Additive Bayesian Network Modelling with the R Package abn

Gilles Kratzer
Zurich University

Fraser Iain Lewis
Danone Nutricia Research

Arianna Comin
National Veterinary Institute

Marta Pittavino
Geneva University

Reinhard Furrer
Zurich University

Abstract

The R package **abn** is designed to fit additive Bayesian models to observational datasets. It contains routines to score Bayesian networks based on Bayesian or information theoretic formulations of generalized linear models. It is equipped with exact search and greedy search algorithms to select the best network. It supports a possible blend of continuous, discrete and count data and input of prior knowledge at a structural level. The Bayesian implementation supports random effects to control for one-layer clustering. In this paper, we give an overview of the methodology and illustrate the package's functionalities using a veterinary dataset about respiratory diseases in commercial swine production.

Keywords: structure learning, graphical models, greedy search, exact search, scoring algorithm, GLM, graph theory, R.

1. Introduction

Bayesian network (BN) modelling has an impressive track record in analysing systems epidemiology datasets (Cornet, Sierra, Tournebize, Gabrielle, and Lewis 2016; McCormick 2014; Hartnack, Odoch, Kratzer, Furrer, Wasteson, L'Abée-Lund, and Skjerve 2019), especially in veterinary epidemiology (McCormick, Sanchez-Vazquez, and Lewis 2013; Ludwig, Berthiaume, Boerlin, Gow, Léger, and Lewis 2013; Firestone, Lewis, Schemann, Ward, Toribio, Taylor, and Dhand 2014; Cornet *et al.* 2016; Hartnack, Springer, Pittavino, and Grimm 2016; Pittavino, Dreyfus, Heuer, Benschop, Wilson, Collins-Emerson, Torgerson, and Furrer 2017; Ruchti, Meier, Würbel, Kratzer, Gebhardt-Henrich, and Hartnack 2018; Ruchti, Kratzer, Furrer, Hartnack, Würbel, and Gebhardt-Henrich 2019; Comin, Jeremiasson, Kratzer, and Keeling 2019). It is a particularly well-suited approach to better understand the underlying structure of data when scientific understanding of the data is at an early stage. BN modelling is designed to sort out directly from indirectly related variables and offers a far richer modelling framework than classical approaches in epidemiology like, e.g., regression techniques or extensions thereof. In contrast to structural equation modelling (Hair, Black, Babin, Anderson, Tatham *et al.* 1998), which requires expert knowledge to design the model, the Additive Bayesian Network (ABN) method is a data-driven approach (Lewis and Ward 2013; Kratzer, Pittavino, Lewis, and Furrer 2019b). It does not rely on expert knowledge, but it can possibly incorporate it. Contrary to other scoring approaches, it can handle a blend of variables distributions. Thanks to its formulation, the ABN method supports variables adjustment and can control for clustering. Within the same framework, the network can be scored and the effect size estimated. This paper aimes at presenting the actual R implementation of the ABN methodology using a simple case study. An older package vignette exists that targets presenting more detailed code.

R is an open-source, reliable and easy-to-use environment for statistical computing. It is very popular in the epidemiological community (R Core Team 2017). In this paper, ABN refers to the methodology and **abn** refers to the R package.

The aim of the R package **abn** is to provide researchers with a free implementation of a set of functions to score, select, analyse and report an ABN model. The main functionalities of the package are Bayesian, information theoretic scoring, exact and greey search algorithms. The R package **abn** is also equipped with ancillary functions to simulate and manipulate ABN models. The package is available through the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=abn`. The unique feature of the R package **abn** compared to other network modelling R packages is the Bayesian-based scoring system, which makes ABN methodology theoretically sound and computationally competitive thanks to the internal use of iteratively nested Laplace approximations (INLA) (Rue, Martino, and Chopin 2009, available at `www.r-inla.org/download`). From an applied perspective, a regression framework is suitable for analyses that target data modelling and seek to report insight of observational data. A byproduct of the Bayesian regression framework used by the R package **abn** is the possibility of adjusting an analysis for clustering, which is a common concern in systems epidemiology and not possible in other network modelling R packages. Beyond its nice theoretical framework, the R package **abn** is equipped with the only R implementation of an exact search based on dynamical programming that targets BN modelling (Koivisto and Sood 2004).

The structure of this paper is as follows: First, we finish this section with a short motivating example and alternative approaches for modelling BNs. Then, Section 2 describes the comprehensive ABN methodology. Section 3 lists and describes the functionalities of the R package. It includes simulation studies for comparing the efficiency of the implemented scores. Section 4 presents a case study of an ABN modelling using data from the field of veterinary epidemiology. We conclude and summarize the article in Section 6. The appendix provides more technical details.

1.1. Motivating example

We start illustrating the main functionality of the R package **abn** by analyzing the so-called `asia` dataset provided by the R package **bnlearn** (Scutari 2010). It is a small synthetic dataset from Lauritzen and Spiegelhalter (1988) about lung diseases (tuberculosis, lung cancer and bronchitis) and visits to Asia. In total, the dataset consists of eight binary variables with 5'000 observations. As this dataset has been used in various illustrations, we do not present it here in detail, nor do we perform an initial exploratory data analysis. We assume all necessary packages are installed and we start with loading the data. The advantage of using the R package **abn** over a classical approach is that we take all data into account without making an a priori choice of what are the independent and the dependent variables.

```
R> library("abn")
R> data("asia", package = "bnlearn")
R> colnames(asia) <- c("Asia", "Smoking", "Tuberculosis",
R+     "LungCancer", "Bronchitis", "Either", "XRay", "Dyspnea")
```

We now determine the relationship between the data. A more theoretical and more detailed explanation is given in Sections 2 and 3.

```
R> distrib <- as.list(rep("binomial", 8))
R> names(distrib) <- names(asia)
R>
R> mycache <- buildscorecache(data.df = asia, data.dist = distrib,
R+     max.parents = 4)
R> mp.dag <- mostprobable(score.cache = mycache)
```

At this time, we have constructed the underlying relationship of the variables. In our context, we also speak of what we have learned about the structure. In a next step, we estimate the parameters describing the relationship between the variables, estimating the odds ratios of the conditional effects.

```
R> # code to plot the computed BN. The actual figure
R> # is manually adapted
R> fabn <- fitabn(object = mp.dag, create.graph = TRUE)
R> require(Rgraphviz)
R> plot(fabn$abnDag)
```

This example highlights that the R package **abn** does not model causality. However, it is possible to structurally constrain the modelling procedure in retaining or banning structures between the variables. To structurally compare two BNs, one usually use the so called Hamming distance(de Jongh and Druzdzel 2009) which is the number of arcs that should be changed on the reference network to exactly map the index one. The unconstrained model has a Hamming distance of 2 compared to the true network whereas the constrained model is at 1.

```
R> mycache <- buildscorecache(data.df = asia, data.dist = distrib,
R+     max.parents = 4, dag.retained = ~LungCancer | Smoking)
```

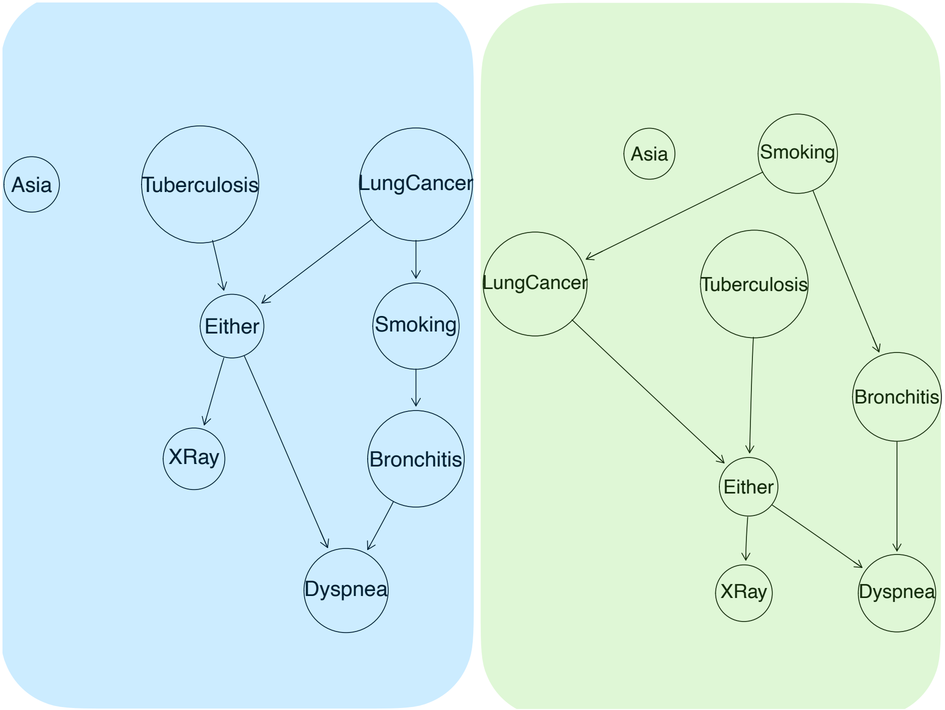

Figure 1: Network for `asia` dataset. Left: unconstrained, right: constrained. The constraint is on the direction of the link between LungCancer and smoking nodes.

```
R> mp.dag <- mostprobable(score.cache = mycache)
R> fabn <- fitabn(object = mp.dag, create.graph = TRUE)
```

The function `fitabn()` computes the parameter estimates from a given structure, a list of distributions and a dataset. The asia dataset is made of Bernoulli variables and then the parameter estimates are log odds ratio. The method used is a Bayesian approach, thus the estimates are the modes of the posterior distributions.

```
R> fabn
```

```
The ABN model was fitted using a Bayesian approach. The estimated modes are:

$Asia
Asia|(Intercept)
           -4.77

$Smoking
Smoking|(Intercept)
              0.012

$Tuberculosis
Tuberculosis|(Intercept)
                   -4.72
```

```
$LungCancer
LungCancer|(Intercept)     LungCancer|Smoking
                 -4.28                   2.26

$Bronchitis
Bronchitis|(Intercept)     Bronchitis|Smoking
                 -0.85                   1.78

$Either
 Either|(Intercept) Either|Tuberculosis   Either|LungCancer
              -11.4                19.1                21.1

$XRay
XRay|(Intercept)      XRay|Either
           -3.09             8.30

$Dyspnea
Dyspnea|(Intercept)  Dyspnea|Bronchitis      Dyspnea|Either
              -2.08                3.31                2.19

Number of nodes in the network: 8 .
```

There is an arrow between the nodes Lung Cancer and Smoking. The function `fitabn()` returns an intercepts which could be interpreted as noise and an odds ratio of 2.26 on the logit scale. In exponentiating it we get the classical odds ratio (OR: 9.58) which is a measure of association between the two random variable. The OR is larger than 1 meaning that Lung Cancer and Smoking are positivelly associated.

In this simple example, we have plenty of observations to estimate 15 parameters. In more realistic cases, we have to control for over-fitting. This will be further discussed in Section 4.4.

1.2. Alternative R packages

Not too surprisingly, there are several R packages available for BN modelling, which often cover a particular model and data niche. The R packages can be divided into two broad classes: the ones targeting parameters and structure learning and the ones targeting inference in BN models. Following R packages target parameters and structure learning. The most popular R package is the R package **bnlearn** (Scutari 2010). It has implementations of most of BD scores but also information theoretic scores for continuous and discrete mixed variables. Additionally, it has implementations of multiple network structure learning via multiple constraint-based and score-based algorithms. This is the most versatile R package for BN modeling and should be the prefered primary software choice. When focusing on a causal framework, the R package **pcalg** is very popular (Kalisch, Mächler, Colombo, Maathuis, and Bühlmann 2012). It contains an implementation of the PC-Algorithm that outputs an essential graph. When dealing with discrete Bayesian networks only, the R package **catnet** allows parameter and structure learning using likelihood-based criteria. The R package **deal** (Bøttcher, Dethlefsen *et al.* 2003) enables learning BN with a mixture of continuous and/or discrete variables under the conditional Gaussian distribution (restriction of discrete nodes being only parent of continuous ones). Other useful R packages are actively maintained on CRAN, but none of them implement scoring procedures that deal simultaneously with multiple exponential family representatives in a Bayesian or likelihood-based framework and allow for grouping correction and epidemiological adjustment. When targeting inference the most used R packages are the **gRbase** (Dethlefsen, Højsgaard *et al.* 2005) and **gRain** (Højsgaard *et al.* 2012). Indeed, those R packages do not have any structural learning algorithm, thus the BN models should be fully provided by the user. But they allows to manipulate efficiently the models parameter to perform prediction and inference. The R package **abn** is the only library for BN modelling inference based on a fully Bayesian regression framework. It contains a unique implementation of an exact search algorithm. It targets systems epidemiology applied research in providing to the user extensive outputs easing biological findings interpretation and reporting. The Task View: gRaphical Models in R (`CRAN.R-project.org/view=gR`) gives a comprehensive overview of the different computing libraries available on CRAN.

Aside from R, multiple implementations in other computing environments such as Weka (Bouckaert 2008) (which is accessible in R via the R package **RWeka**, Hornik, Buchta, and Zeileis 2009), MATLAB (Murphy *et al.* 2001) or open source python or C++ implementation exist. In the epidemiological community, the R implementations are the most popular ones due to the simplicity of using R.

2. Methodological background

This section describes the theoretical foundations of ABN. First, we present the BN modelling paradigm. Then we extend this framework to describe the ABN methodology. Finally, we embed the ABN methodology into a learning scheme.

2.1. Bayesian Network

The idea of studying observational data using using a BN is quite old (Pearl 1985). Formally, a BN for a set of random variables $\boldsymbol{X} = \{X_1, \dots, X_n\}$ is a directed acyclic graph (DAG) where the nodes are the random variables and the directed links are the statistical dependencies between the nodes. A graph G is the union of two sets: the set of *nodes* or *vertices* and the set of *arcs* or *directed links*, *arrows* or *edges*. Thus: $G = (\mathbf{V}, \mathbf{E})$, where $\mathbf{V}$ is a finite set of *vertices* and $\mathbf{E}$ is a finite set of *edges*. An index node X_j is said to be the parent of a node i if the edge set $\mathbf{E}$ contains an edge from j to i. A set of parents for a node j is denoted by $\mathbf{Pa}_j$. Conversely, one can easily define the set of children $\mathbf{Ch}_j$ for a node j.

In a BN, the factorization of the joint probability distribution $\mathrm{P}(\boldsymbol{X})$ through a so-called set of local probability distributions is given by the *Markov property*, which implies that an index node X_j is dependent solely on its set of parents $\mathbf{Pa}_j$

$$P(\boldsymbol{X}) = \prod_{j=1}^{n} P(X_j \mid \mathbf{Pa}_j). \tag{1}$$

In equation (1), the total number of nodes is denoted by n. A BN model, $\mathcal{B}$, is the union of the structure $\mathcal{S}$ and the model parametrization $\boldsymbol{\theta}_{\mathcal{B}}$: $\mathcal{B} = (\mathcal{S}, \boldsymbol{\theta}_{\mathcal{B}})$. The edges represent both *marginal* and *conditional dependencies.* Collectively, they define the structure or network which encodes the conditional independence through graphical separation. Verma and Pearl (1988) shows that if two nodes are d-separated by a set of nodes, then the random variables are conditionally independent given the set of variables. This theorem is the starting point of a class of BN learning algorithms called constraint-based approaches. In a BN, each node X_j, with parent set $\mathbf{Pa}_j$, is parametrised by a local probability distribution: $P(X_j \mid \mathbf{Pa}_j)$. The choice of the parametrisation is the source of an alternative class of learning algorithms called score-based approaches. Two networks are said to be equivalent if they represent the same set of conditonal independence statements. A scoring system is said to be score equivalent if it assign the same score to equivalent networks. Score equivalence is a key notion for the score-based approach.

The Markov blanket (MB) of a node is the set of parents, children and co-parents, i.e., the parents of the specified children. An interesting property of the MB is that this is the set of nodes that fully inform an index node. For example, in the right network of Figure 1, the MB of index node `"Dyspnea"` consists of nodes `"Bronchitis"` and `"Either"`, whereas the MB of index node `"Either"` consists of nodes `"XRay"`, `"Dyspnea"`, `"Tuberculosis"`, `"LungCancer"` and `"Bronchitis"`. Indeed, a node is conditional independent of any non-descending node in a BN given its parents.

2.2. Additive Bayesian Network formulation

An ABN model, $\mathcal{A}$, is a graphical model that extends the usual generalized linear model (GLM) to multiple dependent variables through the factorization of their joint probability distribution (Lewis and Ward 2013). An $\mathcal{A}$ model assumes that each node is a GLM where the covariates are the parents and the distribution depends on the index node. Currently, the available data distributions are binomial(link=logit), Gaussian(link=identity), Poisson(link=log) and multinomial(link=logit) where the last distribution is available for MLE fitting only. The default link function are hard coded.

Given an index node X_j, a set of parents $\mathbf{Pa}_j$, and using the classical notation for the exponential family parametrization (Pitman 1936)

$$P(X_j \mid \mathbf{Pa}_j) = \exp\big(\eta(\theta_j)T(X_j, \mathbf{Pa}_j) - A(\theta_j)\big)\mathrm{d}H(X_j, \mathbf{Pa}_j), \tag{2}$$

where the functions η, T, A, H may be node-dependent (the indices omitted to simplify the notation) and where the parameters θ_j incorporate the configuration of the parents' node. For example, in the case of binary variables, i.e., $X_j \in \{0, 1\}$, equation (2) is simplified when using the classical logit link function to

$$P(X_j = 1 \mid \mathbf{Pa}_j) = \mathrm{logit}^{-1}(\theta_j) = \mathrm{expit}(\theta_j), \tag{3}$$

resulting in the classical logistic regression models for all the nodes. The multinomial random variables are fitted using Poisson log-linear models with coefficients zero for the first class.

The ABN modelling technique is situated in a small data analysis niche. It targets a dataset composed of variables issued from a mixture of different distributions. The main focus is that the final model's parameters should be biologically interpretable. An $\mathcal{A}$ is called additive in the sense that the expected change of the index node with respect to the set of parents is assumed to be additive on the exponential family link scale. Other Bayesian marginal likelihood scores, such as BDe do not assume additivity. As a direct consequence from assuming additivity, an $\mathcal{A}$ does not model interactions in the data, which could become problematic from a biological perspective if interactions are expected. Moreover, as this technic relies on the exponential family to parametrize the model, it implies rather bold assumptions in terms of number of parameters and thus fitting ability. Indeed, if we simply consider a single binary variable with k parents (and no other edges) then the number of parameters in an $\mathcal{A}$ model will scale linearly with k, while the saturated model scales as 2^k, e.g. with 7 parents, $p(X_j|\mathbf{Pa}_j)$ has 8 parameters versus 128 for the saturated model aka conditional probability table.

2.3. Learning algorithm

Many learning strategies have been proposed depending on the ultimate modelling goal being in concordance with the research question. To perform inference in BN, one needs a probabilistic model to compare the networks and a search algorithm to select the optimal structure. If we restrict ourselves to a purely non-dynamic network (which happens when using either temporally dependent data or a dynamical network) and observational data, we can propose two main strategies. The first is constraint-based approaches, where one learns the BN using statistical independence tests. The optimal network is identified using the reciprocal relationship between graphical separation and conditional independence (Spirtes 2001). The second popular approach is based on structural scoring, where each candidate network is scored and the one which has the highest score is kept. In practice, this is computationally intractable for the typical number of variables involved in a research project. The number of possible networks is massive and increases super-exponentially with the number of nodes (Robinson 1977). A practical workaround is to use a decomposable score, i.e., a score that is additive in terms of the network's nodes and depends only on the parents of the index node. This approach is very close to the classical model selection in statistics (Zou and Roos 2017).

The scoring approach paradigm requires that the scores represent how well the structure fits the data (Zou and Roos 2017). Many scores have been proposed for discrete BNs in a Bayesian context (Daly, Shen, and Aitken 2011) that aims at maximizing the posterior probability. Indeed, Heckerman, Geiger, and Chickering (1995) propose the so-called Bayesian Dirichlet (BD) family of scores which use a Dirichlet prior. The BD family regroups the K2, BDeu, BDs and BDla scores (Scutari 2018). For continuous BN with multivariate Gaussian data, using an inverse Wishart prior leads to the BGe score (Consonni and Rocca 2012). By analogy, the scores can be adapted to a mixture of variables such as information theoretic scores. They are less suitable from a theoretical perspective but more polyvalent regarding the possible data distribution. Scores within a frequentist framework have been proposed (Daly *et al.* 2011), such as Bayesian Information Criterion (BIC), Akaike Information Criterion (AIC) and Minimum Description Length (MDL) (Daly *et al.* 2011). They all have in common a goodness-of-fit part and a penalty for model complexity. A direct and natural extension of this idea, implemented in the R package **abn**, uses the posterior score in a Bayesian regression settings. When applied to BN scoring, it is called marginal likelihood (MacKay 1992), as the likelihood is marginalized in the parameter space for the estimation using the regression framework for the parametrization. In a Bayesian setting, the parameter prior acts as a penalty term.

The model learning phase involves two parts: 1. Network, skeleton or structure learning

$\mathcal{S}$; and, 2. Parameter learning where the model parameter is $\boldsymbol{\theta}_\mathcal{B}$. Hence, in a Bayesian framework, constructing an ABN model $\mathcal{A}$ given a set of data $\mathcal{D}$

$$P(\mathcal{A} \mid \mathcal{D}) = \underbrace{P(\theta_\mathcal{A}, \mathcal{S} \mid \mathcal{D})}_{\text{model learning}} = \underbrace{P(\theta_\mathcal{A} \mid \mathcal{S}, \mathcal{D})}_{\text{parameter learning}} \cdot \underbrace{P(\mathcal{S} \mid \mathcal{D})}_{\text{structure learning}} . \tag{4}$$

The two learning steps are interconnected. Several efficient algorithms have been proposed for both learning procedures. In order to learn the relationships between variables, the conditional probability distributions should be fitted. In a frequentist setting and for the Gaussian, Bernouilli and Poisson distributed variables, this can be done using the classical Iterative Reweighed Least Square (IRLS) algorithm (Faraway 2016). The multi-categorical random variables are fitted through a softmax function with the maximization of the conditional likelihood. The structure selection step can be done using a heuristic or exact approach.

An interesting feature of the ABN methodology is to be able to impose external expert knowledge. Indeed, in most of applied data analysis, some part of the network is known. For example, if two random variables are temporally related the direction of the possible arrow is known. Or if, based on existing literature a possible connection is known to be expected. The R package **abn** allows such external causal input through a banning or a retaining matrix. Those matrices are used to compute the list of valid parent combination. However, a more theoretically sounding approach, suggested by Heckerman *et al.* (1995), is to augment the observed data with synthetic data that represents the causal belief. The practical feasibility in an ABN analysis remains an open question (McCormick *et al.* 2013).

3. The R Package abn

This section describes the functionalities of the R package **abn**. We provide insights about how the functions are implemented. We show simulations that compare the model parameter estimations using a Bayesian or an MLE approach and how different scores perform to retrieve networks.

The R package **abn** has three level of functionalities for different purposes:

- **Core functions:** aiming at performing an ABN analysis (scoring and learning)
- **Ancillary functions for analysis:** aiming at supporting the analysis by enhancing plotting abilities, accounting for the uncertainty in the structure through link strength estimation, and comparing structure or getting structural metrics
- **Ancillary functions for simulation:** aiming at helping in simulating ABN models by simulating DAGs and ABN data.

A classical ABN analysis is the sequential application of three R functions see Section 1.1 (typically: `buildscorecache()`, `fitabn()` and `mostprobable()`). The reason why the analysis is divided into three functions is to give more freedom to the end user to tune parameters and to have a better control over possible learning/fitting issues.

3.1. Set of core functions

The two main core functions for scoring are `buildscorecache()` and `fitabn()`. The former is essentially a wrapper of the latter. Those functions have a Bayesian and an MLE implementation which are not equivalent in their output (there is an argument `method` that can be `bayes` or `mle` to choose the implementation). They require minimally a named dataset, the named list of the nodes' distribution and an upper limit for network complexity. `buildscorecache()` first computes an empty list of valid parent combinations using banning and retaining input matrices (which are assumed to be empty by default). Then it iterates through the cache to score the candidate piece of network. In the Bayesian implementation, `buildscorecache()` and `fitabn()` estimate a Bayesian regression using the following parameter priors: weekly informative Gaussian priors with mean 0 and variance 1000 for each of the regression parameters of the model (both binomial and Gaussian), as well as diffuse Gamma priors (with shape and scale equal to 0.001) for the precision parameters in Gaussian nodes in the model using an internal INLA code.

`buildscorecache()` within an MLE setting uses an IRLS algorithm depending on the given list of distributions at each step of the scoring process. For the special case of the binomial nodes, the usual logistic regression is tried. If it fails to estimate the given problem, a bias-reduced tailored algorithm is used (Firth's correction). It is know to improve accuracy of regression coefficients in presence of separation (van Smeden, de Groot, Moons, Collins, Altman, Eijkemans, and Reitsma 2016). If, however, the algorithm fails to return a finite estimate, some predictors are sequentially removed until the design matrix becomes fully ranked. These three steps ensure the R package **abn** to be able to score a dataset even if there is data separation. The multi-categorical random variables can also be estimated through IRLS like algorithm. But a well known weakness of this approach is the very high per-iteration cost due to sparsity of the intermediate matrices. However, fast estimation of multinomial logit models that efficiently takes advantage of the matrix structures have been proposed (Hasan, Wang, and Mahani 2016). It overshoots the purpose of **abn** as it requires tailored parametrisation, which is not feasible in a fast fully automated scoring procedure. An alternative approach to estimate multinomial logistic regressions is to approximate it using multiple sequential binary logistic regressions. The primary purpose of the scoring procedure is to estimate goodness-of-fit metrics, then special care should be taken to compute the log-likelihood as each observations are counted multiple times. This estimation step substantially slows down the estimation process. A robust, easy to implement and fast solution to estimate unregularized multinomial regression models is to use neural networks with no hidden layers, no bias nodes and a softmax output layer. The optimisation is done through maximum conditional likelihood. This procedure is implemented in R by the **nnet** package (Venables and Ripley 2002) and is the chosen solution for multi-categorical estimation in **abn**.

The `fitabn()` function score a given network. It requires a valid DAG, a named dataset and a named list of distributions. It returns the list of score for each node, the parameter estimates, the standard deviation and the p-values. Special care should be taken when interpreting and displaying the p-values. Indeed, the DAG has been selected using the goodness of fit metric, so at least adjustment methods should be used.

A unique feature of `fitabn()` and `buildscorecache()` is the possibility to take one-layer clustering into account. In some situations, data collection has a clear grouping aspect. Therefore, there is a potential risk for non-independence between data points from the same group that could cause over-dispersion. This can lead to analyses which are over-optimistic, as the true level of variation in the data is under-estimated.

To account for this additional variability, a random effect is introduced. Thus, each node becomes a Generalized Linear Mixed Model (GLMM) (Breslow and Clayton 1993; Faraway 2016) instead of a GLM, but in a Bayesian setting (see Section 4.6 for more details). This implies introducing an additional correlation structure via random effects, i.e., adjusting for correlated residuals or non-constant variance. The rationale is that standard errors are underestimated during the first search so the DAG produced during the structure search without random effect have more rather than less structure. That structure could then be trimmed during the MCMC part.

We therefore compute the posterior distribution and check if it widens. In such a case, we must take the clustering in the scoring scheme used into account. From an applied perspective, the major limitation is the computational burden of this approach. Indeed, if the clustering is unlikely to impact the estimates, it is preferable not to take it into account. The model is then much simpler and parsimonious. The clustering adjustment could be performed in a subset of the nodes.

For learning BN, two types of algorithms are implemented: exact and heuristic searches. The exact procedure, `mostprobable()`, runs the exact order-based structure discovery approach of Koivisto and Sood (2004) to find the most probable network. Its input is a cache of pre-computed scores as from `buildscorecache()`, the desired score and structural prior to use. As described in Koivisto and Sood (2004), `mostprobable()` uses dynamic programming to marginalize orders analytically. It cannot handle more than 25 nodes, but on similar size problem, it outperforms any MCMC and probabilistic approaches. It is implemented with two structural priors: the koivisto prior that states that different cardinalities of parents are considered to be equally likely a priori; and a structurally uninformative where parent combinations of all cardinalities are equally likely. The heuristic search algorithms implemented are the hill-climber, the Tabu and the simulated annealing in the function `searchHeuristic()`. The function requires a cache of pre-computed scores, the desired score and some method-dependent arguments.

3.2. Defining a DAG

To specify a DAG, the R package **abn** uses an adjacency matrix, i.e., a named square matrix with an entry in the ijth entry if there is an arc from j (parent) to i (child).

The R package **abn** also recognizes a formula-like syntax, similar to the classical R functions `lm`, `glm`, etc. A typical formula is `~ node1|parent1:parent2 + node2:node3|parent3`. The formula statement has to start with `~`. In this example, `node1` has two parents (`parent1` and `parent2`). `node2` and `node3` have the same `parent3`. The parent'ss names have to exactly match those given in data.df. The symbol `:` is the separator between either children or parents, the symbol `|` separates children (left side) and parents (right side), the symbol `+` separates terms and `.` replaces all the variables in `data.df`. Due to this feature, those symbols cannot be used in the names of the random variables. The banned matrix could be produced by the following statement `~ female|.`, which would prevent the variable female from having any parents. Currently it is not possible to incorporate interaction terms in this formula statement.

3.3. Ancillary functions for analysis

`plotabn()` and `tographiz()` are useful functions for plotting DAGs. `plotabn()` can display a DAG with fitted values and arrows with thickness proportional to arc strength.

The concept of link strength for discrete BN was introduced in Boerlage (1992). A good unpublished overview is given by Ebert-Uphoff (2009). The link strength methodology is especially useful to account for the uncertainty when using BN to model. Indeed, an arc is either present or absent. This strong dichotomisation of structural relations in BN is often hard to interpret. The link strength tends to give a proxy for arc support by the data and in an applied perspective is complementary to the regression coefficients. In practice, the function `linkStrength()` discretizes the dataset using a large choice of histogram rules to compute multiple link strength metrics. Then, the estimates of the empirical distribution are plugged in the definition of the entropy to return the so-called empirical entropy. A well-known problem of empirical entropy is that the estimations are biased due to the sampling noise. It is also known that the bias decreases as the sample size increases. The mutual information estimation is computed from the observed frequencies through a plug-in estimator based on the entropy.

Two functions are useful for comparing DAGs. The `compareDag()` function returns multiple graph metrics to compare two DAGs: the confusion matrix, or error matrix in the machine learning literature (Stehman 1997). `compareDag()` provides a list with the true positive rate, the false positive rate, the accuracy, the G-measure, the F1-score, the positive predictive value, the false omission rate and the Hamming-Distance. The `infoDag()` returns a list for standard metrics for describing a DAG that contains the number of nodes, the number of arcs, the average Markov blanket size, the neighborhood average set size, the parent average set size and children average set size.

The R function `scoreContribution()` computes the score contribution of each individual observation to the total network score and additionally returns the diagonal entries of the hat matrix. This function attempts to produce influential measures adapted to Additive Bayesian Networks models. In a regression context, this is possible for a single or a small group of observations to have a great influence on the results of a regression analysis. Therefore it is important in an applied perspective to detect influential observations and to take them into consideration when interpreting the results (Belsley, Kuh, and Welsch 2005).

3.4. Ancillary functions for simulation

To simulate DAGs and ABN data, the functions `simulateDag()` and `simulateAbn()` are provided. The function `simulateDag()` generates DAGs with an arbitrary arc density. To ensure acyclicity, it samples a triangular adjacency matrix. The arc density is tuned with a binomial sampling probability. The `simulateAbn()` generates ABN data using the R package **rjags** (Plummer 2018). Simulating observations from a given structure is done with a random number generator, respecting the node ordering using JAGS (Plummer *et al.* 2003). It first creates a BUG file in the index repository, then uses it to simulate the data. This function produces a data frame. The purpose of those two functions is to assess the effectiveness of the other functions of the R package **abn**. But they could also be used to plan and conceive systems epidemiology studies in assessing the necessary number of samples as a function of the expected effect size.

4. Case study

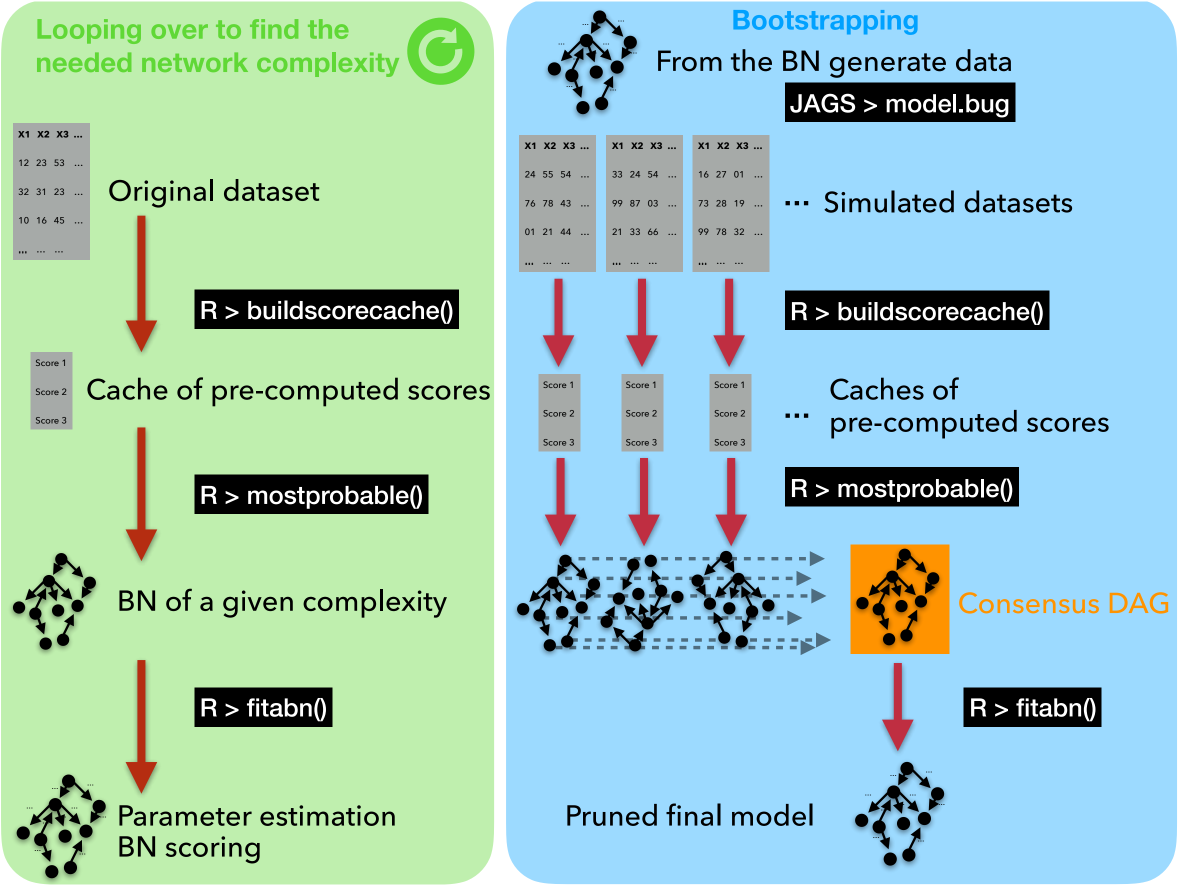

Figure 2: Schematic workflow diagram for a typical ABN analysis. In the green square is the typical workflow for an ABN analysis. The blue square describes the bootstrapping procedure.

The purpose of this case study is to perform a fully reproducible and transparent analysis of an open access observational dataset. The main objective is to produce a reliable network together with the necessary information to allow applied researchers to interpret and report it. The final model is controlled for over-fitting, and some strategies for controlling for clustering are presented. The code for the analysis is provided to help disseminate the ABN approach in the systems epidemiology community by addressing every step from an applied and interpretative perspective. The proposed procedure for performing an ABN analysis could be quite complex for a new user. Figure 2 shows in green the workflow diagram of a classical ABN analysis. It is the sequential application of three functions: `buildscorecache()` for computing a cache of pre-computed scores; a search algorithm such as `mostprobable()` or `searchHeuristic()`; and `fitabn()` for fitting the final model to the data. In blue, the bootstrapping workflow is presented. It is very similar to a classical ABN analysis, except that it is based on simulated datasets to control for possible over-fitting.

4.1. Data description and library loading

The case study dataset collected in March 1987 is about growth performance and abattoir findings in the commercial production of pigs in 15 Canadian farms (Dohoo, Martin, Stryhn *et al.* 2003). The data were collected to study inter-relationships among respiratory diseases (atrophic rhinitis and enzootic pneumonia), ascarid level and daily weight gain. This dataset is well adapted to show the unique ability of BN modelling to disentangle complex relationships with observational datasets. The data is an adapted version of the original dataset. It consists of 341 observations of the 9 variables described in Table 1. A nice feature of this dataset is that it is composed of continuous, discrete and count distributed variables. Furthermore, the dataset has a natural grouping feature due to the different farms. This is used to showcase the capability of the R package **abn** to control for a grouping effect.

Table 1: Description of the variables.

Variable	Meaning	Distribution
AR	presence of atrophic rhinitis	Binomial
pneumS	presence of moderate to severe pneumonia	Binomial
female	sex of the pig (1=female, 0=castrated)	Binomial
livdam	presence of liver damage (parasite-induced white spots)	Binomial
eggs	presence of fecal/gastrointestinal nematode eggs at time of slaughter	Binomial
wormCount	count of nematodes in small intestine at time of slaughter	Poisson
age	days elapsed from birth to slaughter (days)	Continuous
adg	average daily weight gain (grams)	Continuous
farm	farm ID	Discrete

The R package **abn** and its dependencies are available from CRAN. To fully profit from the case study presented below, other software needs to be installed: Rgraphviz (Hansen, Gentry, Long, Gentleman, Falcon, Hahne, and Sarkar 2019) from bioconductor; or alternatively the R package **DiagrammeR** (Iannone 2019) and the JAGS computing library (Plummer *et al.* 2003). One can load the data using the following code

```
R> data("adg", package = "abn")
```

4.2. Data Preparation and package loading

Each variable in the model needs to be associated with a distribution. Thus, one needs to create a named list that contains all the variables and the corresponding distributions. As a remainder, the available data distributions are binomial, Gaussian, Poisson and multinomial, where the last distribution is available for MLE fitting only. The data of this example is well defined, but in general one needs to coerce binary or multinomial variables to factors

```
R> dist <- list(AR = "binomial", pneumS = "binomial",
```

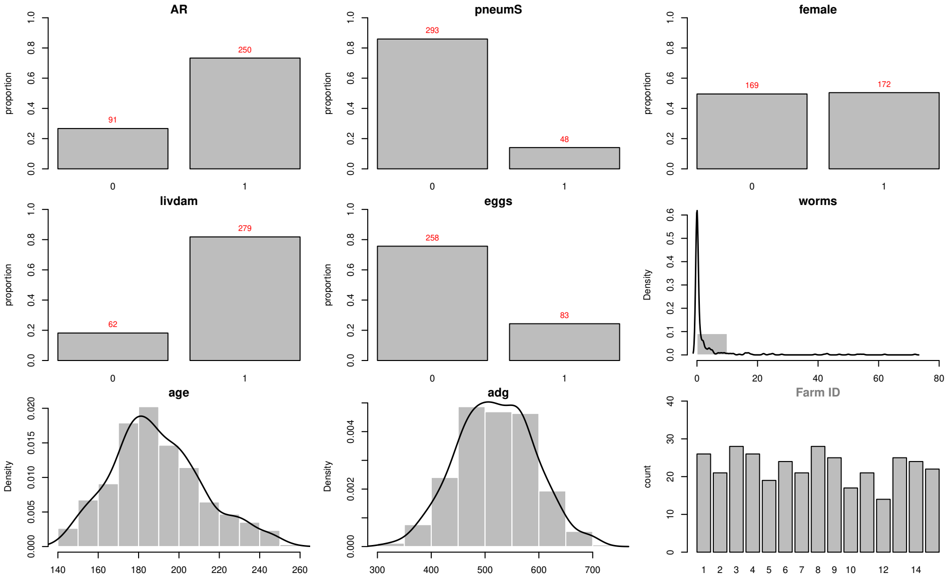

Figure 3: Descriptive distributions of the variables.

```
R+      female = "binomial", livdam = "binomial", eggs = "binomial",
R+      wormCount = "poisson", age = "gaussian", adg = "gaussian")
```

A nice feature of the R package **abn** is the ability to input prior information about structural beliefs about the data to guide the structure search. Indeed, prior expert knowledge is common in systems epidemiology. In this case study, it is reasonable to assume that none of the variables in the model are going to affect the sex of the animal, an inborn trait. To encode this information, we ban all the arcs going to `female` by setting their value to 1 (banned), opposite to the default value 0 (not banned) in an adjacency matrix-like formulation. The rows are children and the columns are banned parents of the index nodes.

```
R> print(banned)
```

```
          AR pneumS female livdam eggs wormCount age adg
AR         0      0      0      0    0         0   0   0
pneumS     0      0      0      0    0         0   0   0
female     1      1      0      1    1         1   1   1
livdam     0      0      0      0    0         0   0   0
eggs       0      0      0      0    0         0   0   0
wormCount  0      0      0      0    0         0   0   0
age        0      0      0      0    0         0   0   0
adg        0      0      0      0    0         0   0   0
```

By default, the R package **abn** assumes no banned nor retained arcs. See Section 3.2 on how to specify banned or retained arcs by using a formula-like syntax.

4.3. Structure search

For computational reasons, it is advised to constrain the maximum number of parents allowed for each node. We start to compute a cache of pre-computed scores with a single parent per node and increase subsequently the number of parents until the network score of the optimal structure does not get larger, even when more parents are allowed. Based on the cache of pre-computed scores, an exact search using the function `mostprobable()` is performed and the network score is computed. In R this is done using a simple `for` loop with the functions `buildscorecache()`, `mostprobable()` and `fitabn()`.

```
R> result <- vector("numeric", 7)
R>
R> # ban: formula statement retain: not constrained
R> for (i in 1:7) {
R+     mycache <- buildscorecache(data.df = as.data.frame(abndata),
R+         data.dists = dist, dag.banned = ~female | .,
R+         dag.retained = NULL, max.parents = i, method = "bayes")
R+     mydag <- mostprobable(score.cache = mycache)
R+     result[i] <- fitabn(object = mydag)$mlik
R+ }
```

Figure 4 displays the network score as a function of the number of parents. The maximum log marginal likelihood (–2709.25) is achieved with 4 parents. As one can see, the network score increases quickly at the beginning but then plateaus. A strong assumption is to say that if increasing the network complexity by one does not change the network score, then the maximum was achieved. Indeed, nothing prevents the maximum network score being constant when increasing the number of parent but still changing further when increasing again. An alternative is to run an analysis with a pre-defined network complexity. Another approach is to use a heuristic search provided by `searchHeuristic()`. The heuristic approaches, contrary to the exact search, do not guarantee the achievement of the optimal network score. These approaches, however, do not suffer from computing limitations, thus they are the only viable solution for large problems. Many efficient heuristic search algorithms are implemented in **bnlearn** (Scutari 2010).

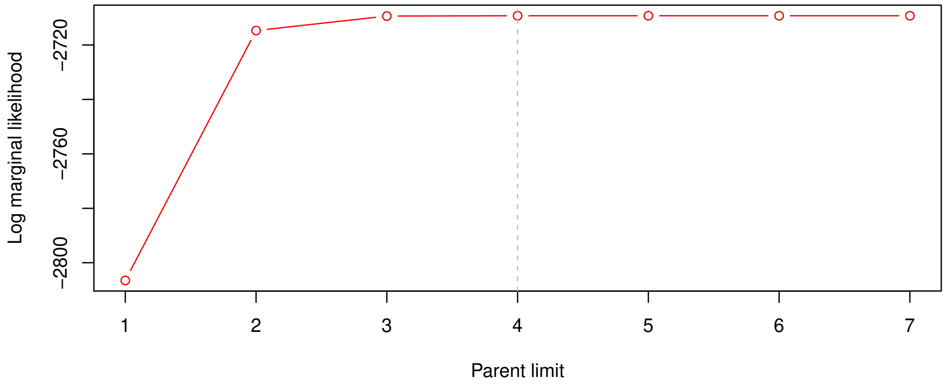

Figure 4: Total network log marginal likelihood as a function of the number of parents.

Figure 5 shows the DAG selected using `mostprobable()` with a model complexity of 4 parents. This DAG has 10 arcs for 8 variables, so this is a relatively sparse model (the average number of parents per node is 1.25, the average size of the Markov blanket is 3.75 and each node has on average 2.5 neighbors; as returned by function `infoDag()` or `summary()`). The square nodes are conditionally Bernoulli distributed, the oval nodes are conditionally normally distributed and the diamond node is conditionally Poisson distributed.

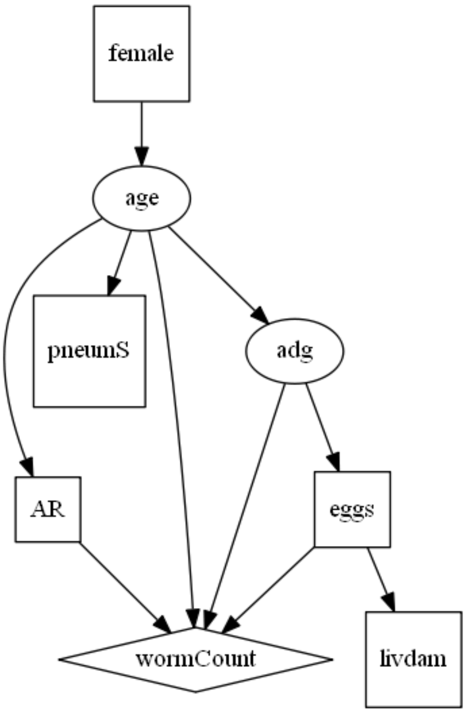

Figure 5: DAG selected using an exact search with a model complexity of four parents.

4.4. Control for over-fitting

One major concern in BN modelling is the tendency for over-fitting. Indeed, the number of observations is generally very limited in comparison to the number of parameters in the model. The number of possible models, i.e., DAGs, increases super-exponentially with the number of random variables (Robinson 1973). The possible number of DAGs with 25 nodes are about a googol (10^{100}).

More conceptually, the question remains of how to present the result of a BN modelling analysis. Multiple strategies have been proposed to create a summary network. One could perform multiple heuristic searches. One common and very simple approach is to produce a single robust BN model of the data mimicking the approach used in phylogenetics to create majority consensus trees. A majority consensus DAG is constructed from all the arcs present in at least 50% of the locally optimal DAGs found in the search heuristics. This creates a single summary network. The function `searchHillclimber()` performs such an analysis. Rather than using the majority consensus network as the most appropriate model for data, an alternative approach would be to choose the single best model found during a large number of heuristic searches using `searchHeuristic()`, or if possible an exact search using `mostprobable()`.

The sensible step of this approach is to set the necessary number of searches needed to be run to provide reasonable coverage of all the features of the model landscape. Then this model should be adjusted for possible over-fitting. Like the majority consensus network, which effectively averages over many different competing models and therefore should generally select only robust structural features, single best DAG strategy should be controlled for over-fitting. Indeed, choosing the DAG from a single model search is far more likely to contain some spurious features, especially when dealing with small datasets of around several hundred observations. It is extremely likely to over-fit, as one can easily demonstrate using simulated data.

A simple assessment of over-fitting can be made by comparing the number of arcs in the majority consensus network with the number of arcs in the best-fitting model found using an exact search or a large number of heuristic searches. We have found that in larger datasets, the majority consensus and best-fitting model can be almost identical, while in smaller datasets the best-fitting models may have many more arcs, suggesting a degree of over-fitting.

An advantage of choosing a DAG from an individual search is that, unlike averaging over lots of different structures as in the construction of a majority consensus network, the model chosen here has a structure which was actually found during a search across the model landscape. In contrast, the majority consensus network is a derived model which may never have been chosen during even an exhaustive search. Indeed, it may comprise contradictory features, a usual risk when averaging across different models. In addition, a majority consensus network need also not to be acyclic, although in practice this can be easily corrected by reversing one or more arcs to produce an appropriate DAG. A simple compromise between the risk of over-fitting when choosing the single highest-scoring DAG and the risk of inappropriately averaging across different distinct data generating processes is to prune the highest-scoring DAG using the majority consensus model. In short, element-wise matrix multiplication of the highest-scoring DAG and the majority consensus DAG gives a new DAG which only contains the structural features in both models. An alternative approach for tackling this problem, would be to use Bayesian model averaging. Structural MCMC will be discussed in Section 6.

Friedman, Goldszmidt, and Wyner (1999) presented a general approach for using parametric and non-parametric bootstrapping to select BN models/DAG structures. They showed that a non-parametric approach seems to converge less rapidly in terms of number of samples but requires fewer assumptions. The non-parametric bootstrapping approach is relatively easy to implement and efficient in extracting robust features from the data. The parametric approach can be implemented by using readily available Markov chain Monte Carlo sampling software such as JAGS or WinBUGS. The basic idea is to take the structure with the best network score, code it, and use these samplers to generate bootstrap datasets from this model. That is, independent realizations from the model which can be used to generate datasets of the same size as the observed data. Then the model search is repeated treating the bootstrap data as the observed data. Generating many bootstrap datasets and conducting searches on each allows us to estimate the percentage support for each arc in the highest-scoring model. In other words, we find out how many of the arcs in the highest-scoring model can be recovered from a dataset of the size that was actually observed. Obviously, the more data, the more statistical power and recoverable structural features. Arcs with a lower level of support, e.g., <50%, can be pruned from the best fitting model, assuming that these are potentially a result of over-fitting. The resulting model - possibly with arcs pruned from the original model - is the chosen model for the data. The 50% threshold is arbitrary and could possibly depend on the expected arc density or model complexity.

An obvious limitation of the parametric bootstrapping method is the fact that the adjustment for over-fitting transfer the assumptions and limitation of the original method to the simulated data. An ABN model assumes the absence of interactions in conditional distributions (i.e. $p(X_j|\mathbf{Pa}_j)$) which might be present in the underlying data. When using non-parametric approach the original data are used and the feature not captured by the model, are still present. Other options for trimming or pruning arcs exist, but without further information it seems to be a reasonnable uninformative guess. This strategy is commonly used in phylogenetic trees.

While parametric bootstrapping as a general technique is well established and conceptually elegant, it may in practice not be computationally feasible. Even when taking the least demanding approach of conducting only one heuristic search per bootstrap dataset, the number of datasets/searches required to get robust support values for each arc in the best fitting model may be large and beyond what is reasonably possible even using high performance computing (HPC) hardware.

4.5. Parametric bootstrapping

The chosen dataset allows us to do fast parametric bootstrapping. Given a BN model - a DAG structure plus parameter priors - the first step is to estimate the posterior parameters. The second step is to implement the DAG structure together with the posterior parameters in a BUG file. To make the implementation as general as possible, we present an approach based on `dcat()` from JAGS that discretizes each posterior density across a fine grid.

The function `fitabn()` with `compute.fixed = TRUE` uses Laplace approximations to estimate the posterior density of each parameter in a BN model. An appropriate domain (range) for each parameter must be supplied by the user using a bit of trial and error. It is crucially important to give a sufficiently wide range so that all of the upper and lower tails of the distribution are included, e.g., the range should encompass where a density plot first drops to approximately zero at each tail.

Figure 6 compares one round of simulated data (in blue) with the original data (in black). By default, `fitabn()` center and standardizes continuous data. As one can see, the simulated data looks fairly good except for the count data, which looks sub-optimally simulated as the long right tail is under-represented in the simulated data. Figure 6 is only comparing margins and not joint distributions.

The code displayed in Annex C performs 5000 bootstrapping steps from a model defined in a BUG file called `model8vPois.bug` using the R package **rjags**. A thinning of 20 is used to reduce auto-correlation in the simulated samples. The global search is depicted with the pseudo-code given in Algorithm 1.

In the following text, the list of DAGs selected from the simulated datasets is called the bootstrapped samples. In order to simulate the variables of the dataset, we need to provide a model for each of them using the aforementioned parameters estimates. For instance, the binomial node `AR` in our DAG has one incoming arc coming from the node `age`. In a regression setting this would be translated into:

$$\text{logit}(\texttt{AR} = 1 \mid age) = \alpha + \beta \cdot \texttt{age} + \varepsilon, \tag{5}$$

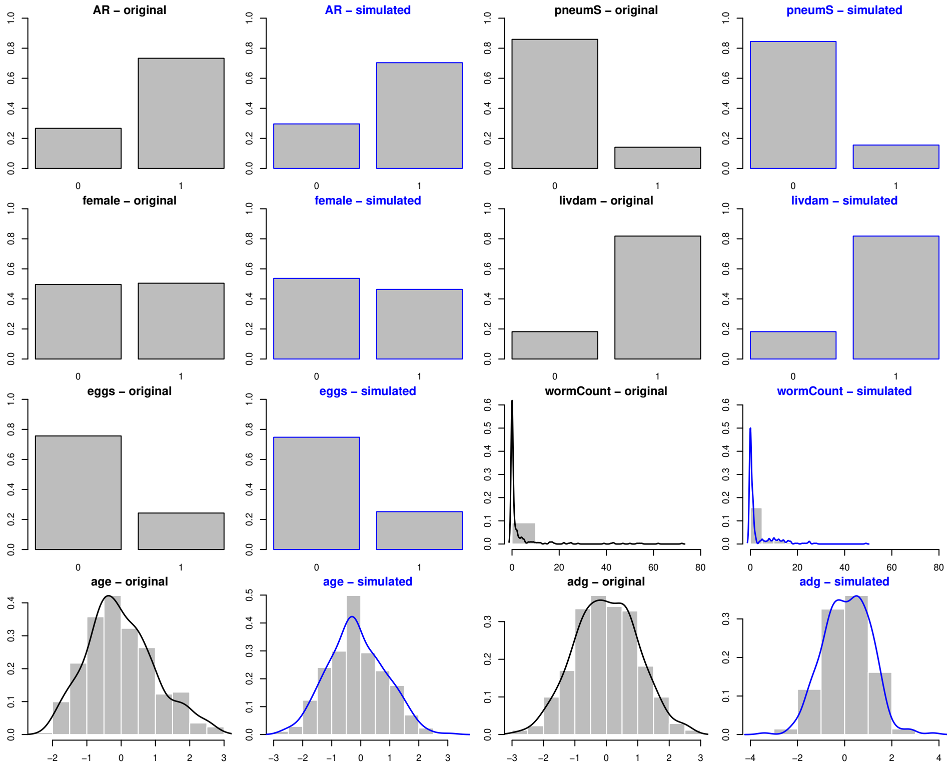

Figure 6: Example of simulated data (in blue) versus original data (in black).

where α is the intercept, β is the regression coefficient for variable `age`, and ε is the error term modeled by a binomial distribution. Then, the values of α and beta is sampled using the function `dcat()` within JAGS from the original discrete distribution of parameters generated by `fitabn()`.

Figure 7 displays the arc distribution from the selected BNs from the simulated datasets, i.e., the bootstrapped samples. The network selected from the original dataset had 10 arcs. This is an indication of over-fitting from the original model. Or alternativelly it could be due to interactions between parents presentin the empirical conditional distributions that by construction are not present in the ABN.

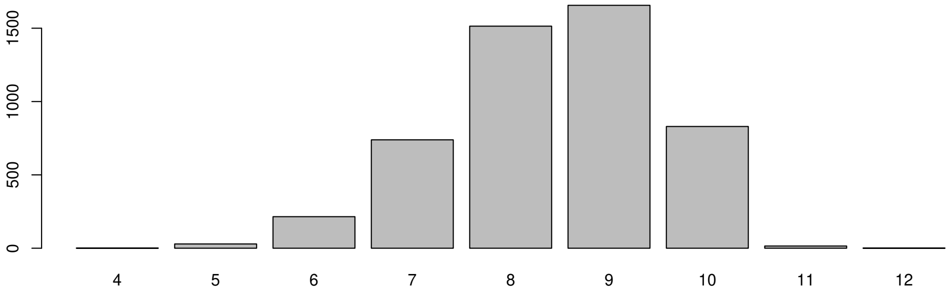

Figure 7: Histogram of the distribution of the number of arcs in the bootstrapped searches

In order to produce the final pruned DAG, we measure the prevalence of each arc in the generated structures and retain only arcs present in at least 50% of the bootstrapped samples.

```
Result: A list of DAGs and bootstrapped dataset
Input  : A fitted ABN model, an original dataset, model complexity limit, a random
         seed, a number of desired bootstrap runs, a BUG file model, a thinning
         parameter
Output: List of matrices and dataframe
for 1:n do
    Read a BUG file and run JAGS over it
    Extract simulated samples
    Format data
    Compute a cache of pre-computed scores
    Run an exact search
    Fit an ABN model to the selected BN
end
```

Algorithm 1: Pseudo-code for bootstrapping ABN models.

The matrix displayed below shows the percentage of arcs retrieved within the bootstrap samples. As one can see, some arcs are clearly supported or excluded, but some are very close to the 50% threshold. This is not surprising given the typical number of observations of an epidemiological study (a few hundred). An alternative approach would be to consider an undirected graph that includes all arcs supported regardeless to direction in more than 50% of the bootstrap samples. Such approach will rely on the idea that ABN methodology is acausal and the existence of a link is more important than the direction of the arc. This decision is likely to be problem-specific.

```
R> print(perdag * 100, digits = 0)
```

```
          AR pneumS female livdam eggs wormCount age adg
AR         0      0      1      0    0         0  76   5
pneumS     1      0      0      0    0         0  40   8
female     0      0      0      0    0         0   0   0
livdam     0      0      0      0   53         2   0   0
eggs       0      0      0     25    0         0   8  75
wormCount 71      1      1      0  100         0 100  60
age       18     17     56      0    3         0   0  26
adg        1      2     11      0    6         0  74   0
```

4.6. Control for clustering

In the `adg` dataset, the data were collected in different farms (random variable encoded as `farm`). A workaround was needed to introduce additional correlation structure via random effects when fitting the data to the structure to avoid overly optimistic parameter estimates. Three strategies for clustering adjustment are possible in an ABN analysis. They are ranked in terms of computational complexity:

- **Adjustment at the regression phase:** given the structure selected without adjustement (eventually controlled for overfitting), it is possible using the function `fitabn()` to adjust the regression coefficients for clustering.
- **Adjustment at the bootstrapping phase:** given the structure selected without adjustment, one can alter the code aiming at controlling for overfitting in adding a Gaussian random effect in the model.
- **Adjustment at the learning phase:** learning the structure with cluster adjusted scores with the function `buildscorecache()`.

Adjustment at the regression phase

The function `fitabn()` has the possibility to control internally for clustering using the arguments group.var and cor.var. One can choose which variable should be adjusted. On a personal computer, computing the adjusted model takes 15 minutes to compute, whereas the unadjusted model ran in less than a second. The grouping variable has 15 levels. In the function `fitabn()`, we apply the grouping adjustment to all random variables.

The following code will produce adjusted regression coefficients:

```
R> marg.f.grouped <- fitabn(object = mydag, group.var = "farm",
R+     cor.vars = c("AR", "livdam", "eggs", "wormCount", "age",
R+         "adg"), compute.fixed = TRUE, n.grid = 1000)
```

Adjustment at the bootstrapping phase

In the BUG file, one can add a random effect for each level of the clustering variable (M=15; the number of farms). The random effects are typically chosen as normally distributed with mean zero and precision parameter determined with a diffuse gamma prior. In practice one modifies the BUG file with a for loop over the number of levels of the clustering variable (M=15; the number of farms) for each variable that should be adjusted. And a list of priors for precision parameters. This approach aims at verifying which of the selected arcs are robust enough to pass the bootstrapping phase when correction is applied to account for the additional variance produced by the clustering.

```
$> for(j in 1:M){\newline
$  % random effect for each group for variable adg\newline
$  rv.adg[j] ~ dnorm(0.0,prec.rv.adg);\newline
$}\newline
$> % priors definitions\newline
$> % rv priors\newline
$>\newline
$> prec.rv.adg ~ dgamma(1,5E-05);\newline
```

Adjustment at the learning phase

The finest adjustment is done directly at the learning phase when pre-computing the network scores. This is also the most computationally demanding and it can become numerically unstable given the number of models that should be computed. This is done using the group.var and cor.var arguments in `buildscorecache()`. In the following code the same structural constraints have been used and the number of possible parent is limited to 4.

```
R> # recompute the cache of scores using GLMM
R> mycache <- buildscorecache(data.df = abndata, data.dists = dist,
R+     group.var = "farm", cor.vars = c("AR", "pneumS",
R+         "female", "livdam", "eggs", "wormCount", "age",
R+         "adg"), dag.banned = ~female | ., dag.retained = NULL,
R+     max.parents = 4)
R>
R> # exact search
R> dag.adjusted <- mostprobable(score.cache = mycache)
R>
R> # recompute the marginal using GLMM
R> marg.f.grouped <- fitabn(object = dag.adjusted, group.var = "farm",
R+     cor.vars = c("AR", "pneumS", "female", "eggs",
R+         "wormCount", "age", "adg"), compute.fixed = TRUE,
R+     n.grid = 100, control = list(max.mode.error = 0,
R+         epsabs.inner = 0.1, max.hessian.error = 0.5,
R+         epsabs = 0.1, error.verbose = FALSE, hessian.params = c(0.01,
R+             0.1), factor.brent = 10, loggam.inv.scale = 0.1))
```

Figure 8 shows the parameter distribution adjusted for clustering using random effect. The adjustment has been applied to every node except the node female. After postproceesing, the random effect applied to this node is poorly estimable. Indeed, the number of female versus castrated pigs across farms is the fairly similar, which implies that the clustering has a negligible effect. As one can see when comparing figures 8 and 13, the quantiles have not widened much.

4.7. Accounting for uncertainty in BN models

Calculating the so-called link strength is useful for both visualization and approximate inference, and it can be seen as a proxy for arc uncertainty. The strength of the edges is a complementary metric to regression coefficients. We use a link strength metric called the true average link strength percentage (PLS), which expresses by how many percentage points the uncertainty in variable Y is reduced by knowing the state of its parent X if the states of all other parent variables are known (averaged over the parent states using their actual joint probability). The actual definition is:

$$PLS(X \to Y) = \frac{H(Y|Z) - H(Y|X,Z)}{H(Y|Z)}. \tag{6}$$

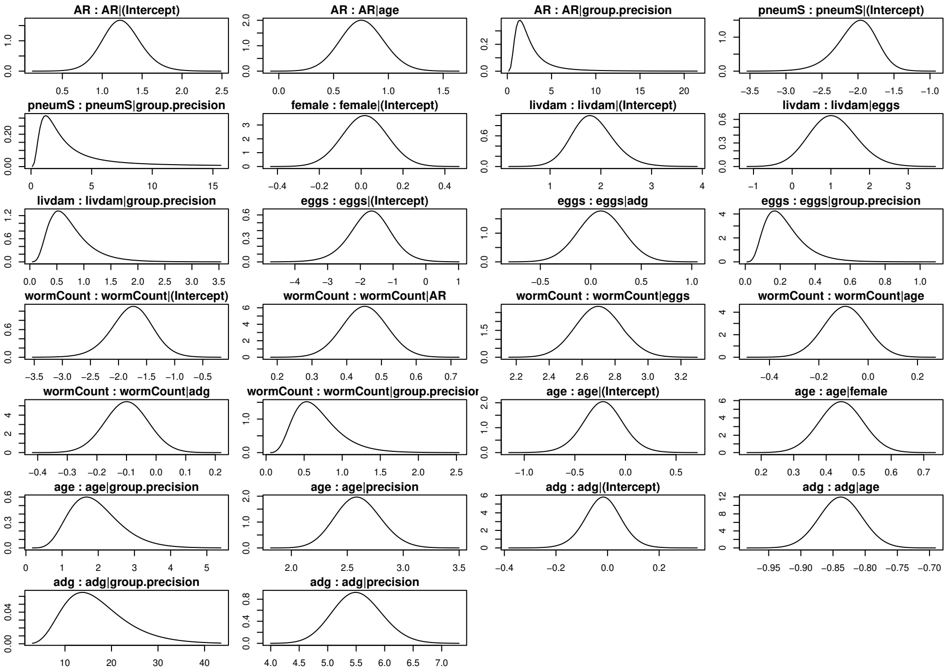

Figure 8: Marginal densities of model parameter corrected for random effect.

For the case of an arc going from node X to node Y and where the remaining set of parents of Y is denoted as Z. H is the entropy computed using the empirical distribution. The link strength is set to zero if there is no arc going from X to Y. The matrix below displays the percent link strength:

```
R> print(LS, digits = 3)

             AR pneumS female livdam   eggs wormCount   age   adg
AR        0.000      0 0.0000      0 0.0000         0 0.089 0.000
pneumS    0.000      0 0.0000      0 0.0000         0 0.000 0.000
female    0.000      0 0.0000      0 0.0000         0 0.000 0.000
livdam    0.000      0 0.0000      0 0.0405         0 0.000 0.000
eggs      0.000      0 0.0000      0 0.0000         0 0.000 0.128
wormCount 0.282      0 0.0000      0 0.4068         0 0.434 0.376
age       0.000      0 0.0189      0 0.0000         0 0.000 0.000
adg       0.000      0 0.0000      0 0.0000         0 0.390 0.000
```

4.8. Presentation of the results

The posterior marginals represent estimates of the parameters at each node, i.e., the arcs in the DAG. As the variables are coming from different distributions, they have different biological interpretations. In Figure 9, the square nodes are Bernoulli distributed, the oval nodes are normally distributed and the diamond node is Poisson distributed. The posterior marginals represent correlation coefficients for Gaussian nodes, log rate ratios for Poisson nodes, and log odds ratios for binomial nodes. Binomial and Poisson nodes need to be exponentiated to get the odds ratios or rate ratios, respectively.

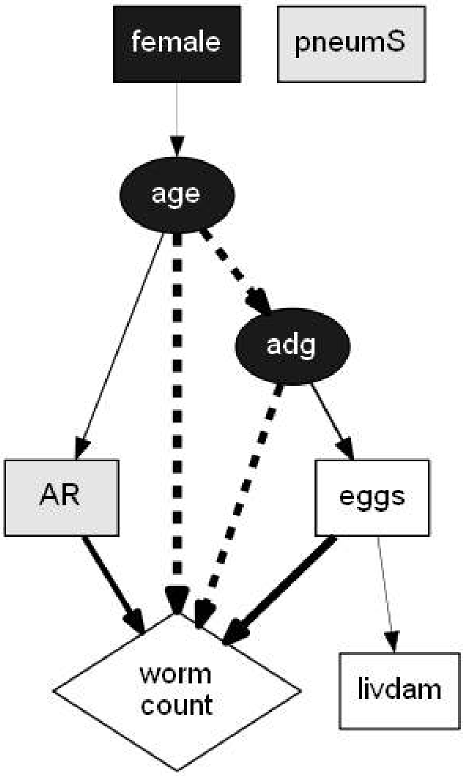

Figure 9: Final ABN model with arc width proportional to link strength

Table 2: Marginals posterior distribution of the parameter estimates

	2.5Q	median	97.5Q	interpretation	LS
AR|age	1.46	2.14	3.19	odds ratio	0.09
livdam|eggs	0.92	2.84	10.23	odds ratio	0.04
eggs|adg	0.71	1.10	1.73	odds ratio	0.13
wormCount|AR	1.38	1.57	1.79	rate ratio	0.28
wormCount|eggs	11.38	14.85	19.59	rate ratio	0.41
wormCount|age	0.77	0.91	1.08	rate ratio	0.43
wormCount|adg	0.78	0.90	1.05	rate ratio	0.38
age|female	0.31	0.44	0.58	correlation	0.02
adg|age	-0.90	-0.84	-0.77	correlation	0.39

5. Simulation study

Simulation studies are needed to test models and implemented methods. As an illustration we show here one simulation that illustrates the efficiency of the implemented scoring system. More simulation studies are in the annexes: A about DAGs structural metrics and B about a quality assesment of the estimation of the regression coefficients.

The parameters which are important from a simulation point of view are the BN dimension, i.e., the number of nodes of the BN, the structure density, i.e., average number of parents per node, and the sample size. Additionally to those structure-wise metrics, important factors that impact simulations are the intensity of the arc link, the variability of the arc's distributions and the mixture of variables. Indeed, scores used are only approximately the same over Markov equivalence classes and the discrepancy increases when mixing distributions.

5.1. Comparison of score accuracy

In order to estimate the efficiency of the different scores implemented in the R package **abn**, BNs with a given arc density have been simulated, from which 20 datasets have been simulated with different sample sizes. The metrics used to display performance of the score are the true positive (number of arcs properly retrieved), false positive (selecting an arc when there is not one) and false negative (selecting no arc when there is one). The scores used are marginal posterior likelihood in a Bayesian regression framework (abn), maximum likelihood without penalization for complexity (mlik), AIC, BIC and MDL.

Figure 10 shows for an increasing number of observation $n = 5, 10, 50, \ldots, 10000$ the accuracy of five scores mentioned above for a given BN density using boxplots to display the variability of the distribution of different metrics. As one can see in Figure 10, the mlik (maximum likelihood) is a sub-optimal score for BN learning as Daly *et al.* (2011) expected. None of the other scores are better or worse in general and they give different results for a specific dataset. Interestingly from an applied perspective, the marginal posterior likelihood seems to achieve very promising results with a limited number of observations.

6. Summary and discussion

The R package **abn** is a free implementation of the ABN methodology that allows users to fit Bayesian and MLE ABN models to observational datasets. The R package **abn** contains functions to analyze, select, plot and simulate ABN models. The functions are designed to be usable with limited knowledge of state-of-the-art Bayesian inference methods.

The R package **abn** suffers from several applied and modelling limitations. Some of these limitations are inherent in the Bayesian modeling framework some are tentatively tackled in an ABN satellite suite of R packages. These packages are at different development stages, but they are all designed to work synergetically with the R package **abn** and hopefully will be integrated in the long term.

From an applied perspective, researchers often have to extract a limited number of variables from a large observational dataset. In an ABN context, the exact search algorithm is computationally very sensitive to the number of variables in the network. An ABN analysis targets epidemiological modeling problems where a set of variables of importance rather than a unique outcome can be clearly identified for the modelling. Most of the classical model selection technique requires a model and a clear outcome to assess the predictive power of the covariate. BNs modeling is a typical setting where the variables should be selected to build the model or researchers are interested in a multi-outcome dataset. Driven by these findings, a variable selection approach that solves those issues has been proposed to facilitate the ABN analysis. The R package **varrank** (Kratzer and Furrer 2018a,b) is an implementation of the minimum redundancy maximum relevance model (Battiti 1994) that performs multi-outcome variable ranking. It can be used prior to the R package **abn** to perform dimensionality reduction and to focus on modelling important variables. A possible alternative could be to use

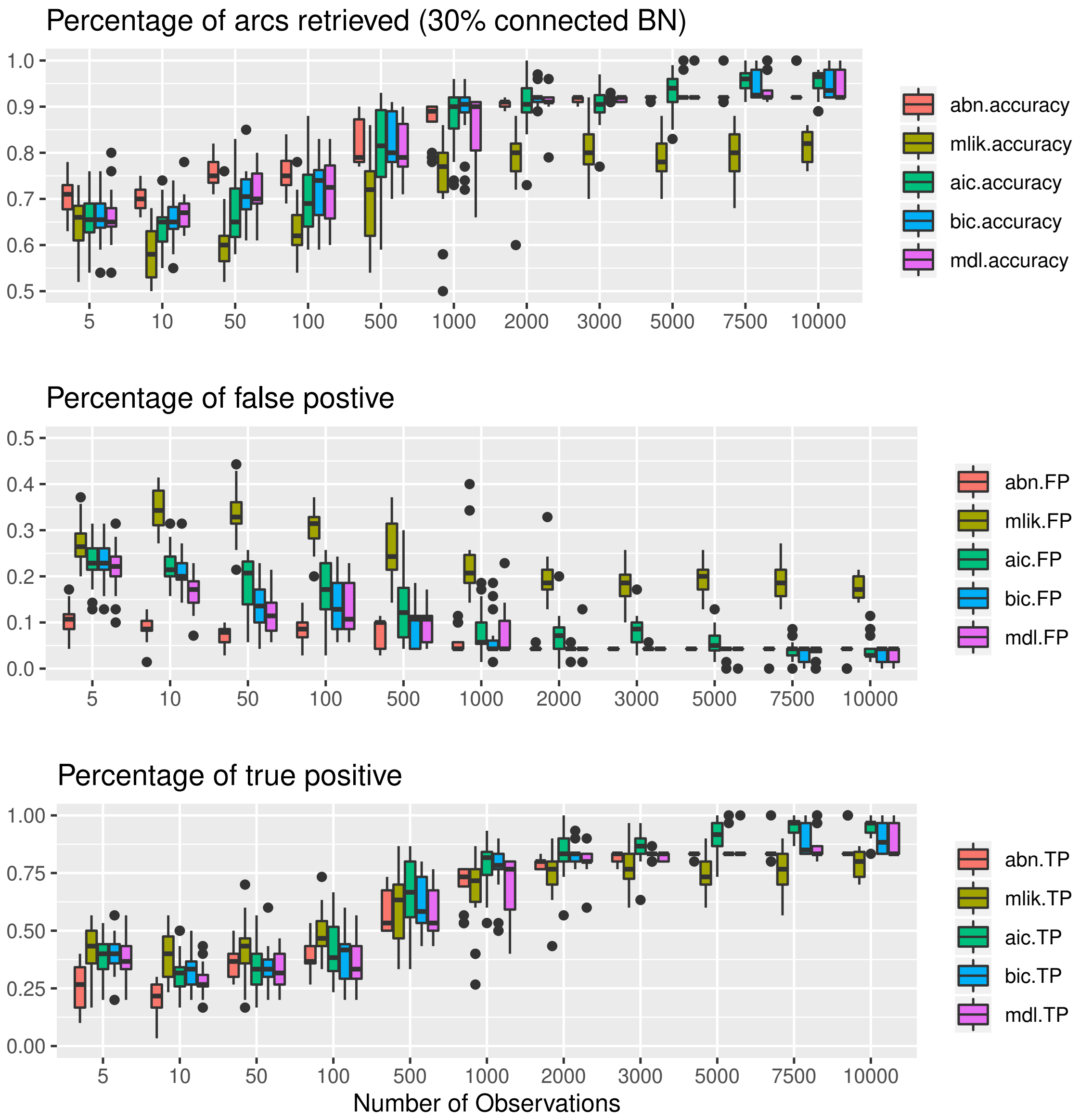

Figure 10: Comparison of score accuracy for marginal posterior likelihood (abn), maximum likelihood (mlik), AIC, BIC and MDL for a given BN density in function of the number of observations using accuracy, true positive, false positive and false negative metrics.

Random Forest's variable importance that also supports multi-outcome formulation (Strobl, Boulesteix, Zeileis, and Hothorn 2007).

A more theoretical limitation of the R package **abn** is the fact that the parameter priors implemented are designed to be non-informative. Bayesian model selection algorithms with uninformative priors will asymptotically always select the simpler model, regardless of the data. This is known as Lindley's paradox (Lindley 1957). A byproduct of more informative priors is the ability to guide the posterior when the likelihood is hard to estimate. This could be useful when there is data separation or in the case of paucity of data. Indeed, while probably suitable in large datasets, flat priors can quickly turn troublesome when the data is not informative for a parameter of interest (Kratzer, Furrer, and Pittavino 2019a). From a long-term perspective, it is certainly of interest to equip **abn** with weakly informative priors whatever the distribution to incorporate and extend the R package **abn**.

One general limitation of the BN modelling approach and the R package **abn** is the strong assumption of data independence coming from the regression framework used. Multiple approaches have been proposed to model temporal dynamics, such as Dynamic Bayesian Network modelling (DBNs) (Murphy and Russell 2002), Temporal Nodes Bayesian Networks (TNBNs) (Arroyo-Figueroa and Sucar 1999), VAR processes (Ahelegbey, Billio, and Casarin 2016) and state-space or hidden Markov models (Le Strat and Carrat 1999).

The two pillars of the statistical inference are the estimation of the effect size and the quantification of the model uncertainties. In a regression context, this is achieved by reporting regression coefficients and their confidence intervals (or sub-optimally using solely the coefficients' p-values). This is done in ABN analysis when reporting the quantiles of the posterior distribution of the coefficients. However, this is not fully satisfactory. Indeed, the uncertaint quantification is performed conditionally to one model. A superior approach would be to perform Bayesian model averaging, using MCMC over the possible structures. The computed MCMC sample can be used to extract the arc probability of presence or absence. In a broader perspective, it could be possible to compute the probability of any structural query over the MCMC sample. This approach is implemented in the R package **BiDAG** (Suter and Kuipers 2018) or in the R package **mcmcabn** (Kratzer and Furrer 2019; Kratzer, Lewis, Willi, Meli, Boretti, Hofmann-Lehmann, Torgerson, Furrer, and Hartnack 2020). The former is based on an innovative order search, whereas the latter is based on a fully structural formulation. However, known to be slower in convergence and less efficient with large networks than the R package **BiDAG**, the R package **mcmcabn** has a fully transparent formulation regarding the structural priors used. It is Which eases the interpretation of findings. Moreover, in BN modeling applied to systems epidemiology due to the limited number of observations that are usually possible from a financial, temporal or logistical perspective, integrating prior knowledge into the inference scheme seems highly desirable.

7. Future developments

The case study showed that more exponential distributions should be implemented in order to better grasp the diversity of variables of interest in systems epidemiology. Indeed, zero-inflated variables are poorly represented by the Poisson distribution. A negative binomial would probably be a better alternative. This present paper presents an analysis that takes advantage of bootstrapping to decrease model variance. Figure 10 shows that information theoretic scores perform differently depending on the context. Thus, there could be an opportunity to construct a boosted score dedicated to the ABN methodology. This idea should be addressed in a dedicated simulation study.

Author's contributions

G.K. wrote and conceived the manuscript with support from A.C. and R.F. G.K. is author of half of the functions implemented in the R package **abn**. G.K. performed the numerical simulations and contributed to the analysis. G.K wrote the package's documentation. F.I.L. is the original creator and author of half of the functions in the R package **abn**. F.I.L. provided critical feedback and helped to shape the research project. A.C. identified the case study dataset, performed the analysis and contributed to the interpretation of the findings. M.P. contributed to the R package **abn** documentation. R.F. is the PhD supervisor of G.K. and M.P. R.F. probed the R package **abn** and provided useful suggestions that lead to numerous improvements on the package. R.F. provided input on the statistical framework, model implementation and findings reporting. All authors revised the manuscript.

Computational details

The results in this paper were obtained using R 3.6.1 (2019-07-05) – "Action of the Toes" with the **abn** 2.2 package. All computations were carried out using RStudio 1.2.5001. R and all packages used are available from the Comprehensive R Archive Network (CRAN) at https://CRAN.R-project.org/. RStudio is available at https://www.rstudio.com/. The JAGS version is 4.3.0. The INLA version is 3.6.

A. DAGs simulation: more technical details

In order to simulate DAGs, the function `simulateDag()` generates triangular matrices with user tuneble arc density. The node ordering is sampled from the node definition.

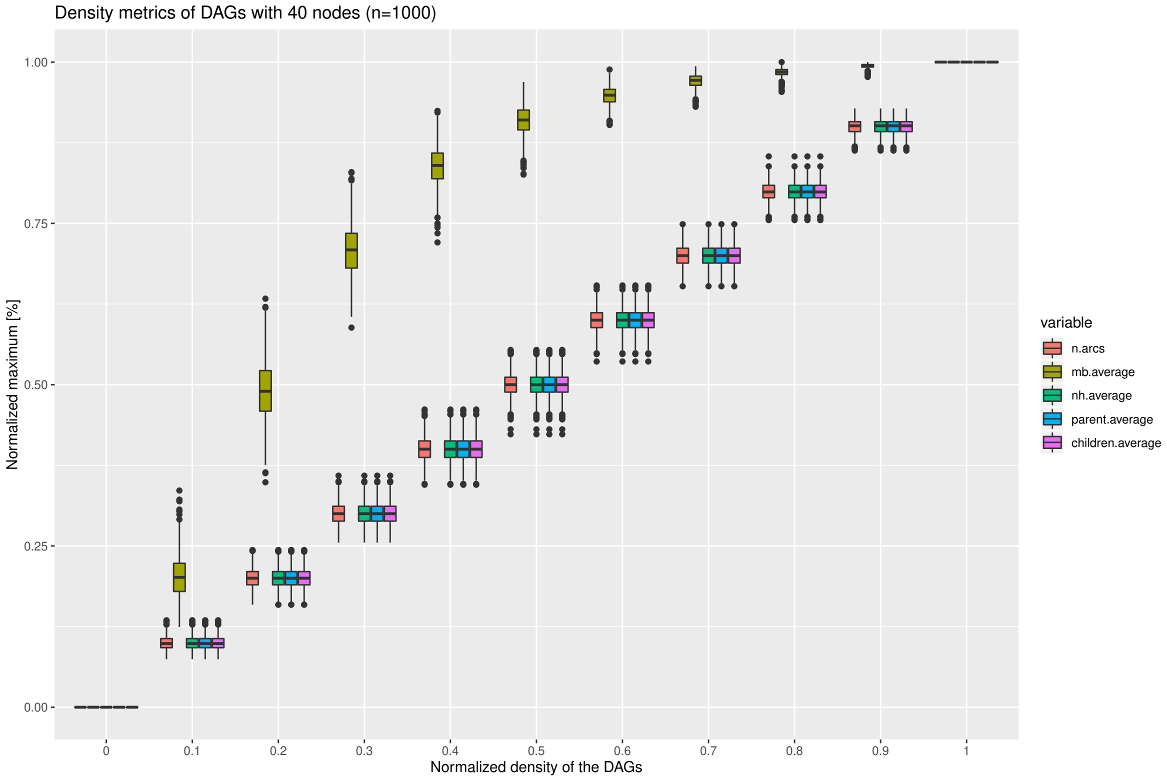

Figure 11: Normalized DAG structural metrics from simulated Bayesian networks of 40 nodes with different arc densities

In Figure 11, some normalized (i.e., divided by the maximum possible) DAGs structural metrics are displayed: n.arcs is the number of arcs, mb.average is the average size of the Markov blanket (the set of parents, children and co-parents), nh.average is the average number of neighbours, parent.average is the average number of parents and children.average is the average number of children in the BN. For each level of network complexity, 1000 BN with 40 nodes were simulated. The normalized distribution of the BN metrics computed with `infDag()` is reported. As one can see in Figure 11, the only normalized metric that exhibits non-linear behaviour with arc density is the Markov blanket.

B. Regression coefficients estimation: quality assurance check

The R package **abn** contains two functions to estimate the regression coefficent based on two statistical paradigms: the MLE formulation with an IRLS estimation, and a Bayesian formulation with diffuse priors estimated with INLA. Figure 12 shows a quality assurance check of the implementations.

The Bayesian and MLE implementations are compared in Figure 12 to estimate the accuracy of parameters. Two network densities, 20% and 80% of the possible arcs expressed, were simulated 50 times. Then the regression coefficients were computed for different sample sizes and the coefficient of variation of the given node as a proxy for the distribution's variability. As one can see in Figure 12, the error is measured as the maximum Root Mean Squared Error (max RMSE) on a log-log scale. Both implementations produce very similar results even if the estimation frameworks are very different.

C. Parametric bootstrapping: the R code

This section contains the R code needed to perform parametric bootstrapping described by the pseudo-code in Algorithm 1.

```
R> vars <- colnames(abndata)
R>
R> # load data for jags
R> source("PostParams.R")
R>
R> # select nr. bootstrap samples to run
R> set.seed(123)
R>
R> # get 5000 random numbers to set different initial values
R> n <- sample(1:100000, 5000)
R>
R> # specify max number of parents based on previous search
R> max.par <- 4
R>
R> # Simulate data and run ABN on such dataset
R> for (i in 1:length(n)) {
R+     print(paste("/n Running simulation", i))
R+     # pick initials
```

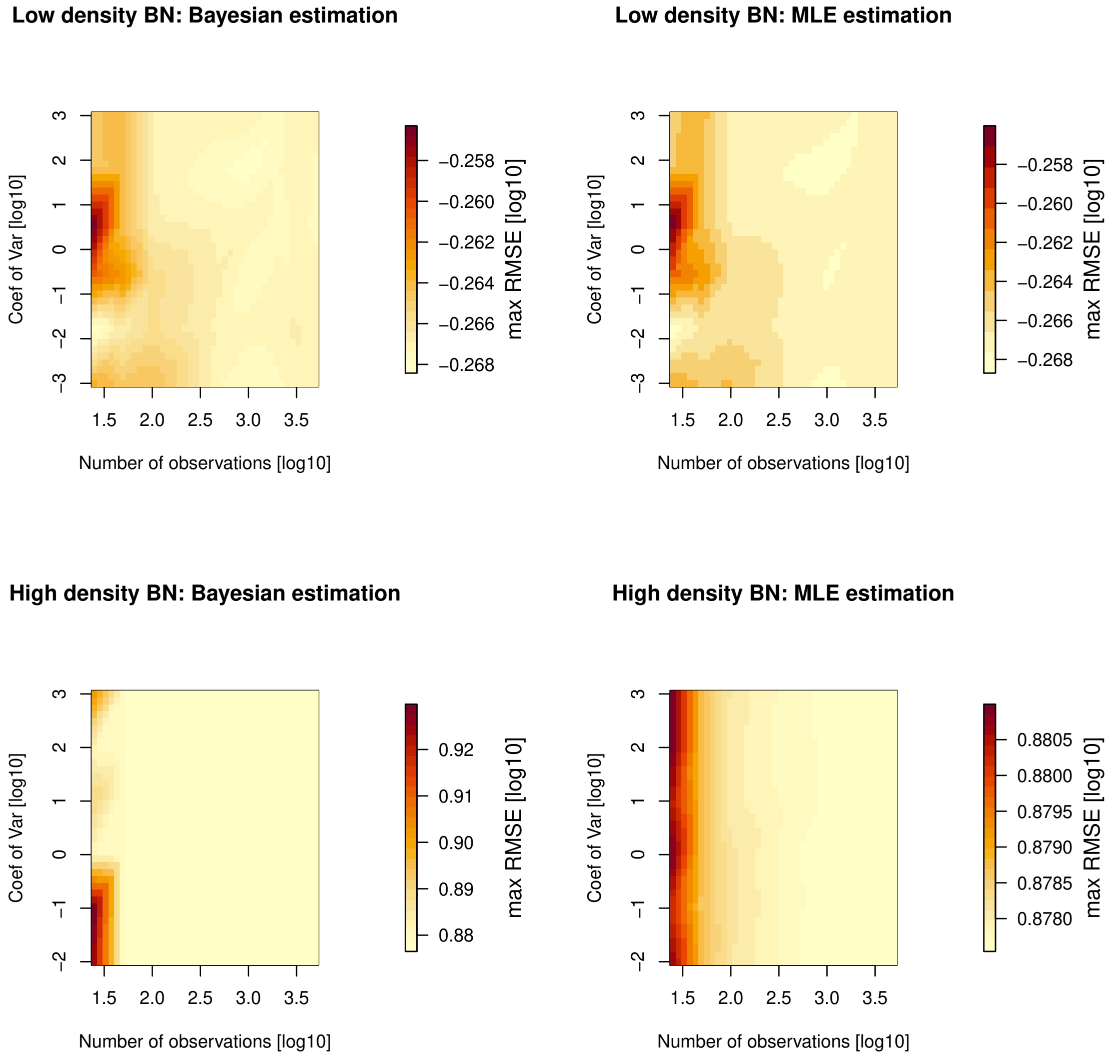

Figure 12: Comparison between Bayesian and MLE implementations to estimate regression coefficients in an ABN framework. The panels show the maximum Root Mean Squared Error (max RMSE) in function of the network density, the distribution variability and the sample size.

```
R+      init <- list(.RNG.name = "base::Mersenne-Twister", .RNG.seed = n[i])
R+      # run model
R+      jj <- jags.model(file = "plot/model8vPois.bug", data = list(AR.p = AR.p,
R+          pneumS.p = pneumS.p, female.p = female.p, livdam.p = livdam.p,
R+          eggs.p = eggs.p, wormCount.p = wormCount.p, age.p = age.p,
R+          prec.age.p = prec.age.p, adg.p = adg.p, prec.adg.p = prec.adg.p),
R+          inits = init, n.chains = 1, n.adapt = 5000)
R+      # run more iterations
```

```
R+     update(jj, 100000)
R+     # sample data (same size as original: 341) with a sampling
R+     # lag (20) to reduce autocorrelation
R+     samp <- coda.samples(jj, c("AR", "pneumS", "female", "livdam",
R+         "eggs", "wormCount", "age", "prec.age", "adg", "prec.adg"),
R+         n.iter = n.obs * 20, thin = 20)
R+     # build dataframe in the same shape as the original one
R+     dt.boot <- as.data.frame(as.matrix(samp))
R+     dt.boot <- dt.boot[, vars]
R+     # now coerce to factors if need be and set levels
R+     abndata <- as.data.frame(abndata)
R+     for (j in 1:length(vars)) {
R+         if (is.factor(abndata[, j])) {
R+             dt.boot[, j] <- as.factor(dt.boot[, j])
R+             levels(dt.boot[, j]) <- levels(abndata[, j])
R+         }
R+     }
R+     # Build a cache of all local computations
R+     mycache <- buildscorecache(data.df = dt.boot, data.dists = dist,
R+         dag.banned = banned, dag.retained = retain, max.parents = max.par)
R+     # Run the EXACT SEARCH
R+     mp.dag <- mostprobable(score.cache = mycache)
R+     fabn <- fitabn(object = mp.dag)
R+     # Save the results obtained
R+     save(mycache, mp.dag, fabn, dt.boot, file = sprintf("boot.%04d.RData",
R+         i))
R+ }
```

D. Numerical stability check: Parameter estimation

Once the trimmed DAG is obtained to keep robust structural features, we can extract the marginal posterior densities. Indeed, a BN has a qualitative part (the structure) and a quantitative part (parameters estimates). They are both equally important for interpreting and reporting the findings. As the estimation is based on the Laplace approximation and the R package **abn** does not rely on conjugate priors, a numerical check is performed. Figure 13 shows the marginal densities. The critical assumption is that the data contains sufficient information to accurately estimate the density of individual parameters of the model. This is a stronger requirement than simply being able to estimate an overall goodness of fit metric. It could happen that INLA or the internal C code cannot estimate the densities with enough numerical stability. Many user-tunable parameters can be supplie to `fitabn()` by providing a list to the `control` argument.

The shape of the posterior parameter distributions looks satisfactory (see Figure 13).

References

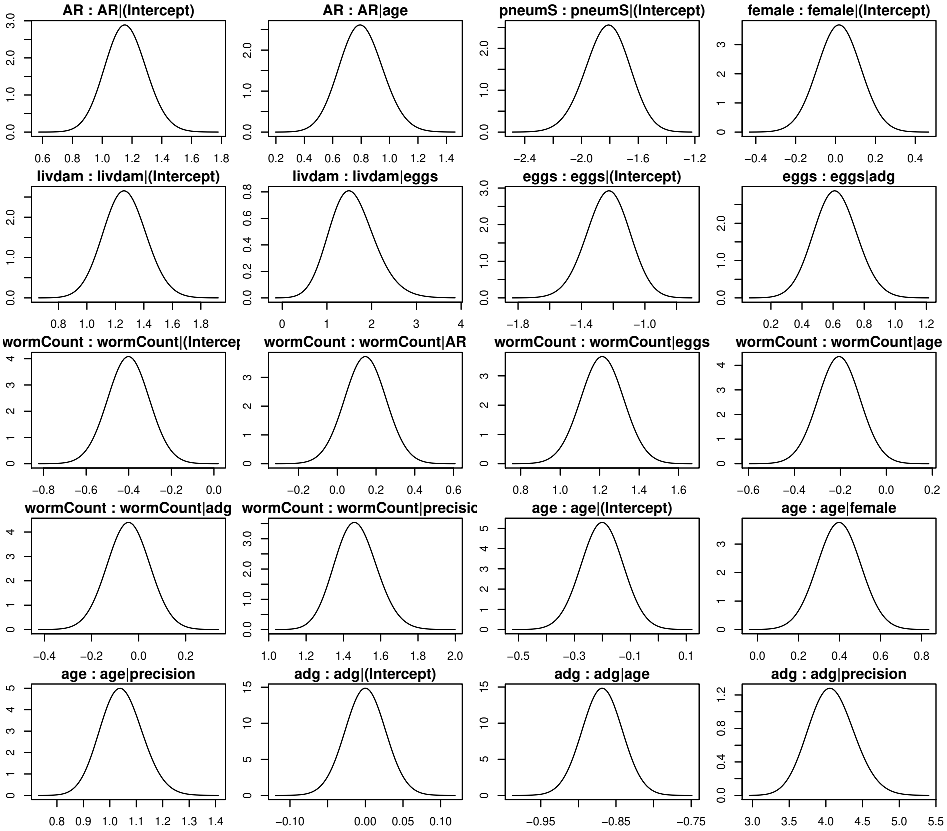

Figure 13: Marginal densities of model parameters for the adg data

Ahelegbey DF, Billio M, Casarin R (2016). “Bayesian Graphical Models for Structural Vector Autoregressive Processes.” *Journal of Applied Econometrics*, **31**(2), 357–386.

Arroyo-Figueroa G, Sucar LE (1999). “A Temporal Bayesian Network for Diagnosis and Prediction.” In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pp. 13–20. Morgan Kaufmann Publishers Inc.

Battiti R (1994). “Using Mutual Information for Selecting Features in Supervised Neural Net Learning.” *IEEE Transactions on Neural Networks*, **5**(4), 537–550.

Belsley DA, Kuh E, Welsch RE (2005). *Regression diagnostics: Identifying influential data and sources of collinearity*, volume 571. John Wiley & Sons.

Boerlage B (1992). *Link Strength in Bayesian Networks.* Ph.D. thesis, University of British Columbia.

Bøttcher SG, Dethlefsen C, *et al.* (2003). ***deal****: A Package for Learning Bayesian Networks.* Department of Mathematical Sciences, Aalborg University.

Bouckaert RR (2008). “Bayesian Network Classifiers in Weka for Version 3-5-7.” *Artificial Intelligence Tools*, **11**(3), 369–387.

Breslow NE, Clayton DG (1993). “Approximate Inference in Generalized Linear Mixed Models.” *Journal of the American Statistical Association*, **88**(421), 9–25.

Comin A, Jeremiasson A, Kratzer G, Keeling L (2019). “Revealing the Structure of the Associations Between Housing System, Facilities, Management and Welfare of Commercial Laying Hens Using Additive Bayesian Networks.” *Preventive Veterinary Medicine*, **164**, 23–32.

Consonni G, Rocca LL (2012). “Objective Bayes Factors for Gaussian Directed Acyclic Graphical Models.” *Scandinavian Journal of Statistics*, **39**(4), 743–756.

Cornet D, Sierra J, Tournebize R, Gabrielle B, Lewis FI (2016). “Bayesian Network Modeling of Early Growth Stages Explains Yam Interplant Yield Variability and Allows for Agronomic Improvements in West Africa.” *European Journal of Agronomy*, **75**, 80–88.

Daly R, Shen Q, Aitken S (2011). “Learning Bayesian Networks: Approaches and Issues.” *The Knowledge Engineering Review*, **26**(2), 99–157.

de Jongh M, Druzdzel MJ (2009). “A comparison of structural distance measures for causal Bayesian network models.” *Recent Advances in Intelligent Information Systems, Challenging Problems of Science, Computer Science series*, pp. 443–456.

Dethlefsen C, Højsgaard S, *et al.* (2005). “A Common Platform for Graphical Models in R: The **gRbase** Package.” *Journal of Statistical Software*, **14**(17), 1–12.

Dohoo IR, Martin W, Stryhn H, *et al.* (2003). *Veterinary Epidemiologic Research*. V413 DOHv. AVC Incorporated Charlottetown, Canada.

Ebert-Uphoff I (2009). “Tutorial on How to Measure Link Strengths in Discrete Bayesian Networks.” URL http://hdl.handle.net/1853/29804.

Faraway JJ (2016). *Extending the Linear Model with R: Generalized Linear, Mixed Effects and Nonparametric Regression Models*. CRC Press.

Firestone SM, Lewis FI, Schemann K, Ward MP, Toribio JAL, Taylor MR, Dhand NK (2014). “Applying Bayesian Network Modelling to Understand the Links Between On-Farm Biosecurity Practice During the 2007 Equine Influenza Outbreak and Horse Managers’ Perceptions of a Subsequent Outbreak.” *Preventive Veterinary Medicine*, **116**(3), 243–251.

Friedman N, Goldszmidt M, Wyner A (1999). “Data Analysis with Bayesian Networks: A Bootstrap Approach.” In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pp. 196–205. Morgan Kaufmann Publishers Inc.

Hair JF, Black WC, Babin BJ, Anderson RE, Tatham RL, *et al.* (1998). *Multivariate Data Analysis*, volume 5. Prentice Hall Upper Saddle River, NJ.

Hansen KD, Gentry J, Long L, Gentleman R, Falcon S, Hahne F, Sarkar D (2019). ***Rgraphviz**: Provides plotting capabilities for R graph objects*. ProglangR package version 2.28.0, URL https://bioconductor.org/packages/Rgraphviz/.

Hartnack S, Odoch T, Kratzer G, Furrer R, Wasteson Y, L’Abée-Lund TM, Skjerve E (2019). “Additive Bayesian Networks for Antimicrobial Resistance and Potential Risk Factors in Non-Typhoidal Salmonella Isolates from Layer Hens in Uganda.” *BMC Veterinary Research*, **15**(1), 212.

Hartnack S, Springer S, Pittavino M, Grimm H (2016). "Attitudes of Austrian Veterinarians Towards Euthanasia in Small Animal Practice: Impacts of Age and Gender on Views on Euthanasia." *BMC Veterinary Research*, **12**(1), 26.

Hasan A, Wang Z, Mahani A (2016). "Fast Estimation of Multinomial Logit Models: R Package **mnlogit**." *Journal of Statistical Software*, **75**(3), 1–24. ISSN 1548-7660. doi:10.18637/jss.v075.i03. URL https://www.jstatsoft.org/v075/i03.

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

Højsgaard S, *et al.* (2012). "Graphical Independence Networks with the **gRain** Package for R." *Journal of Statistical Software*, **46**(10), 1–26.

Hornik K, Buchta C, Zeileis A (2009). "Open-Source Machine Learning: R Meets Weka." *Computational Statistics*, **24**(2), 225–232.

Iannone R (2019). ***DiagrammeR**: Graph/Network Visualization.* R Package version 1.0.1, URL https://CRAN.R-project.org/package=DiagrammeR.

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

Koivisto M, Sood K (2004). "Exact Bayesian Structure Discovery in Bayesian Networks." *Journal of Machine Learning Research*, **5**(May), 549–573.

Kratzer G, Furrer R (2018a). ***varrank**: an R package for variable ranking based on mutual information with applications to observed systemic datasets.* R package version 0.2, URL https://CRAN.R-project.org/package=varrank.

Kratzer G, Furrer R (2018b). "**varrank**: An R Package for Variable Ranking Based on Mutual Information with Applications to Observed Systemic Datasets." *arXiv preprint arXiv:1804.07134.*

Kratzer G, Furrer R (2019). "Is a Single Unique Bayesian Network Enough to Accurately Represent Your Data?" *arXiv preprint arXiv:1902.06641.*

Kratzer G, Furrer R, Pittavino M (2019a). "Comparison between Suitable Priors for Additive Bayesian Networks." In R Argiento, D Durante, S Wade (eds.), *Bayesian Statistics and New Generation*. Springer-Verlag.

Kratzer G, Lewis FI, Willi B, Meli ML, Boretti FS, Hofmann-Lehmann R, Torgerson P, Furrer R, Hartnack S (2020). "Bayesian Network Modeling Applied to Feline Calicivirus Infection Among Cats in Switzerland." *Frontiers in Veterinary Science*, **7**, 73.

Kratzer G, Pittavino M, Lewis F, Furrer R (2019b). ***Abn**: An R Package for Modelling Multivariate Data Using Additive Bayesian Networks.* R package version 2.2, URL https://CRAN.R-project.org/package=abn.

Lauritzen S, Spiegelhalter DJ (1988). "Local Computations with Probabilities on Graphical Structures and Their Application to Expert Systems (With Discussion)." *Journal of the Royal Statistical Society, Series B*, **50**, 157–224.

Le Strat Y, Carrat F (1999). “Monitoring Epidemiologic Surveillance Data Using Hidden Markov Models.” *Statistics in Medicine*, **18**(24), 3463–3478.

Lewis FI, Ward MP (2013). “Improving Epidemiologic Data Analyses Through Multivariate Regression Modelling.” *Emerging Themes in Epidemiology*, **10**(1), 4.

Lindley DV (1957). “A Statistical Paradox.” *Biometrika*, **44**(1/2), 187–192.

Ludwig A, Berthiaume P, Boerlin P, Gow S, Léger D, Lewis FI (2013). “Identifying Associations in Escherichia Coli Antimicrobial Resistance Patterns Using Additive Bayesian Networks.” *Preventive Veterinary Medicine*, **110**(1), 64–75.

MacKay DJ (1992). “Bayesian Interpolation.” *Neural Computation*, **4**(3), 415–447.

McCormick B (2014). “Frequent Symptomatic or Asymptomatic Infections May Have Long-Term Consequences on Growth and Cognitive Development.” In *Old Herborn University Seminar Monographs*, pp. 23–39. Institute for Microbiology und Biochemistry Herborn, Germany.

McCormick B, Sanchez-Vazquez M, Lewis F (2013). “Using Bayesian Networks to Explore the Role of Weather as a Potential Determinant of Disease in Pigs.” *Preventive Veterinary Medicine*, **110**(1), 54–63.

Murphy K, *et al.* (2001). “The Bayes Net Toolbox for MATLAB.” *Computing Science and Statistics*, **33**(2), 1024–1034.

Murphy KP, Russell S (2002). “Dynamic Bayesian Networks: Representation, Inference and Learning.”

Pearl J (1985). *Bayesian Networks: A Model for Self-Activated Memory for Evidential Reasoning.* University of California, Los Angeles. Computer Science Department.

Pitman EJG (1936). “Sufficient Statistics and Intrinsic Accuracy.” *Mathematical Proceedings of the Cambridge Philosophical Society*, **32**(4), 567–579.

Pittavino M, Dreyfus A, Heuer C, Benschop J, Wilson P, Collins-Emerson J, Torgerson P, Furrer R (2017). “Comparison Between Generalized Linear Modelling and Additive Bayesian Network; Identification of Factors Associated with the Incidence of Antibodies Against Leptospira Interrogans SV Pomona in Meat Workers in New Zealand.” *Acta Tropica*, **173**, 191–199.

Plummer M (2018). ***rjags**: Bayesian Graphical Models using MCMC.* R Package version 4.9, URL https://CRAN.R-project.org/package=rjags.

Plummer M, *et al.* (2003). “Jags: A Program for Analysis of Bayesian Graphical Models Using Gibbs Sampling.” In *Proceedings of the 3rd International Workshop on Distributed Statistical Computing*, volume 124. Vienna, Austria.

R Core Team (2017). *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Robinson RW (1973). “Counting Labeled Acyclic Digraphs, New Directions in the Theory of Graphs (Proc. Third Ann Arbor Conf., Univ. Michigan, Ann Arbor, Mich., 1971).”

Robinson RW (1977). "Counting Unlabeled Acyclic Digraphs." In *Combinatorial Mathematics V*, pp. 28–43. Springer-Verlag.

Ruchti S, Kratzer G, Furrer R, Hartnack S, Würbel H, Gebhardt-Henrich SG (2019). "Progression and Risk Factors of Pododermatitis in Part-Time Group Housed Rabbit Does in Switzerland." *Preventive Veterinary Medicine*, **166**, 56–64.

Ruchti S, Meier AR, Würbel H, Kratzer G, Gebhardt-Henrich SG, Hartnack S (2018). "Pododermatitis in Group Housed Rabbit Does in Switzerland – Prevalence, Severity and Risk Factors." *Preventive Veterinary Medicine*, **158**, 114–121.

Rue H, Martino S, Chopin N (2009). "Approximate Bayesian Inference for Latent Gaussian Models by Using Integrated Nested Laplace Approximations." *Journal of the Royal Statistical Society: Series B*, **71**(2), 319–392.

Scutari M (2010). "Learning Bayesian Networks with the **bnlearn** R Package." *Journal of Statistical Software*, **35**(3), 1–22.

Scutari M (2018). "Dirichlet Bayesian Network Scores and the Maximum Relative Entropy Principle." *Behaviormetrika*, **45**(2), 337–362.

Spirtes P (2001). "An Anytime Algorithm for Causal Inference." In *AISTATS*. MIT Press.

Stehman SV (1997). "Selecting and interpreting measures of thematic classification accuracy." *Remote sensing of Environment*, **62**(1), 77–89.

Strobl C, Boulesteix AL, Zeileis A, Hothorn T (2007). "Bias in Random Forest Variable Importance Measures: Illustrations, Sources and a Solution." *BMC Bioinformatics*, **8**(1), 25.

Suter P, Kuipers J (2018). ***BiDAG**: Bayesian Inference for Directed Acyclic Graphs (BiDAG).* R package version 1.1.2, URL https://CRAN.R-project.org/package=BiDAG.

van Smeden M, de Groot JA, Moons KG, Collins GS, Altman DG, Eijkemans MJ, Reitsma JB (2016). "No Rationale for 1 Variable per 10 Events Criterion for Binary Logistic Regression Analysis." *BMC Medical Research Methodology*, **16**(1), 163.

Venables W, Ripley B (2002). "Modern Applied Statistics with S. Springer, New York."

Verma T, Pearl J (1988). *Influence Diagrams and D-Separation.* UCLA, Computer Science Department.

Zou Y, Roos T (2017). "On Model Selection, Bayesian Networks, and the Fisher Information Integral." *New Generation Computing*, **35**(1), 5–27.

Affiliation:

Gilles Kratzer
Zurich University
Department of Mathematics, Winterthurerstrasse 190, CH-8057 Zürich, Switzerland

E-mail: gilles.kratzer@math.uzh.ch
URL: http://r-bayesian-networks.org/