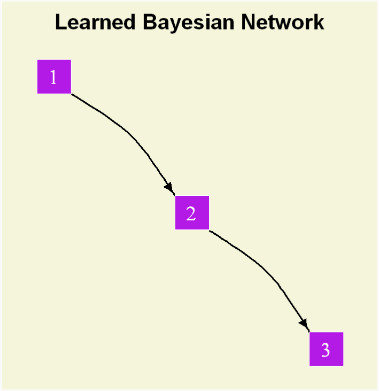
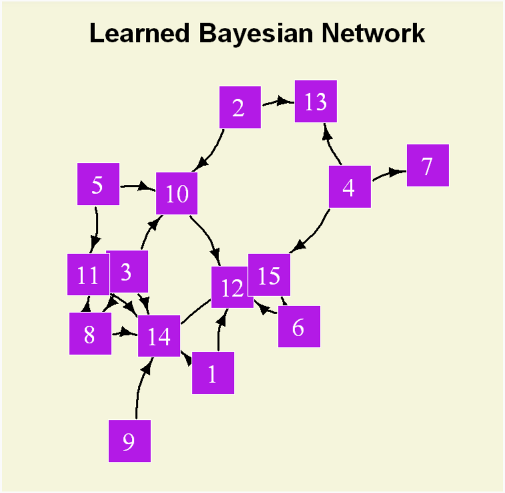


Figure 4: Dummy Dataset of with 3 nodes

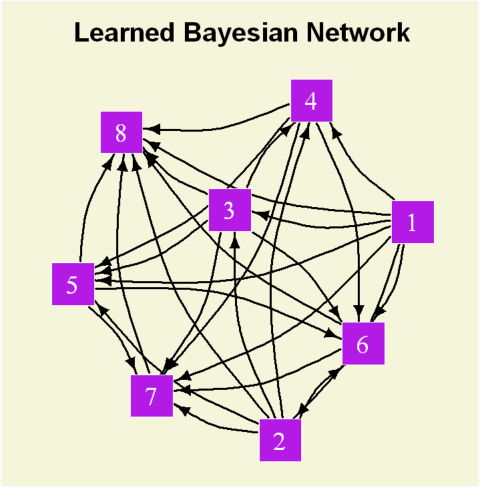
Results - 15-node dataset

Binary dataset with 15 features and 50 samples. The fixed maximum number of parents is 10.



Results - 8-node large dataset

Binary dataset with 8 features and 600 samples. The logarithm implementation is used. The fixed maximum number of parents is 7.



Results - Ordering difference

Dataset with 15 features and 50 samples. The fixed maximum number of parents is 10.

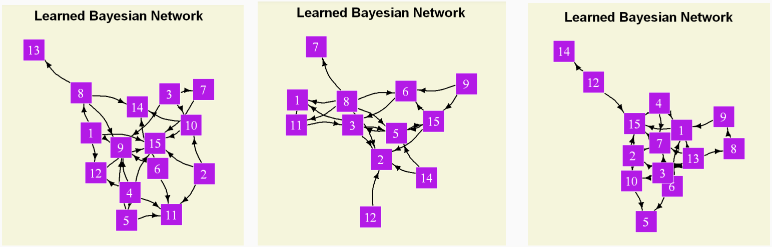


Figure 7: K2 algorithm tested in the same dataset with different variable orderings.

Results - Real World Dataset

Wisconsin Breast Cancer diagnosis dataset with 10 features and 683 samples. The logarithm implementation is used. The fixed maximum number of parents is 10.

- Clump Thickness (1 - 10)
- Uniformity of Cell Size (1 - 10)
- Uniformity of Cell Shape (1 - 10)
- Marginal Adhesion (1 - 10)
- Single Epithelial Cell Size (1 - 10)
- Bare Nuclei (1 - 10)
- Bland Chromatin (1 - 10)
- Normal Nucleoli (1 - 10)
- Mitoses (1 - 10)
- Benign (2) or Malignant (4)

Results - Real World Dataset

Wisconsin Breast Cancer diagnosis dataset with 10 features and 683 samples. The logarithm implementation is used. The fixed maximum number of parents is 10.

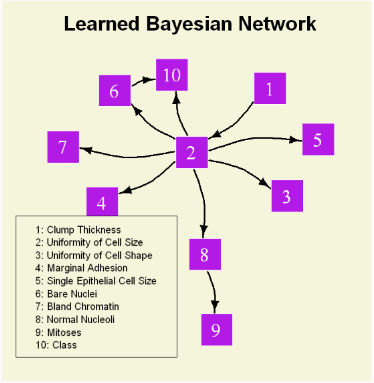


Figure 8: Wisconsin Breast Cancer Dataset's learnt BN.

BNStruct library

The *bnstruct* package provides objects and methods for learning the structure and parameters of bayesian networks in different situations:

- datasets with missing entries
- modeling of evolving systems
- belief propagation
- ...

We use the *BNDataset* and *learn.network* functions to provide some insight on how this library can work to solve Directed Acyclic Graphs.

BNStruct library

The *learn.network* function accepts as input the algorithm to use in order to learn the Bayesian Network structure. The **K2 algorithm** is not natively implemented but has to be included with a separate CRAN package.

The native possibilities are:

- sm (Silander-Myllymaki)
- mmpc (Max-Min Parent-and-Children)
- mmhc (Max-Min Hill Climbing, default)
- hc (Hill Climbing)
- sem (Structural Expectation Maximization)

Results - BNStruct library

Making use of the built-in BN datasets, we learn the data with the Silander-Myllymaki algorithm.

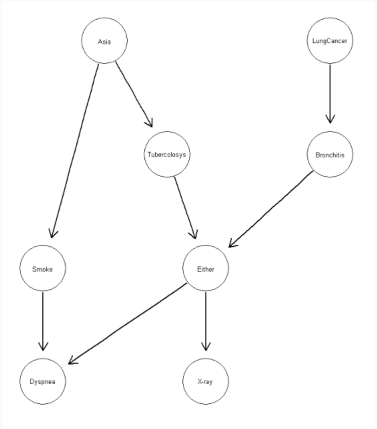


Figure 9: Asia Dataset with 8 nodes

Results - BNStruct library

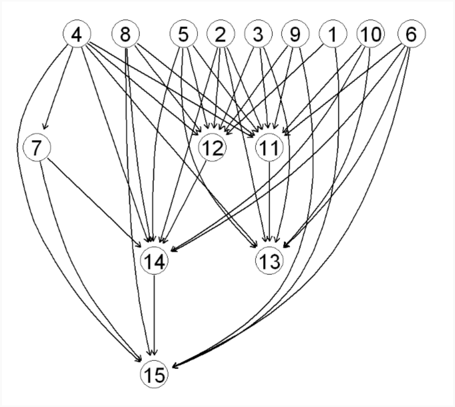


Figure 10: Dummy Dataset with 15 nodes