Knowledge-Based Systems 278 (2023) 110882

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Sensitivity and robustness analysis in Bayesian networks with the bnmonitor R package

Manuele Leonelli [a,*], Ramsiya Ramanathan [b], Rachel L. Wilkerson [c]

[a] School of Science and Technology, IE University, Madrid, Spain
[b] Dipartimento di Scienze Statistiche, Università di Bologna, Bologna, Italy
[c] Tesserwell, LLC, United States of America

ARTICLE INFO

Article history:
Received 26 July 2021
Received in revised form 31 July 2023
Accepted 2 August 2023
Available online 9 August 2023

Keywords:
Bayesian networks
Model-checking
Probabilistic graphical models
R package
Sensitivity analysis

ABSTRACT

Bayesian networks are a class of models that are widely used for the diagnosis, prediction, and risk assessment of complex operational systems. Multiple approaches, as well as implemented software, now guide their construction via learning from data or expert elicitation. However, current software only includes minimal functionalities to explore the assumptions, quality of fit, and sensitivity to learned parameters of a constructed Bayesian network. Here, we illustrate the usage of the bnmonitor R package: the first comprehensive software for model-checking of a Bayesian network. An applied data analysis using bnmonitor is carried out over a medical dataset to illustrate the use of its wide array of functions.

1. Introduction

Investigating the validity of the assumptions of a statistical model and the quality of its predictions is a fundamental step of any real-world applied analysis to ensure that the conclusions drawn from the model are reliable and credible [1,2]. Depending on the research community or the model used, this analysis step may take different names, such as model validation, model-checking, sensitivity, or robustness analysis. Henceforth, the term model-checking refers to any activity that investigates the "ability of a model to describe the system that it is intended to describe both in the output and in the mechanism by which that output is generated" [3].

The focus of this paper is on the model class of Bayesian networks (BNs) (first introduced by Pearl [4]), the most commonly used probabilistic graphical model, which gives an intuitive visualization of the dependence structure between variables as interest as well as an efficient platform to answer inferential queries. The array of domains where BNs are used is constantly increasing (e.g. [5–10]).

Although methods for model-checking in BNs have been developed ([3,11–14], among others), their use in practice has been limited (see [15–18], for some exceptions). Conversely, for other modeling approaches, model-checking is always carried out. For instance, in linear regression modeling, one always checks the distribution of the residuals to assess if the model's assumptions are met. In Bayesian inference, posterior predictive checks compare the original dataset to one simulated from the predictive model distribution to assess the appropriateness of the model.

A possible reason behind the limited use of model-checking techniques in BNs may be the lack of widely available implemented methods in software. The `bnmonitor` R package has been recently developed to fill this gap and provide practitioners with a wide array of functions for BNs' model-checking. The R programming language was chosen for two reasons. First, because there are now a large number of packages to both learn BNs from data and to carry out inferential tasks, including `bnlearn` [19], `BayesNetBP` [20] and `gRain` [21]. Second, to take advantage of R's advanced graphical capabilities, by using methods from the `ggplot2` [22] and `qgraph` [23] packages.

Here we provide an overview of the capabilities of `bnmonitor` by carrying out an applied BN analysis over a medical dataset. Although `bnmonitor` requires as input BNs as `bnlearn` objects, the latter package has a wide array of conversion functions from formats used in most other software. Therefore, `bnmonitor` can be used with models developed in any other programming language or commercial software. Before carrying out the analysis, an introduction to both BNs and the model-checking techniques implemented in `bnmonitor` is given. Although `bnmonitor` can consider BNs with either discrete variables or continuous ones under the assumption that the joint distribution is multivariate Gaussian, for this paper, only the discrete case is considered since this is the most common in practical applications (see [24], for details on the continuous case).

* Corresponding author.
E-mail address: manuele.leonelli@ie.edu (M. Leonelli).

https://doi.org/10.1016/j.knosys.2023.110882

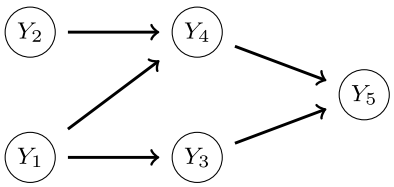


Fig. 1. A simple DAG of a BN over a random vector $\boldsymbol{Y} = (Y_1, Y_2, Y_3, Y_4, Y_5)$.

2. Bayesian networks and their use

Let $[n] = \{1, 2, \ldots, n\}$. Consider a random vector $\boldsymbol{Y} = (Y_1, \ldots, Y_n)$ of interest, where Y_i takes values in a discrete space $\mathbb{Y}_i$, $i \in [n]$. For a subset $A \subset [n]$, we denote $\boldsymbol{Y}_A = (Y_i)_{i \in A}$ and $\mathbb{Y}_A = \times_{i \in A} \mathbb{Y}_i$. For three random vectors $\boldsymbol{Y}_A$, $\boldsymbol{Y}_B$ and $\boldsymbol{Y}_C$, where $A, B, C \subset [n]$, $\boldsymbol{Y}_A$ is said to be conditionally independent of $\boldsymbol{Y}_B$ given $\boldsymbol{Y}_C$ and write $\boldsymbol{Y}_A \perp\!\!\!\perp \boldsymbol{Y}_B | \boldsymbol{Y}_C$ if

$$P(\boldsymbol{Y}_A = \boldsymbol{y}_A | \boldsymbol{Y}_B = \boldsymbol{y}_B, \boldsymbol{Y}_C = \boldsymbol{y}_C) = P(\boldsymbol{Y}_A = \boldsymbol{y}_A | \boldsymbol{Y}_C = \boldsymbol{y}_C), \tag{1}$$

for all $\boldsymbol{y}_A \in \mathbb{Y}_A$, $\boldsymbol{y}_B \in \mathbb{Y}_B$ and $\boldsymbol{y}_C \in \mathbb{Y}_C$, where P denotes a probability mass function. In the following, as a shorthand, we write $P(\boldsymbol{Y}_A = \boldsymbol{y}_A | \boldsymbol{Y}_B = \boldsymbol{y}_B)$ as $P(\boldsymbol{y}_A | \boldsymbol{y}_B)$, for any sets $A, B \subseteq [n]$.

2.1. Bayesian networks: definition

A BN gives a visual representation of conditional independence by means of a directed acyclic graph (DAG). More formally, a BN over a vector of discrete variables $\boldsymbol{Y}$ is a pair $\mathcal{B} = (G, \boldsymbol{\theta})$, where: G is a DAG whose vertices are the variables in $\boldsymbol{Y}$, and, roughly speaking, edges represent direct probabilistic dependencies between the variables; $\boldsymbol{\theta}$ is a vector of the (conditional) probabilities $P(y_i | \boldsymbol{y}_{\Pi_i})$ for each $y_i \in \mathbb{Y}_i$ given a, possibly empty, vector value $\boldsymbol{y}_{\Pi_i} \in \mathbb{Y}_{\Pi_i}$ of the parents of Y_i in G. The BN $\mathcal{B}$ represents the joint distribution over $\boldsymbol{Y}$ factorized according to the structure of G:

$$P(\boldsymbol{y}) = \prod_{i \in [n]} P(y_i | \boldsymbol{y}_{\Pi_i})$$

A BN $\mathcal{B}$ encodes $n - 1$ conditional independences of the form

$$Y_i \perp\!\!\!\perp \boldsymbol{Y}_{\Lambda_i} | \boldsymbol{Y}_{\Pi_i},$$

where $\boldsymbol{Y}_{\Lambda_i}$ are the non-descendants of Y_i in G.

Fig. 1 reports a DAG over five discrete variables $(Y_1, \ldots, Y_5)$. A BN $\mathcal{B}$ with this DAG embeds the conditional independences $Y_2 \perp\!\!\!\perp Y_1$, $Y_3 \perp\!\!\!\perp Y_2 | Y_1$, $Y_4 \perp\!\!\!\perp Y_3 | Y_1, Y_2$ and $Y_5 \perp\!\!\!\perp Y_1, Y_2 | Y_3, Y_4$. The definition of the BN is completed by the specification of $\boldsymbol{\theta}$, i.e. the probabilities P of each variable conditional on the possible values of the parents. The mass function of $\mathcal{B}$ factorizes as

$$P(\boldsymbol{y}) = P(y_5 | y_4, y_3) P(y_4 | y_2, y_1) P(y_3 | y_1) P(y_2) P(y_1)$$

The DAG associated with a BN provides an intuitive overview of the relationships existing between variables of interest, as well as a framework to assess if any generic conditional independence holds for three arbitrary sets of variables (see Eq. (1)) via the so-called d-separation criterion (see e.g. [4]). Since the exact propagation of probabilities is am NP-complete problem, algorithms for the computation of posterior probability queries that take advantage of the underlying DAG have been devised and implemented in software.

2.2. Constructing Bayesian networks

The definition of a BN $\mathcal{B}$ consists of the selection of the underlying DAG G and, given that DAG, of the associated probabilities $\boldsymbol{\theta}$. In practice, these two steps are not independently carried out since, for instance, a very complex structure G leads to an enormous number of probabilities $\boldsymbol{\theta}$ to be defined, which may be unfeasible to assess faithfully. Both steps can be carried out using two approaches: (i) *expert-elicitation*: when experts and stakeholders are guided to come up with their definition of G and $\boldsymbol{\theta}$; (ii) *data-driven*: when data and appropriate algorithms are used to choose an optimal combination of G and $\boldsymbol{\theta}$. Often in practice, these two approaches are used in conjunction: for instance, the DAG G may have to respect some constraints determined by experts, and data is used to find the best G respecting them; alternatively, data may be available only for some variables, and therefore the associated probabilities are learned via data, while the remaining ones are expert-elicited.

There is a vast literature on expert elicitation and aggregation techniques (e.g. [25–27]). In particular, protocols have been defined to guide the appropriate definition of a BN using expert judgments [28–30]. Eliciting the dependence structure, namely the DAG G, is particularly challenging, but there are now also guidelines for this task [31,32]. Furthermore, efficient methods to integrate expert opinions into commonly used data-driven algorithms have been defined [33–35].

The literature on data-driven learning of BNs is now well-established, although novel algorithms continue to appear to account for more flexible structures and to improve speed and accuracy (e.g. [36,37]). The selection of the DAG G using data is usually called *structural learning*, and there are two main classes of algorithms for this task: (i) *constraint-based* algorithms select the DAG by checking if the associated conditional independences hold (most notably, the PC algorithm of Spirtes et al. [38]); (ii) *score-based* algorithms heuristically explore the space of all possible DAGs to find the one optimizing some chosen criterion. For parameter estimation, i.e. the estimation of $\boldsymbol{\theta}$ using data, either a frequentist (e.g. maximum likelihood) or a Bayesian approach can be used. If score-based structural learning is used, for the frequentist approach, it is then common to guide the search minimizing the BIC of the model, while, for the Bayesian approach, the search is based on the marginal likelihood reviewed in Section 2.3 below.

BNs define an overall probability distribution for the vector Y of interest and therefore are an instance of *generative modeling*, in contrast to discriminative modeling used for classification purposes. BNs can also be used for classification tasks, and such BNs usually have a constrained structure of their DAG. These are called BN classifiers [39,40]. The `bnmonitor` R package can be equally used to investigate BN classifiers. However, the implemented methods are not explicitly tailored for classification problems (as, for instance, the methods of Bolt and van der Gaag [41]).

2.3. Learning Bayesian networks: the Bayesian approach

Some of the methods for BN model-checking we introduce in the next section are based on the Bayesian approach for learning the probabilities of a BN, although they do not require the BN to be constructed in this way. For this reason, the approach is reviewed next.

Consider a BN over a random vector $\boldsymbol{Y} = (Y_1, \ldots, Y_n)$, whose sample space is $\mathbb{Y} = \times_{i=1}^n \mathbb{Y}_i$ and $\mathbb{Y}_i$ is the sample space of Y_i. For ease of notation, we denote a generic probability from the BN as $\theta_{i\boldsymbol{z}k} = P(Y_i = k | \boldsymbol{Y}_{\Pi_i} = \boldsymbol{z})$ where $k \in \mathbb{Y}_i$ and $\boldsymbol{z} \in \mathbb{Y}_{\Pi_i}$ (hence the use of $\boldsymbol{\theta}$ to denote the vector of all probabilities). Thus, i indexes the variable, $\boldsymbol{z}$ is the parent configuration and k is the value taken by Y_i.

Suppose a dataset $\mathcal{D}$ is observed and let $N_{i\boldsymbol{z}k}$ be the number of observations in $\mathcal{D}$ such that $Y_i = k$ and $\boldsymbol{Y}_{\Pi_i} = \boldsymbol{z}$. The likelihood of the BN with DAG G is

$$P(\mathcal{D} | \boldsymbol{\theta}, G) \propto \prod_{i \in [n]} \prod_{\boldsymbol{z} \in \mathbb{Y}_{\Pi_i}} \prod_{k \in \mathbb{Y}_i} \theta_{i\boldsymbol{z}k}^{N_{i\boldsymbol{z}k}}.$$

Within the Bayesian approach, the model definition is completed by the choice of a prior distribution for $\boldsymbol{\theta}$. A Dirichlet prior $\mathcal{D}(\alpha_{izk})_{k\in\mathbb{Y}_i}$, $\alpha_{izk} > 0$, is independently assigned to each $\boldsymbol{\theta}_{iz} = (\theta_{izk})_{k\in\mathbb{Y}_i}$, the mass function of Y_i conditional on a specific value of its parents. The overall prior distribution for $\boldsymbol{\theta}$ can be written as

$$p(\boldsymbol{\theta}|G) = \prod_{i\in[n]}\prod_{\boldsymbol{z}\in\mathbb{Y}_{\Pi_i}} \frac{\Gamma\left(\sum_{k\in\mathbb{Y}_i}\alpha_{izk}\right)}{\prod_{k\in\mathbb{Y}_i}\Gamma(\alpha_{izk})}\prod_{k\in\mathbb{Y}_i}\theta_{izk}^{\alpha_{izk}-1},$$

where $\Gamma(\cdot)$ denotes the Gamma function. The values α_{izk} are henceforth assumed equal to $\#\mathbb{Y}_i$, i.e. the number of elements in the sample space of Y_i (see [42], for a justification of this).

Results first appeared in [43] guarantee that the posterior distribution $p(\boldsymbol{\theta}|\mathcal{D}, G)$ can be written as

$$p(\boldsymbol{\theta}|\mathcal{D}, G) = c\prod_{i\in[n]}\prod_{\boldsymbol{z}\in\mathbb{Y}_{\Pi_i}} \frac{\Gamma\left(\sum_{k\in\mathbb{Y}_i}\alpha_{izk}+N_{izk}\right)}{\prod_{k\in\mathbb{Y}_i}\Gamma(\alpha_{izk}+N_{izk})}\prod_{k\in\mathbb{Y}_i}\theta_{izk}^{\alpha_{izk}+N_{izk}-1},$$

where c is a normalizing constant. The above expression can be seen as the product of independent Dirichlet distributions for $\boldsymbol{\theta}_{ij}$ $\mathcal{D}(\alpha_{izk}+N_{izk})_{k\in\mathbb{Y}_i}$.

The choice of the best DAG is based on the maximization of the so-called *marginal likelihood* $p(\mathcal{D}|G)$ which can be derived from the previous results (see [43]) as

$$P(\mathcal{D}|G) = \prod_{i\in[n]}\prod_{\boldsymbol{z}\in\mathbb{Y}_{\Pi_i}} \frac{\Gamma(\sum_{k\in\mathbb{Y}_i}\alpha_{izk})}{\Gamma(\sum_{k\in\mathbb{Y}_i}(\alpha_{izk}+N_{izk}))}\prod_{k\in\mathbb{Y}_i}\frac{\Gamma(\alpha_{izk}+N_{izk})}{\Gamma(\alpha_{izk})}. \quad (2)$$

The space of all possible DAGs is searched using heuristics driven by the maximization of Eq. (2): for instance, hill-climbing or tabu search, as implemented in the widely-used `bnlearn` R package [19]. These searches are often coupled with resampling or cross-validation techniques to prevent overfitting [44].

Notice that the approach reviewed above can be distinct from Bayesian structural learning, which also defines a prior distribution over the space of all possible DAGs. Using approximating MCMC algorithms, each possible DAG structure is assigned a posterior probability using data (e.g. [45]).

3. Model-checking in Bayesian networks

The term model-checking refers to the set of processes and activities intended to check the representativity of the model and the accuracy of the inference results. Although many techniques could be used for BNs (see the discussion at the end of the paper), the bnmonitor package implements two broad methodologies that, following the terminology commonly used in BNs, we call *robustness to data* and *sensitivity analysis*.

The term robustness to data refers to methods that test the compatibility between data and an assumed model. For expert-elicited BNs, it is sometimes the case that data becomes available after the model definition. As strongly advocated by Box [46], it is fundamental to critically examine the assumed model in light of the data available and, if required, update it. For BNs derived via structural learning, one must check data quality and whether the data support the BN assumptions. For instance, the data may suggest that the symmetric conditional independence assumption of BNs is not tenable (see Section 4).

Sensitivity analysis refers to the quantification that minor variations in the model inputs have on outputs of interest. For expert-elicited BNs, this is fundamental: during the elicitation process, it is often the case that some posterior probabilities appear to be unreasonable to the experts, although they are a coherent consequence of their beliefs. Sensitivity analysis allows for the identification of critical probabilities, whose values may need to be reviewed and elicited with higher precision. For data-driven BNs, it is often helpful to check that learned probabilities are sound to stakeholders and relevant experts since the data used may have needed to be of better quality or represent the actual population of interest.

All these techniques are reviewed next and illustrated in a practical application in Section 4.

3.1. Robustness to data

Suppose a dataset $\mathcal{D} = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_m)$ of m observations of the random vector $\boldsymbol{Y}$ has been collected, where $\boldsymbol{y}_j = (y_{j1}, \ldots, y_{jn})$ and y_{ji} denotes the jth observation for the ith variable. We further let $\boldsymbol{y}_{[j-1]} = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{j-1})$ and $\boldsymbol{y}_j^{-i} = (y_{j1}, \ldots, y_{j(i-1)}, y_{j(i+1)}, \ldots, y_{jn})$. The dataset $\mathcal{D}$ could be the one used for learning the BN, in which case the focus would be on model fit. Conversely, $\mathcal{D}$ could be a test dataset not used for estimation, and therefore the focus would be on out-of-sample prediction. This second approach is often used to avoid overfitting [47].

Most methods reviewed next follow the *prequential framework* of Dawid [48], in basing criticism of a model on the quality of the predictions it sequentially makes. In a Bayesian setup, the probability of observing a new specific observation $\boldsymbol{y}_p$ given a dataset $\mathcal{D}$ is summarized by the *predictive mass function*

$$P(\boldsymbol{y}_p|\mathcal{D}) = \int P(\boldsymbol{y}_p|\boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D})d\boldsymbol{\theta}, \quad (3)$$

which is generally computationally expensive. However, Eq. (4) below shows that it can be derived in closed form, similarly to the marginal likelihood of Eq. (2).

Let $P_j(\cdot)$ denote the predictive mass function given the first $j-1$ observations, i.e. $P(\cdot|\boldsymbol{y}_{[j-1]})$. The observation $\boldsymbol{y}_j$ follows $\boldsymbol{y}_{[j-1]}$ in $\mathcal{D}$ and the logarithmic score

$$S_j = -\log P_j(\boldsymbol{y}_j),$$

measures the level of surprise of observing it after $\boldsymbol{y}_{[j-1]}$. To understand this, suppose that the observation $\boldsymbol{y}_j$ was predicted with probability 0.1. Then $S_j = -\log(0.1) = 2.3$. Conversely, an observation that was predicted to happen with probability 0.9 would carry a score of 0.1. Thus, the less likely an observation is predicted to occur, the more "surprising" it is if it does occur.

By cumulating the logarithmic score over all observations, an overall level of surprise is written as

$$\sum_{j=1}^{m} S_j = \sum_{j=1}^{m} -\log P(\boldsymbol{y}_j|\boldsymbol{y}_{[j-1]}) = -\log\prod_{j=1}^{m} P(\boldsymbol{y}_j|\boldsymbol{y}_{[j-1]}) = -\log P(\mathcal{D}). \quad (4)$$

Eq. (4) is simply the negative marginal log-likelihood which can be computed in closed form using Eq. (2). Thus, the marginal likelihood is retrieved by summing over the predictive densities of Eq. (3). Notice that, although computed from predictive distributions, Eq. (4) is invariant to the order of the observations in $\mathcal{D}$ (a property entertained only by the logarithmic score and not by other scoring rules, see e.g. [49]).

The statistics, or *monitors* (following the terminology of Cowell et al. [50]), defined next use the logarithmic score to measure the agreement between the model and the data.

3.1.1. Global monitor

The *global monitor* is simply the negative of the marginal log-likelihood, i.e.

$$\mathrm{GM}(\mathcal{D}, \mathcal{B}) = -\log P(\mathcal{D}) \quad (5)$$

Such a monitor is useful when there are two or more hypothesized network structures (for instance, defined by experts or

output of two different search methods) and interest is in assessing which DAG better represents the data. The computation of the global monitor is immediate since it is based on the simple formula in Eq. (2).

3.1.2. Node monitors

While the global monitor provides an overview of the whole BN, the *node monitor* returns the contribution of each node to the negative of the marginal log-likelihood, i.e. the global monitor. This is defined as

$$\mathrm{NM}_i(\mathcal{D}, \mathcal{B}) = -\sum_{\boldsymbol{z}\in\mathbb{Y}_{\Pi_i}} \log \frac{\Gamma(\sum_{k\in\mathbb{Y}_i}\alpha_{\boldsymbol{z}k})}{\Gamma(\sum_{k\in\mathbb{Y}_i}(\alpha_{\boldsymbol{z}k}+N_{\boldsymbol{z}k}))} - \sum_{k\in\mathbb{Y}_i}\log\frac{\Gamma(\alpha_{\boldsymbol{z}k}+N_{\boldsymbol{z}k})}{\Gamma(\alpha_{\boldsymbol{z}k})}, \tag{6}$$

and it straightforwardly follows from Eq. (2) that

$$\mathrm{GM}(\mathcal{D}, \mathcal{B}) = \sum_{i\in[n]} \mathrm{NM}_i(\mathcal{D}, \mathcal{B}).$$

Thus, the computation of the node monitors is again immediate since it is based on the simple algebraic formula in Eq. (6). Vertices with a higher value of node monitor are vertices which, if the BN were to be learned with a score-based structural learning algorithm, would have a high impact on the learning process. As already noticed, both node and global monitors are not actually affected by the ordering of the observations.

3.1.3. Sequential node monitors

The next class of monitors assesses the quality of the predictions made by sequentially considering each observation in $\mathcal{D}$. First, define

$$S_{ji} = -\log P_j(y_{ji}) \tag{7}$$

which is the logarithmic score of the predictive mass function for observing $Y_i = y_{ji}$ after $\boldsymbol{y}_{[j-1]}$. Thus,

$$\mathrm{GM}(\mathcal{D}, \mathcal{B}) = \sum_{i\in[n]}\sum_{j\in[m]} S_{ji}.$$

Next, the logarithmic score S_{ji} is normalized. Define

$$E_{ji} = -\sum_{y_i\in\mathbb{Y}_i} P_j(y_i)\log P_j(y_i), \qquad V_{ji} = \sum_{y_i\in\mathbb{Y}_i} P_j(y_i)\log P_j(y_i)^2 - E_{ji}^2, \tag{8}$$

where E_{ji} and V_{ji} are the expectation and the variance of the logarithmic score in Eq. (7). The *sequential marginal node monitor* for Y_i after $\boldsymbol{y}_{[j-1]}$ is defined as

$$\mathrm{SMND}_{ji}(\mathcal{D}, \mathcal{B}) = \frac{\sum_{k=1}^{j} S_{ki} - \sum_{k=1}^{j} E_{ki}}{\sqrt{\sum_{k=1}^{j} V_{ki}}}. \tag{9}$$

Seillier-Moiseiwitsch and Dawid [51] demonstrated that SMND_{ji} asymptotically, for $j \to \infty$, follows a standard Normal distribution under the null hypothesis:

$H_0 : \mathcal{D}$ is generated by $\mathcal{B}$.

Therefore, values of SMND_{ji} over 1.96 in absolute value would, in a standard frequentist test of hypothesis setting, lead to rejecting such a null hypothesis at the 0.05 significance level. Consequently, values of SMND_{ji} over 1.96 in absolute value may indicate poor model fit since it leads to rejecting the hypothesis that the chosen BN generates the data.

While global and node monitors are independent of the ordering of $\mathcal{D}$, SMND_{ji} is not. Such a monitor has a much more natural interpretation when there is a natural ordering of the observed data (for instance, a temporal one). When a temporal component is not immediately obvious, ordering the data according to some covariate of the observations may be helpful. For instance, modeling healthcare outcomes might benefit from ordering the data according to the time each observation spent in the hospital. The sequential marginal node monitor is well suited to detect when the model is no longer a good fit to the data for values of that covariate. This is the strategy we follow in Section 4 below. However, even if no ordering can be found between the observations, SMND_{ji} can help detect nodes that do not fit the data well overall. A possible method to assess if the ordering of $\mathcal{D}$ affects SMND_{ji} is to compute such monitor using different permutations of the observations.

The *sequential conditional node monitor* is defined similarly to SMND_{ji}, but starting from the logarithmic score

$$S^*_{ji} = -\log P(y_{ji}|\boldsymbol{y}^{-i}_{[j]}),$$

where the predictive distribution is also conditional on $\boldsymbol{y}_j$, except for the value it takes for the variable Y_i, i.e. y_{ji}. The sequential conditional node monitor, that we henceforth call SCND_{ji}, is then defined by straightforwardly adapting Eqs. (8) and (9) (see [50], for details). Its interpretation is the same as the one of SMND_{ji}.

Computationally, sequential node monitors require the computation of a node monitor m times in datasets of increasing size, where m is the number of observations in $\mathcal{D}$. For each of the m derivations of the node monitor, they further require the computation of the expectation and variance as in Eq. (8). Notice that the sequential conditional node monitor further requires an extra marginalization operation over the BN at each of the m derivations.

3.1.4. Parent–child monitors

The sequential monitors SMND_{ji} and SCND_{ji} inform about the lack of fit of a particular node to data. Once a problematic node is identified, it might be interesting to assess which conditional mass functions associated with it are causing the lack of fit. This may be due to the predictions from a specific configuration of the parents only. The *sequential parent–child monitor* assesses the quality of the predictions from the model when only data associated with a specific configuration of the parents of a node is retained. Let $\mathcal{D}_{\pi_i}$ be the subset of the dataset including only the observations of $\mathcal{D}$ such that $\boldsymbol{Y}_{\Pi_i} = \pi_i$, for $\pi_i \in \mathbb{Y}_{\Pi_i}$ and let $P_j^{\pi_i}(\cdot)$ be the predictive mass function after the first $j-1$ observations of $\mathcal{D}_{\pi_i}$ have been processed. Similarly to sequential node monitors, define

$$E^{\pi_i}_{ji} = -\sum_{y_i\in\mathbb{Y}_i} P_j^{\pi_i}(y_i)\log P_j^{\pi_i}(y_i), \qquad V^{\pi_i}_{ji} = \sum_{y_i\in\mathbb{Y}_i} P_j^{\pi_i}(y_i)\log P_j^{\pi_i}(y_i)^2 - (E^{\pi_i}_{ji})^2$$

The sequential parent–child monitor for the vertex Y_i and parent configuration $\pi_i \in \mathbb{Y}_{\Pi_i}$ is defined as

$$\mathrm{SPCM}^{\pi_i}_{ji}(\mathcal{D}_{\pi_i}, \mathcal{B}) = \frac{-\sum_{k=1}^{j}\log P_k^{\pi_i}(y_{ki}^{\pi_i}) - \sum_{k=1}^{j} E^{\pi_i}_{ki}}{\sqrt{\sum_{k=1}^{j} V^{\pi_i}_{ki}}}, \tag{10}$$

where $y^{\pi_i}_{ki}$ is the kth observation for the variable Y_i in $\mathcal{D}_{\pi_i}$.

The interpretation of sequential parent–child monitors is the same as for sequential node monitors, where values above 1.96 in absolute value should be viewed with suspicion. Of course, unless the original dataset is vast, the conclusions from such monitors may be less reliable since they are based on a smaller set of observations, and convergence to the standard Normal distribution may be questionable. Suppose the parent–child monitors indicate a poor fit for only one specific configuration of parent values. In that case, one could still use the BN when computing quantities specific to the other configurations of the parents.

Furthermore, as shown by Wilkerson and Smith [52], values above 1.96 in absolute value may indicate that the assumption of symmetric conditional independences made by BNs may be too restrictive and that the data may, on the other hand, support context-specific conditional independences. Then the modeler could choose to use an alternative, more flexible model instead, for instance, a context-specific BN [53] or a staged tree [54].

Notice that the complexity of parent–child monitors is less than that of sequential node monitors since in general $\mathcal{D}_{\pi_i}$ is a subset of $\mathcal{D}$.

3.1.5. Influential observations

The influence of the jth observation of $\mathcal{D}$ is defined as

$$\mathrm{I}(\boldsymbol{y}_j) = |\log P(\mathcal{D}|G) - \log P(\mathcal{D}_{-j}|G)|, \tag{11}$$

where $\mathcal{D}_{-j}$ is the dataset without the jth observation. High influence values denote observations that strongly contribute to the model's marginal likelihood. This implies, for instance, that if the BN were learned using a score-based structural search algorithm, influential observations strongly drove the choice of the DAG G.

3.2. Sensitivity analysis in Bayesian networks

While the previous statistics are specifically designed for assessing the agreement between data and the chosen model, the next set of tools for model-checking investigates how perturbations of the inputs of a model affect outputs of interest. In the literature on BNs, such an investigation is usually called *sensitivity analysis* [11,12]. For BNs, the inputs of the model are the conditional probabilities $P(Y_i = k|\boldsymbol{Y}_{\Pi_i} = \boldsymbol{z})$, where $k \in \mathbb{Y}_i$ and $\boldsymbol{z} \in \mathbb{Y}_{\Pi_i}$. As in Section 2.3, these probabilities are sometimes denoted as $\theta_{i\boldsymbol{z}k}$, for ease of notation.

Let $O, E \subset [n]$ be the index of the output and evidence variables, respectively. The output variable is one whose probability distribution is of interest, possibly conditional on a specific value of $\boldsymbol{y}_E$ that may be observed. The interest is in studying how $P(y_O|\boldsymbol{y}_E)$ varies in terms of one input parameter $\theta_{i\boldsymbol{z}k}$. More specifically, the probability $P(y_O|\boldsymbol{y}_E)$ seen as a function of $\theta_{i\boldsymbol{z}k}$ is called a *sensitivity function* and can be computed as [12]:

$$\mathrm{S}(\theta_{\boldsymbol{z}k}) = \frac{a + b\theta_{i\boldsymbol{z}k}}{c + d\theta_{i\boldsymbol{z}k}}, \tag{12}$$

where $a, b, c, d \in [0, 1]$ and depend on the BN, the output probability and the input parameter chosen. If $E = \emptyset$, then the denominator of Eq. (12) is equal to one.

Sensitivity functions are used to assess which input probabilities of the BN have a significant impact on output probabilities of interest. For a data-driven learned BN, these functions could indicate that the collection of more data for specific configurations of some nodes is needed to ensure that output probabilities are correctly learned. For expert-elicited networks, sensitivity functions highlight inputs that may require extra effort or the consultation of additional experts for their accurate elicitation. Furthermore, these sensitivity functions are often shown during the elicitation process of a BN to ensure that the experts realize the implications of the inputs they are defining [29].

It is common to observe in practical expert-elicited BNs that some probabilities implied by those defined by experts, although formally computed using probability rules, need to be fine-tuned. When that happens, some of the input parameters need to be updated to ensure these probabilities take reasonable values to the experts. Consider a probability of interest $P(y_O|\boldsymbol{y}_E)$ and suppose it should equal some new value. We then want to identify changes in the input probabilities $\theta_{i\boldsymbol{z}k}$ to ensure such a change is met, i.e. we want to solve the equation

$$S(\theta_{i\boldsymbol{z}k}) = \text{new\_value}, \tag{13}$$

Table 1
List of statistics defined in Section 3 together with the name of the functions in the `bnmonitor` package.

Name	Equation name	Equation number	Function name
Global Monitor	GM	(4)	`global_monitor`
Node Monitor	NM_i	(5)	`node_monitor`
Sequential Marginal Node Monitor	SMND_{ji}	(8)	`seq_marg_monitor`
Sequential Conditional Node Monitor	SCND_{ji}	NA	`seq_cond_monitor`
Sequential Parent–Child Monitor	SPCM^{π}_{ji}	(9)	`seq_pa_ch_monitor`
Influential observations	I	(10)	`influential_obs`
Sensitivity function	S	(11)	`sensitivity`
Changes to meet constraint	NA	(12)	`sensquery`
CD distance	CD	(13)	`CD`

his exercise is also commonly done during elicitation to ensure the final BN fully reflects the experts' opinions [11,29]. Furthermore, the solution to this problem is often needed during a scenario analysis (see e.g. [55]) based on a BN model, where some probabilities are changed to represent a possible future scenario, and the consequences of such a change are assessed.

Last, assume a specific change of a parameter $\theta_{i\boldsymbol{z}k}$ has been identified, for instance, solving Eq. (13). Denote the probability mass function of the BN with this new input probability as P'. It is then helpful to assess the impact of such a local change on the overall distribution of the BN. This can be assessed by computing the distance between the original P and the new P'. Standard measures to do this are the well-known Kullback–Leibler divergence and the Jeffreys distance [56]. A distance designed explicitly for this task was introduced by Chan and Darwiche [57] and henceforth called *CD distance*. The CD distance between two probability mass functions P and P' is defined as

$$\mathrm{CD}(P, P') = \log \max_{\boldsymbol{y}\in\mathbb{Y}} \left(\frac{P(\boldsymbol{y})}{P'(\boldsymbol{y})}\right) - \log \min_{\boldsymbol{y}\in\mathbb{Y}} \left(\frac{P(\boldsymbol{y})}{P'(\boldsymbol{y})}\right). \tag{14}$$

All above mentioned distances/divergences are implemented in `bnmonitor` to quantify the overall effect of changes in the conditional probabilities of the BN. The computation of such distances is also helpful to rank all parameters $\theta_{i\boldsymbol{z}k}$ for which there is a solution to Eq. (13). Then the modeler could decide to change the one having the smallest effect on the overall BN distribution.

3.3. Implementation in `bnmonitor`

The above statistics, as well as others not described here, because beyond the scope of this paper, are implemented in the `bnmonitor` R package, which can be directly installed from CRAN. Currently, the package is at version 0.1.4 under a GPL-3 license. A website reporting the documentation, vignettes, news, and other information can be found at https://manueleleonelli.github.io/bnmonitor/. Some of the dependencies of `bnmonitor`, although all on CRAN, require packages on `Bioconductor` both directly and through the `gRbase` package, which depends on `RBGL`. All these required packages can be installed using the code:

```
install.packages("BiocManager")
BiocManager::install(c("graph", "Rgraphviz", "RBGL"))
```

Table 1 links the statistics introduced in this section to their function name in `bnmonitor`.

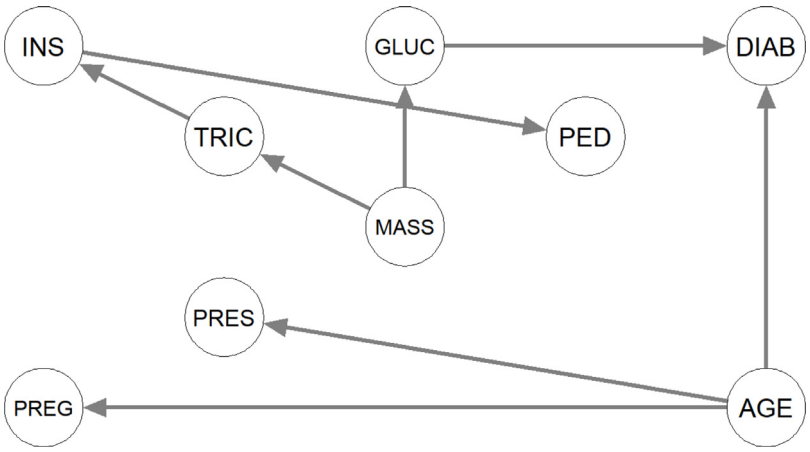


Fig. 2. Learned BN for the `diabetes` dataset using tabu with a BIC score (Tabu-BIC).

4. An applied analysis with `bnmonitor`

The capabilities of the `bnmonitor` R package and the required syntax are now illustrated by an extensive model investigation for a BN learned using the `diabetes` dataset bundled in the package. The code to replicate the analysis is available at https://github.com/manueleleonelli/bnmonitor_paper.

4.1. Data description

The `diabetes` dataset is a discretized version of the famous Pima Indian Diabetes dataset from the UCI machine learning repository reporting medical reports of Pima-Indian women at least 21 years of age.[1] Continuous variables were discretized into binary ones using the equal quantile method [58]. The resulting `diabetes` dataset has 768 observations and the following nine variables.

- `PREG`: number of times pregnant (low/high);
- `GLUC`: plasma glucose concentration (low/high);
- `PRES`: diastolic blood pressure (low/high);
- `TRIC`: triceps skin fold thickness (low/high);
- `INS`: 2-hour serum insulin (low/high);
- `MASS`: body mass index (low/high);
- `PED`: diabetes pedigree function (low/high);
- `AGE`: age (low/high);
- `DIAB`: test for diabetes (neg/pos).

The `DIAB` variable is an indicator (pos) for a positive test for diabetes between one and five years from the examination determining the other variables, or (neg) a negative test for diabetes five or more years later.

We select 400 observations at random for model estimation (`train` data), while the remaining observations are used for assessing the quality of the models (`test` data).

4.2. Learning a BN

To start the analysis, the `bnmonitor` and the `diabetes` dataset are loaded. All other required packages are automatically installed when `bnmonitor` is installed.

```
library("bnmonitor")
data(diabetes)
```

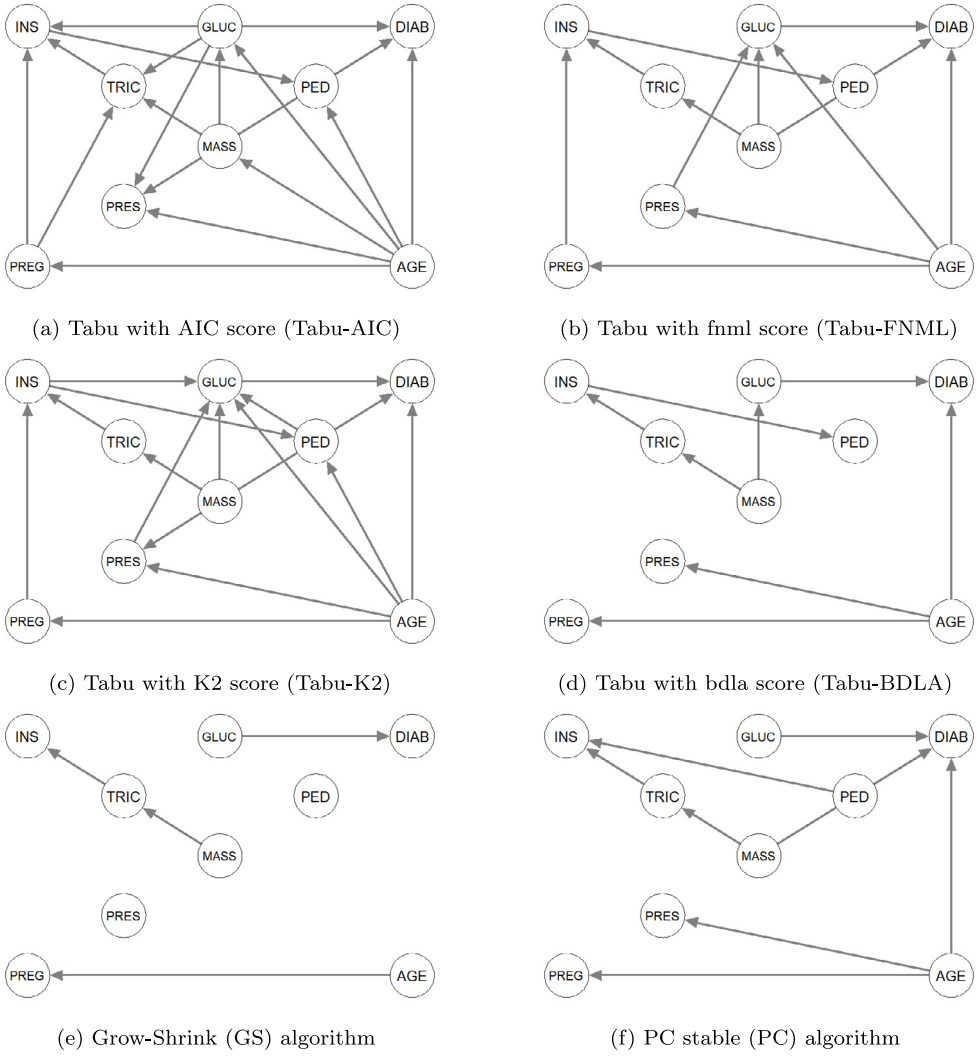
The DAG of a BN is learned for the `diabetes` train dataset using one thousand bootstrap replications as implemented in the `boot.strength` function of `bnlearn` in conjunction with various search routines and scoring rules. All search routines are used with their default inputs of `bnlearn`, and no restriction on the number of parents is imposed. Edges that appear at least seventy percent of the time in the bootstrap routine are retained in the final model. The `blacklist` option forbids edges that do not respect the causal order of the variables. The visualization of the DAG is obtained via the `qgraph` function. The code used for tabu search based on the maximization of the BIC is:

```
boot.tabu.bic <- boot.strength(train, R = 1500,
    m = nrow(train), algorithm = "tabu",
    algorithm.args = list(score = "bic", blacklist = bl),
    cpdag = F, debug = FALSE)

arc.set <- boot.tabu.bic[(boot.tabu.bic$strength > 0.7)
    & (boot.tabu.bic$direction >= 0.5), ]

dag.tabu.bic <- bnlearn::empty.graph(colnames(train))
bnlearn::arcs(dag.tabu.bic) <- arc.set[,1:2]
```

The output is reported in Fig. 2. The DAG suggests that, for example, the outcome of the diabetes test (DIAB) is independent of all other variables given the woman's age (AGE) and glucose concentration (GLUC). Similarly, the result of the pedigree function (PED) is independent of all other variables given the serum insulin (INS).

Other scores used in conjunction with a tabu search or constraint-based algorithms return different DAG structures. This is shown in Fig. 3, where the DAGs resulting from other learning algorithms from the `bnlearn` package are shown. Although different, such DAGs share some common structures. For instance, plasma glucose concentration (GLUC) has a direct influence on the outcome of the diabetes test (DIAB). Similarly, the woman's age (AGE) always directly affects the number of pregnancies (PREG). The only variables that have a direct effect on the outcome of the diabetes test (DIAB) are the plasma glucose concentration (GLUC), as already noticed, the woman's age (AGE), and the diabetes pedigree function (PED). In general, DAGs resulting from score-based algorithms (i.e. Figs. 2 and 3(a)–3(d)) have more edges than DAGs resulting from constraint-based algorithms.

[1] We chose this dataset because it best showcases the function of our monitors. However, we acknowledge that this data is used here without the consent of or compensation for the original Akimel O'odham participants.



(a) Tabu with AIC score (Tabu-AIC)

(b) Tabu with fnml score (Tabu-FNML)

(c) Tabu with K2 score (Tabu-K2)

(d) Tabu with bdla score (Tabu-BDLA)

(e) Grow-Shrink (GS) algorithm

(f) PC stable (PC) algorithm

Fig. 3. DAGs learned using other scores for score-based search or constraint-based algorithms.

Table 2
Global and node monitors for the DAGs learned over the `diabetes` dataset.

Vertex	Tabu-BIC	Tabu-AIC	Tabu-FNML	Tabu-K2	Tabu-BDLA	GS	PC
AGE	221.85	221.85	221.85	221.85	221.85	221.85	221.85
PREG	161.95	161.95	161.95	161.95	161.95	161.95	161.95
MASS	223.07	224.64	223.07	223.07	223.07	223.07	223.07
GLUC	221.92	213.93	216.96	222.60	221.92	222.98	222.98
PRES	208.10	207.03	208.10	203.74	208.10	222.17	208.10
TRIC	199.39	199.25	199.39	199.39	199.39	199.39	199.39
INS	202.36	194.10	191.09	191.09	202.36	202.36	202.29
PED	220.93	223.06	220.93	223.06	220.93	222.77	222.77
DIAB	180.85	177.75	177.75	177.75	180.85	180.49	177.75
GM	1840.42	1823.55	1821.09	1824.49	1840.42	1857.04	1840.15

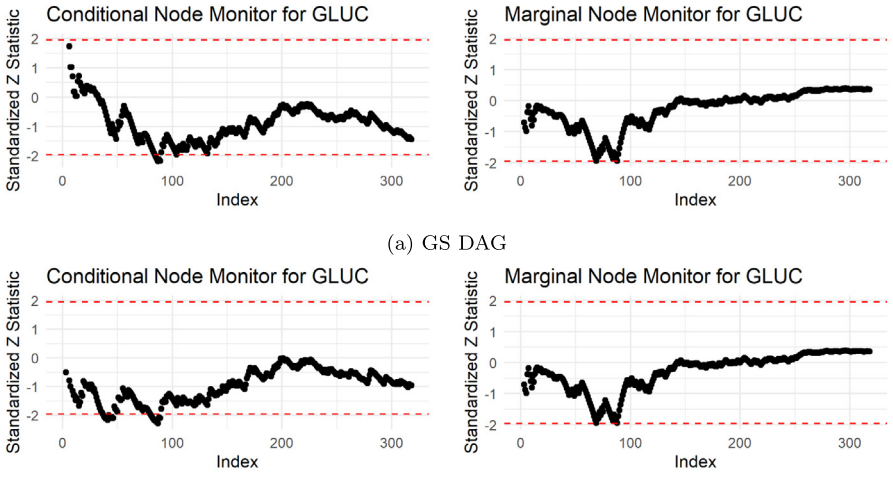
4.3. Using robustness monitors

We start the robustness analysis by computing the contributions of each individual node to the marginal log-likelihood of the model via the function `node_monitor` over the test dataset.

```
node_monitor(dag = dag.tabu.bic, df = test)
```

The output is shown in the first two columns of Table 2. Furthermore, the last row shows the output of `global_monitor (dag.tabu.bic, test)`, coinciding with the sum of the previous rows. Although the nodes MASS and AGE have a high contribution to the likelihood, they are roots of the DAG in Fig. 2, which are often irrelevant for checking robustness (see e.g. [13]). The glucose concentration (GLUC) also contributes greatly to the likelihood. For this reason, this node will be investigated further, together with the outcome of the diabetes test (DIAB)

The node and global monitors of all other learned networks are also reported in Table 2. The global monitor suggests that the model learned with the GS constraint-based algorithm better fits the test data. The node monitors highlight the differences and similarities between the learned DAGs. For instance, the node



(a) GS DAG

(b) Tabu-BIC DAG

Fig. 4. Sequential node monitors for the GLUC variable. The red dashed line indicates $|Z_i| = 1.96$.

monitors of TRIC are almost all the same, and it is only slightly lower for DAG learned optimizing the AIC. In this DAG, TRIC has two additional parents (GLUC and PREG), which therefore have minimal impact on the quality of the model. Most importantly, we notice that the main difference between the GS DAG and the others comes from the PRES node. In the GS DAG, the glucose concentration (GLUC) has no parents and is independent of all other variables, while in the other DAGs, it has at least one parent.

Although not penalizing for complexity, the global monitor favors the simplest model (the GS DAG has only four edges, while all others have larger edge sets). In our experience, this is only sometimes the case, and it is a characteristic of the dataset considered in this analysis. This DAG would result in the best-scoring one even if the global monitor were computed over the train dataset. Given all these observations, and without reason to prefer any other models, the modeler would tend to choose the GS DAG.

Next, we continue to investigate the similarities and the differences of the learned DAGs taking advantage of the capabilities of `bnmonitor`. The marginal and conditional node monitors for GLUC are computed over the BIC and GS DAGs using the functions `seq_marg_monitor` and `seq_cond_monitor`. The test data is increasingly ordered by level of insulin.

```
plot(seq_marg_monitor(dag.tabu.bic, test, "GLUC"))
plot(seq_marg_monitor(dag.gs, test, "GLUC"))
plot(seq_cond_monitor(dag.tabu.bic, test, "GLUC"))
plot(seq_cond_monitor(dag.gs, test, "GLUC"))
```

The output in Fig. 4 shows that the estimated probabilities for GLUC are robust and a good fit for the data, since the monitors are almost always within the 1.96 confidence bands. There is a suspicious decrease of fit for observations numbered between 75 and 125 (when increasingly ordered for level of insuline), but overall this does not seem too concerning. This implies that adding or deleting the MASS vertex as parent of GLUC (as shown in Figs. 2 and 3), does not affect the fit of the model to the test data.

Fig. 5 shows the sequential monitors for the DIAB node when observations are increasingly ordered skin thickness. The conditional monitors show a poor fit to the whole test dataset for both the Tabu-BIC and GS DAGs. To investigate further the fit to data for the GS DAG, we construct parent–child monitors, that examine the forecasts from the subset of the data which has a specific value of the parents of a node. The parent of DIAB is GLUC. For instance, the following code constructs the parent–child monitor when the parents take the low value:

```
plot(seq_pa_ch_monitor(dag.gs, test, "PRES",
                       pa.names = c("GLUC"), pa.val = c("low")))
```

Fig. 6 shows the monitors for the two parent configurations of the node DIAB. The model struggles to forecast the outcome of the diabetes test for participants who have a high glucose level since Fig. 6(b) shows a large number of points beyond 1.96. Conversely, the forecasts given low levels of glucose configurations are accurate. This may be an indication that the assumption of symmetric conditional independence is not tenable and that a model accounting for context-specific independence is more appropriate (see e.g. [59]).

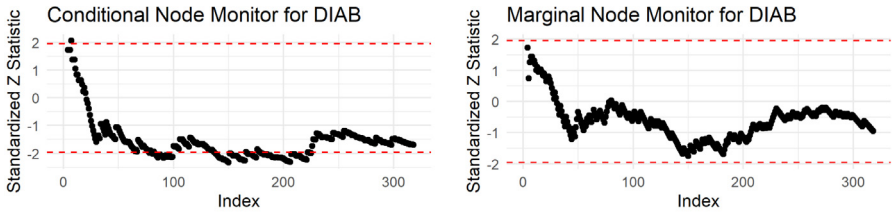
As a last check, the influence of the observations in the test data is computed via the `influential_obs` function. The most influential observations can be thought of as the most unusual ones or as those that would have the biggest impact if the model were learned via a score-based structural learning algorithm over the test data. They can be computed for the Tabu-BIC DAG using the code:

```
influence <- influential_obs(dag.tabu.bic, diabetes)
plot(influence)
```

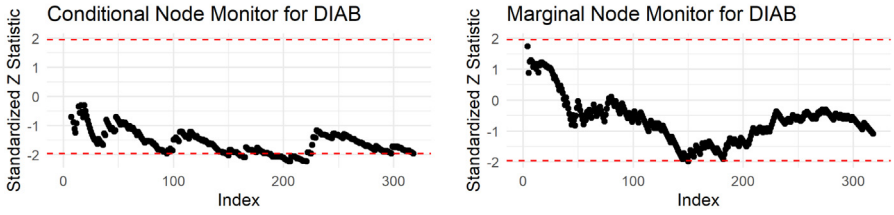
Fig. 7 shows that a few observations highly contribute to the likelihood of the test data (influence over eight). Since identical observations have the same influence, the measure is plotted only once for each combination of the variables' levels (this is why only around 200 points are plotted in Fig. 7). The actual observations with high influence can be derived as follows.

```
subset(influence,score > 8)
```

The output is shown in Table 3. It can be seen that the observations with the highest influence are all for high levels of blood pressure (PRES), low serum insulin (INS), and a positive diabetes test (DIAB). Similar results have been observed over the GS DAG,

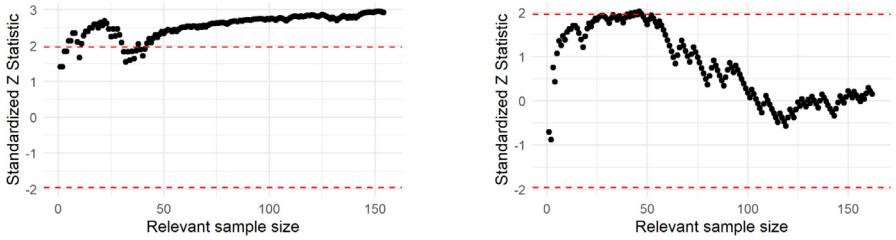


(a) GS DAG



(b) Tabu-BIC DAG

Fig. 5. Sequential node monitors for the DIAB variable. The red dashed line indicates $|Z_i| = 1.96$.



(a) GLUC = high (b) GLUC = low

Fig. 6. Parent–child monitors for the DIAB vertex.

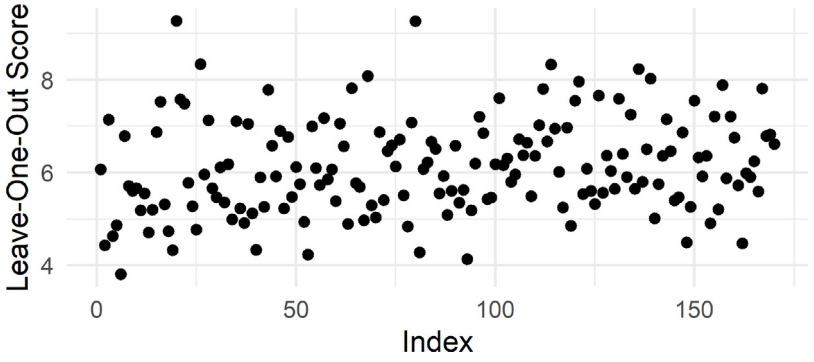


Fig. 7. Influence score of the observations in the diabetes dataset.

Table 3
Most influential observations in the diabetes dataset.

AGE	PREG	MASS	GLUC	PRES	TRIC	INS	PED	DIAB	Score
low	high	high	low	high	low	low	low	pos	9.272
low	high	low	high	high	high	low	low	pos	9.265
low	low	low	low	high	high	low	low	pos	8.334
low	high	high	high	high	low	low	high	pos	8.330
high	low	high	low	high	high	low	high	pos	8.234
high	low	low	low	low	high	low	high	neg	8.080
high	low	high	high	low	low	high	low	neg	8.027

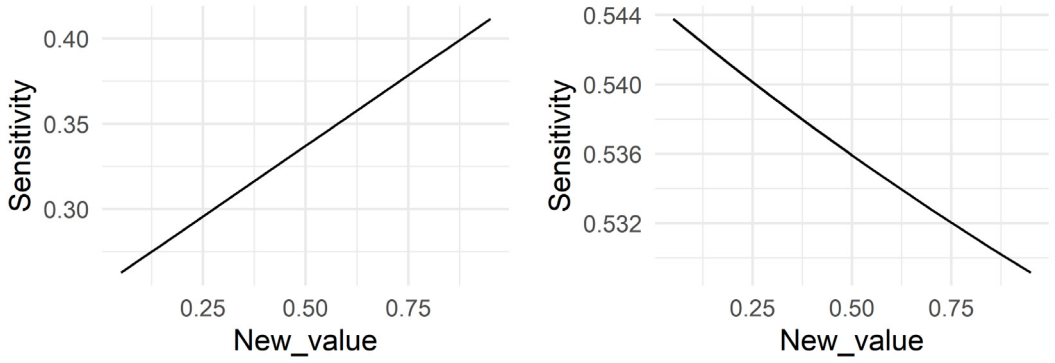
thus highlighting once more the slight difference between the two DAG structures.

4.4. Checking the learned probabilities

The implications of the posterior probabilities are next investigated. First, a `bn.fit` object, henceforth called `bn`, is created. The conditional probability tables are learned using the maximum likelihood approach over the whole dataset.

```
bn <- bnlearn::bn.fit(dag.tabu.bic, diabetes)
```

The vertex of most interest is `DIAB` reporting the result of a diabetes test (either positive or negative). As an illustration, consider first how the probability of a positive test depends on the probability of `GLUC = high` conditional on `MASS = high`. This



(a) `sens g` (b) `sens m`

Fig. 8. Plots of the sensitivity functions for the `diabetes` dataset.

can be done via the function `sensitivity`, whose output can be plotted using the `plot` method.

```
sens_g <- sensitivity(bn, interest_node = "DIAB",
    interest_node_value = "pos", node = "GLUC",
    value_node = "high", value_parents = "high",
    new_value ="all")
plot(sens_g)
```

The output is reported in Fig. 8(a). The plot shows that as the conditional probability of having a high glucose level increases, then the one of a positive test does.

In the previous example, the probability of interest was the marginal probability of a positive test, but more general conditional probabilities can also be studied. As an illustration, consider the conditional probability of a high number of pregnancies given a positive diabetes test as a function of the probability of a high body mass index. This can be done similarly to the previous code, but now the `evidence_nodes` and `evidence_states` inputs must be fixed.

```
sens_m <-  sensitivity(bn, interest_node = "PREG",
    interest_node_value = "high",
    evidence_nodes = "DIAB", evidence_states = "pos",
    node = "MASS", value_node = "high",
    value_parents = NULL, new_value ="all")
plot(sens_m)
```

The output is reported in Fig. 8(b). The posterior (conditional) probability of a high number of pregnancies decreases when the probability of a high level of body mass index increases. Notice that: (i) the decrease is non-linear, as expected from the results of Coupé and Van der Gaag [12]; (ii) since `MASS` is a root vertex, the `value_parents` input is set to `NULL`; (iii) the range of variation of the probability of interest is very narrow, indicating that the varied probability has a minimal effect on the output probability.

The CD distance computes the overall difference in the BN when a specific parameter is changed. This is computed using the `CD` function for the parameters considered in Fig. 8.

```
cd_g <- CD(bn, node = "GLUC", value_node = "high",
    value_parents = "high", new_value ="all")
plot(cd_g)

cd_m <- CD(bn, node = "MASS", value_node ="high",
    value_parents = NULL, new_value="all")
plot(cd_m)
```

Table 4
Possible probability changes output of `sensquery`.

Node value	Node	Value parents	Original value	Suggested change	CD distance
AGE	low		0.5156250	0.4120818	0.4178864
GLUC	low	low	0.5958549	0.4784748	0.4743775
PRES	low	low	0.6717172	0.7838409	0.5722294
DIAB	pos	low,low	0.0785124	0.1810470	0.9534626
MASS	low		0.5026042	0.1643043	1.6369608

The output is given in Fig. 9. The CD distance is zero at the original value of the parameter. It can be seen that although the sensitivity function associated with variations in the probability of a low body mass index was almost constant (Fig. 8(b)), variations of this parameter are associated with more extensive changes to the overall BN probability distribution.

As a final illustration of the capabilities of `bnmonitor`, suppose a change on a probability implied by the learned BN is imposed, for instance, to carry out a scenario analysis. As an example, the conditional probability of a positive diabetes test given a high-pressure level is calculated using the `cpquery` function from the `bnlearn` package as 0.37.

```
bnlearn::cpquery(bn, event = (DIAB == "pos"),
    evidence = (PRES == "high"))
```

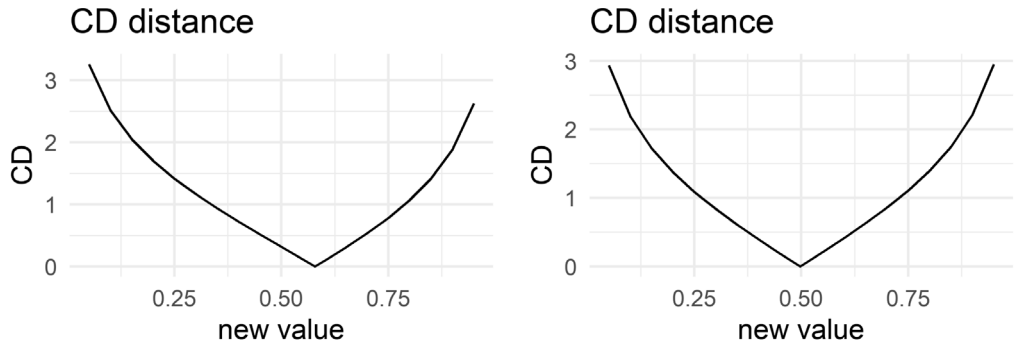
Suppose this probability is believed to be at least 0.4. The function `sensquery` outputs the probability changes that would meet this constraint.

```
sensquery(bn, interest_node = "DIAB",
    interest_node_value = "pos", new_value = 0.4,
    evidence_nodes = "PRES", evidence_states = "high")
```

The output is reported in Table 4. Five possible changes in the conditional probabilities of the model meet this constraint. The function further reports the CD distance associated with the changes, which ranges from 0.42 up to 1.64. According to the CD distance, the best parameter to update would be from AGE's (marginal) distribution.

5. Computational experiments

Next, the computational cost of calculating the various monitors and statistics using the `bnmonitor` package is explored. First,



(a) `cd_g` (b) `cd_m`

Fig. 9. Plots of the CD distances for the `diabetes` dataset.

Table 5
Details about 8 datasets from the probabilistic graphical models literature and about their associated DAG learned with an hill-climbing algorithm based on the optimization of the BIC score.

Dataset	# Observations	# Variables	Max # Levels	# Edges	Average Degree	# Parameters
Asia	5000	8	2	7	1.75	17
Cachexia	77	7	3	6	1.71	35
Chds	500	4	3	3	1.50	10
Coronary	1841	6	2	8	2.67	19
Ksl	1083	9	2	12	2.67	26
Mathmarks	88	5	3	4	1.60	26
PhD	915	6	3	4	1.33	16
Titanic	2201	4	4	5	2.50	23

we consider eight datasets from the literature on probabilistic graphical models. If required, variables are discretized using the equal-frequency method. Table 5 gives information about the datasets and the associated learned DAG of the BN model.

For each dataset, we compute our monitors and record the required computational time. The global monitor and the influence are computed once for each BN (node monitors are not reported since the required computational time is equal to global monitors). Sequential monitors are computed for each node, and each dataset's average computing time is reported (parent–child monitors are not reported since they are as SMNM but over a smaller dataset). For sensitivity functions, the `sensequery` function, and the CD distance, we repeat 100 times the following procedure. The parameter to be varied is chosen by selecting at random a variable and one of its levels; if the variable has parents in the underlying DAG, the levels of the parent variables are also chosen at random. The probability of interest is chosen by selecting two variables at random with replacement; if the variables are the same, then the probability of interest is marginal, and one level of the variable is chosen at random; if the variables are different, the probability of interest is conditional, and the levels of the two variables are selected at random. The reported computational time is then the average of the 100 repetitions.

The results are reported in Table 6. In general, the most expensive monitors to compute are the sequential ones, since they require the computation of the predictive distribution a number of times equal to the sample size. The computational times are higher for BNs estimated from bigger datasets. The computation of SCNM is slightly longer than SMNM since it further requires a marginalization step over the network. The computation of the influence can also be expensive and there is some indication that it is affected by the number of nodes of the network. The other statistics are computed extremely quickly by `bnmonitor`, with `sensitivity` requiring a bit more time than the others.

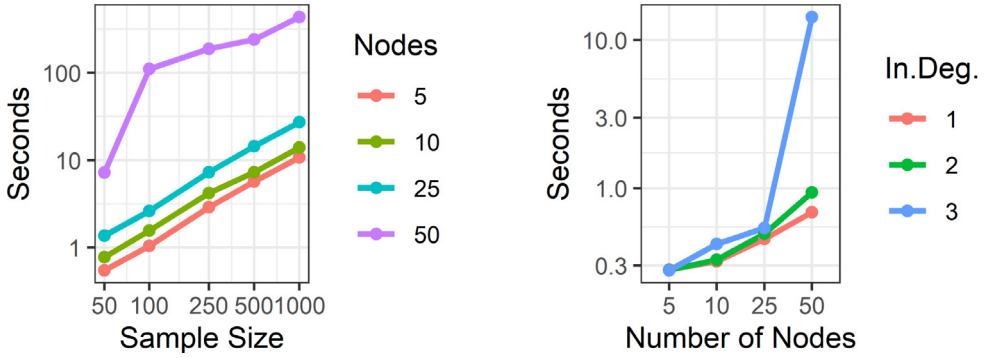
Table 6
Computational times (in seconds) of the monitors for the 8 datasets from the probabilistic graphical models literature. Computations carried out with an intel CORE i7 vPro 8th Gen.

Dataset	GM	SMNM	SCNM	I	S	`sensquery`	CD
Asia	0.08	67.69	71.44	4.59	0.35	0.03	0.03
Cachexia	0.08	1.00	1.08	4.24	0.36	0.03	0.03
Chds	0.05	5.26	5.48	0.94	0.32	0.03	0.01
Coronary	0.07	23.07	25.47	4.00	0.32	0.02	0.03
Ksl	0.11	15.25	15.80	51.72	0.35	0.02	0.05
Mathmarks	0.07	1.04	1.01	2.92	0.30	0.02	0.02
PhD	0.06	10.56	11.18	6.25	0.30	0.02	0.02
Titanic	0.05	24.76	25.19	1.03	0.30	0.02	0.03

Table 6 reports the results. In general, the most expensive monitors to compute are the sequential ones since they require the computation of the predictive distribution a number of times equal to the sample size. The computational times are higher for BNs estimated from more extensive datasets. The computation of SCNM is slightly longer than SMNM since it further requires a marginalization step over the network. The computation of the influence can also be expensive, and there is some indication that it is affected by the number of nodes in the network. The other statistics are computed extremely quickly by `bnmonitor`, with `sensitivity` requiring more time than the others.

To further investigate the effect of various network and data features, we perform a simulation study to assess the computational time of the SCNM and the sensitivity function. For SCNM, we simulate networks with an increasing number of vertices using the algorithm of Melançon and Philippe [60] and a maximum in-degree of three to mimic the form of real-world networks. The associated conditional probability tables are filled with uniform random numbers between zero and one, and datasets of increasing size are simulated from such a BN. The SCNMs are then computed for each network node, and the average computing time is reported. The results are shown in Fig. 10(a). For BNs with up to 25 nodes, the computational time increases linearly with the sample size. For the case of 50 nodes, the computational time increases abruptly due to the extra marginalization step required by the SCNM.

The computational times of the `sensitivity` function are also investigated via simulation. Networks of varying number of nodes and max in- degree are simulated, and the sensitivity function to compute is chosen using the same routine as above. The results are reported in Fig. 10(b). The computation of `sensitivity` requires less than a second for all cases except for a



(a) SCNM (b) Sensitivity

Fig. 10. Computing times for the simulation study.

BN with 50 nodes and max in-degree of 3. In such a case, the computational times increase abruptly due to the complexity of the marginalization step to compute the required probabilities.

6. Discussion

We demonstrated the usage of the `bnmonitor` R package with a real-world dataset and the insights modelers can get by conducting thorough model-checking. Methods to assess the fit to data and the effect of parameter changes on outputs of interest were showcased, as well as the plotting capabilities of `bnmonitor`, which take advantage of the R environment and the seamless integration with the famous `bnlearn` package.

As mentioned, `bnmonitor` can also deal with continuous random variables and provides various functions to investigate Gaussian BNs. Furthermore, since all variables were binary in our application when we varied a probability by default, the other was set as one minus the new value. However, in the generic categorical case, various methods to adjust probabilities [61] are available and implemented in `bnmonitor`.

Currently, `bnmonitor` implements sensitivity methods where only one parameter at a time can be varied, although it can support more general analyses (see e.g. [62]). The implementation of the so-called multi-way methods is currently under investigation, but it is known that these have become computationally very expensive [63]. However, recent algorithmic advances have made such analyses possible [64,65], and we plan to include these novel algorithms in future releases of `bnmonitor`.

Furthermore, sensitivity analysis in BNs is an active field of research (see e.g. the recent contributions of Ballester-Ripoll and Leonelli [66] and Wright and Smith [67], in the field of sensitivity to evidence and propagation errors, respectively). The newly proposed methods will be implemented in `bnmonitor` to provide users with an even more comprehensive array of functions for model-checking.

CRediT authorship contribution statement

Manuele Leonelli: Writing – review & editing, Writing – original draft, Software, Methodology, Formal analysis, Data curation, Conceptualization. **Ramsiya Ramanathan:** Software, Data curation, Conceptualization. **Rachel L. Wilkerson:** Software, Methodology, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data is already included in the software package described in the paper.

References

[1] E. Borgonovo, E. Plischke, Sensitivity analysis: A review of recent advances, European J. Oper. Res. 248 (3) (2016) 869–887.
[2] A. Saltelli, S. Tarantola, F. Campolongo, Sensitivity analysis as an ingredient of modeling, Statist. Sci. 15 (4) (2000) 377–395.
[3] J. Pitchforth, K. Mengersen, A proposed validation framework for expert elicited Bayesian networks, Expert Syst. Appl. 40 (1) (2013) 162–167.
[4] J. Pearl, Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan Kaufmann, 1988.
[5] M. Akhavan, M.V. Sebt, M. Ameli, Risk assessment modeling for knowledge based and startup projects based on feasibility studies: A Bayesian network approach, Knowl.-Based Syst. 222 (2021) 106992.
[6] C. Bielza, P. Larranaga, Bayesian networks in neuroscience: A survey, Front. Comput. Neurosci. 8 (2014) 131.
[7] B. Cai, X. Kong, Y. Liu, J. Lin, X. Yuan, H. Xu, R. Ji, Application of Bayesian networks in reliability evaluation, IEEE Trans. Ind. Inform. 15 (4) (2018) 2146–2157.
[8] Y. Chen, R. Chen, J. Hou, M. Hou, X. Xie, Research on users' participation mechanisms in virtual tourism communities by Bayesian network, Knowl.-Based Syst. 226 (2021) 107161.
[9] B. Drury, J. Valverde-Rebaza, M.-F. Moura, A. de Andrade Lopes, A survey of the applications of Bayesian networks in agriculture, Eng. Appl. Artif. Intell. 65 (2017) 29–42.
[10] S. McLachlan, K. Dube, G.A. Hitman, N.E. Fenton, E. Kyrimi, Bayesian networks in healthcare: Distribution by medical condition, Artif. Intell. Med. 107 (2020) 101912.
[11] H. Chan, A. Darwiche, When do numbers really matter? J. Artificial Intelligence Res. 17 (2002) 265–287.
[12] V.M. Coupé, L.C. Van der Gaag, Properties of sensitivity analysis of Bayesian belief networks, Ann. Math. Artif. Intell. 36 (4) (2002) 323–356.
[13] R.G. Cowell, R.J. Verrall, Y. Yoon, Modeling operational risk with Bayesian networks, J. Risk Insurance 74 (4) (2007) 795–827.
[14] J. Rohmer, Uncertainties in conditional probability tables of discrete Bayesian belief networks: A comprehensive review, Eng. Appl. Artif. Intell. 88 (2020) 103384.
[15] S.H. Chen, C.A. Pollino, Good practice in Bayesian network modelling, Environ. Model. Softw. 37 (2012) 134–145.
[16] M. Hänninen, P. Kujala, Influences of variables on ship collision probability in a Bayesian belief network model, Reliab. Eng. Syst. Saf. 102 (2012) 27–40.
[17] J. Kleemann, E. Celio, C. Fürst, Validation approaches of an expert-based Bayesian belief network in northern ghana, west africa, Ecol. Model. 365 (2017) 10–29.
[18] T. Makaba, W. Doorsamy, B.S. Paul, Bayesian network-based framework for cost-implication assessment of road traffic collisions, Int. J. Intell. Transp. Syst. Res. 19 (1) (2021) 240–253.
[19] M. Scutari, Learning Bayesian networks with the bnlearn R package, J. Stat. Softw. 35 (3) (2010) 1–22.
[20] H. Yu, J. Moharil, R.H. Blair, Bayesnetbp: An r package for probabilistic reasoning in Bayesian networks, J. Stat. Softw. 94 (3) (2020) 1–31.

[21] S. Højsgaard, Graphical independence networks with the gRain package for R, J. Stat. Softw. 46 (10) (2012) 1–26.
[22] H. Wickham, Ggplot2: Elegant Graphics for Data Analysis, Springer-Verlag New York, 2016.
[23] S. Epskamp, A.O.J. Cramer, L.J. Waldorp, V.D. Schmittmann, D. Borsboom, Qgraph: Network visualizations of relationships in psychometric data, J. Stat. Softw. 48 (4) (2012) 1–18.
[24] C. Görgen, M. Leonelli, Model-preserving sensitivity analysis for families of Gaussian distributions, J. Mach. Learn. Res. 21 (84) (2020) 1–32.
[25] S. French, Aggregating expert judgement, Rev. Real Acad. Cienc. Exact. Fis. Natl. Ser. A. Mat. 105 (1) (2011) 181–206.
[26] A.M. Hanea, G.F. Nane, T. Bedford, S. French, Expert Judgement in Risk and Decision Analysis, Springer, 2021.
[27] A. O'Hagan, Expert knowledge elicitation: Subjective but scientific, Amer. Statist. 73 (sup1) (2019) 69–81.
[28] A.C. Constantinou, N. Fenton, W. Marsh, L. Radlinski, From complex questionnaire and interviewing data to intelligent Bayesian network models for medical decision support, Artif. Intell. Med. 67 (2016) 75–93.
[29] S. Renooij, Probability elicitation for belief networks: Issues to consider, Knowl. Eng. Rev. 16 (3) (2001) 255–269.
[30] G. Zhang, V.V. Thai, Expert elicitation and Bayesian network modeling for shipping accidents: A literature review, Saf. Sci. 87 (2016) 53–62.
[31] C. Werner, T. Bedford, R.M. Cooke, A.M. Hanea, O. Morales-Napoles, Expert judgement for dependence in probabilistic modelling: A systematic literature review and future research directions, European J. Oper. Res. 258 (3) (2017) 801–819.
[32] R.L. Wilkerson, J.Q. Smith, Customized structural elicitation, in: Expert Judgement in Risk and Decision Analysis, Springer, 2021, pp. 83–113.
[33] A. Cano, A.R. Masegosa, S. Moral, A method for integrating expert knowledge when learning Bayesian networks from data, IEEE Trans. Syst. Man Cybern. B 41 (5) (2011) 1382–1394.
[34] A.C. Constantinou, N. Fenton, M. Neil, Integrating expert knowledge with data in Bayesian networks: Preserving data-driven expectations when the expert variables remain unobserved, Expert Syst. Appl. 56 (2016) 197–208.
[35] Y. Zhou, N. Fenton, M. Neil, Bayesian network approach to multinomial parameter learning using data and expert judgments, Internat. J. Approx. Reason. 55 (5) (2014) 1252–1268.
[36] J. Dai, J. Ren, W. Du, Decomposition-based Bayesian network structure learning algorithm using local topology information, Knowl.-Based Syst. 195 (2020) 105602.
[37] Z. Wang, X. Gao, Y. Yang, X. Tan, D. Chen, Learning Bayesian networks based on order graph with ancestral constraints, Knowl.-Based Syst. 211 (2021) 106515.
[38] P. Spirtes, C.N. Glymour, R. Scheines, D. Heckerman, Causation, Prediction, and Search, MIT Press, 2000.
[39] C. Bielza, P. Larranaga, Discrete Bayesian network classifiers: A survey, ACM Comput. Surv. 47 (1) (2014) 1–43.
[40] N. Friedman, D. Geiger, M. Goldszmidt, Bayesian network classifiers, Mach. Learn. 29 (2) (1997) 131–163.
[41] J.H. Bolt, L.C. van der Gaag, Balanced sensitivity functions for tuning multi-dimensional Bayesian network classifiers, Internat. J. Approx. Reason. 80 (2017) 361–376.
[42] D. Heckerman, A tutorial on learning with Bayesian networks, in: Innovations in Bayesian Networks, Springer, 2008, pp. 33–82.
[43] D. Heckerman, D. Geiger, D.M. Chickering, Learning Bayesian networks: The combination of knowledge and statistical data, Mach. Learn. 20 (3) (1995) 197–243.
[44] N. Friedman, M. Goldszmidt, A. Wyner, Data analysis with Bayesian networks: a bootstrap approach, in: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, 1999, pp. 196–205.
[45] N. Friedman, D. Koller, Being Bayesian about network structure. A Bayesian approach to structure discovery in Bayesian networks, Mach. Learn. 50 (2003) 95–125.
[46] G.E. Box, An apology for ecumenism in statistics, in: Scientific Inference, Data Analysis, and Robustness, Elsevier, 1983, pp. 51–84.
[47] L. Devroye, L. Györfi, G. Lugosi, A Probabilistic Theory of Pattern Recognition, Vol. 31, Springer Science & Business Media, 2013.
[48] A.P. Dawid, Prequential Data Analysis, in: Lecture Notes - Monograph Series, JSTOR, 1992, pp. 113–126.
[49] J.M. Bernardo, A.F. Smith, Bayesian Theory, John Wiley & Sons, 2009.
[50] R.G. Cowell, P. Dawid, S.L. Lauritzen, D.J. Spiegelhalter, Probabilistic Networks and Expert Systems: Exact Computational Methods for Bayesian Networks, Springer Science & Business Media, 2006.
[51] F. Seillier-Moiseiwitsch, A. Dawid, On testing the validity of sequential probability forecasts, J. Amer. Statist. Assoc. 88 (421) (1993) 355–359.
[52] R.L. Wilkerson, J.Q. Smith, Bayesian diagnostics for chain event graphs, 2019, arXiv:1910.04679.
[53] C. Boutilier, N. Friedman, M. Goldszmidt, D. Koller, Context-specific independence in Bayesian networks, in: Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence, 1996, pp. 115–123.
[54] R.A. Collazo, C. Görgen, J.Q. Smith, Chain Event Graphs, CRC Press, 2018.
[55] S. French, Cynefin: Uncertainty, small worlds and scenarios, J. Oper. Res. Soc. 66 (10) (2015) 1635–1645.
[56] S. Kullback, R.A. Leibler, On information and sufficiency, Ann. Math. Stat. 22 (1) (1951) 79–86.
[57] H. Chan, A. Darwiche, A distance measure for bounding probabilistic belief change, Internat. J. Approx. Reason. 38 (2) (2005) 149–174.
[58] F. Nojavan, S.S. Qian, C.A. Stow, Comparative analysis of discretization methods in Bayesian networks, Environ. Model. Softw. 87 (2017) 64–71.
[59] F. Carli, M. Leonelli, E. Riccomagno, G. Varando, The R package stagedtrees for structural learning of stratified staged trees, J. Stat. Softw. 102 (6) (2022) 1–30.
[60] G. Melançon, F. Philippe, Generating connected acyclic digraphs uniformly at random, Inform. Process. Lett. 90 (4) (2004) 209–213.
[61] S. Renooij, Co-variation for sensitivity analysis in Bayesian networks: Properties, consequences and alternatives, Internat. J. Approx. Reason. 55 (4) (2014) 1022–1042.
[62] M. Leonelli, E. Riccomagno, A geometric characterisation of sensitivity analysis in monomial models, Internat. J. Approx. Reason. 151 (2022) 64–84.
[63] H. Chan, A. Darwiche, Sensitivity analysis in Bayesian networks: from single to multiple parameters, in: Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence, 2004, pp. 67–75.
[64] R. Ballester-Ripoll, M. Leonelli, You only derive once (YODO): Automatic differentiation for efficient sensitivity analysis in Bayesian networks, in: International Conference on Probabilistic Graphical Models, PMLR, 2022, pp. 169–180.
[65] R. Ballester-Ripoll, M. Leonelli, The YODO algorithm: An efficient computational framework for sensitivity analysis in Bayesian networks, Internat. J. Approx. Reason. 159 (2023) 108929.
[66] R. Ballester-Ripoll, M. Leonelli, Computing Sobol indices in probabilistic graphical models, Reliab. Eng. Syst. Saf. 225 (2022) 108573.
[67] S.K. Wright, J.Q. Smith, Bayesian networks, total variation and robustness, 2018, arXiv:1811.07179.