mcmcabn: A Structural Mcmc Sampler for Dags Learned from Observed Systemic Datasets

Gilles Kratzer, Reinhard Furrer

2020-03-01

What is the package designed for?

The three main problems addressed by this R package are:

1. Selecting the most probable structure based on a cache of pre-computed scores.
2. Controlling for overfitting.
3. Sampling the landscape of high scoring structures.

The latter could be beneficial in an applied perspective to avoid reducing the richness of Bayesian network modeling to only **one** structure. Indeed, it allows users to quantify the marginal impact of relationships (arcs in a network) of interest by marginalizing out over structures or nuisance dependencies (i.e., all other possible relationships). Structural Monte Carlo Markov Chain (MCMC) seems an elegant and natural way to estimate the *true* marginal impact, so one can determine if it's magnitude is big enough to consider as a worthwhile intervention.

Note: in this vignette *structural* MCMC means that the MCMC moves are performed at structure level (i.e., Bayesian network or Directed Acyclic Graph (DAG)) and not at node ordering level for example. The advantage is that explicit priors can be formulated. A DAG is a graph used to encode the joint probability of a set of variables. It is made of nodes (random variables) and arrows or arcs (directed links representing the relationships among variables).

How does it work?

The `mcmcabn` R package takes a cache of pre-computed network scores as input (possibly computed using `buildScoreCache()` function from `abn` R package). Then, the `mcmcabn()` function generates MCMC samples from the posterior distribution of the most supported DAGs. To do so the user should choose a structural prior and a learning scheme. `mcmcabn` is equipped with multiple structural priors: the so-called Koivisto prior (an uninformative prior), the Ellis prior (a modular flat prior) and a possibly user define prior and three algorithms for inferring the structure of Bayesian networks: the classical Monte Carlo Markov Chain Model Choice ($(MC)^3$), the new edge reversal (REV) from Grzegorczyk and Husmeier (2008) and the Markov blanket resampling (MBR) from Su and Borsuk (2016).

What are the R package functionalities?

The workhorse of the R package is the `mcmcabn()` function. It generates MCMC samples from the posterior distribution of the DAGs. This object of class `mcmcabn` can be summarized with a comprehensive verbal description or a plot. From those samples, one can either compute the most probable structure and plot the cumulative maximum score achieved or analyze with the `query()` R function that recognizes the formula statement. For example, what is the probability of having an arc between two nodes, one node being in the Markov blanket of another, or what is the probability to **not** have an arc between two nodes? Alternatively, one can query the frequency of an arc in one direction relative to the opposite direction.

Alternative R packages

BiDAG is an MCMC sampler that combined partition MCMC with constrained based methods. This R package is efficient to sample large DAGs.

What is the structure of this document?

We first illustrate the package with a simple example. Then some more technical details are provided.

Simple *mcmcabn* example

The package is available on CRAN and can be installed directly in R. However it needs two R packages: abn and gRbase that requires libraries not stored on CRAN but on bioconductor. Hence those two packages **must** be installed **before** installing `mcmcabn`:

```
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
BiocManager::install(c("RBGL", "Rgraphviz", "graph"),  version = "3.8")

install.packages("mcmcabn")
```

Once installed, the `mcmcabn` R package can be loaded using:

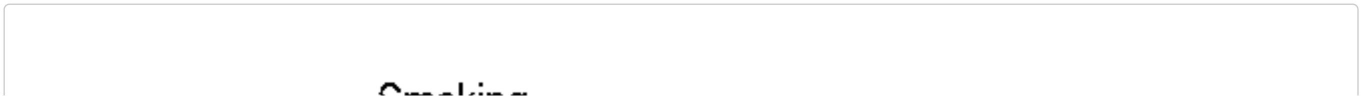
```
library(mcmcabn)
```

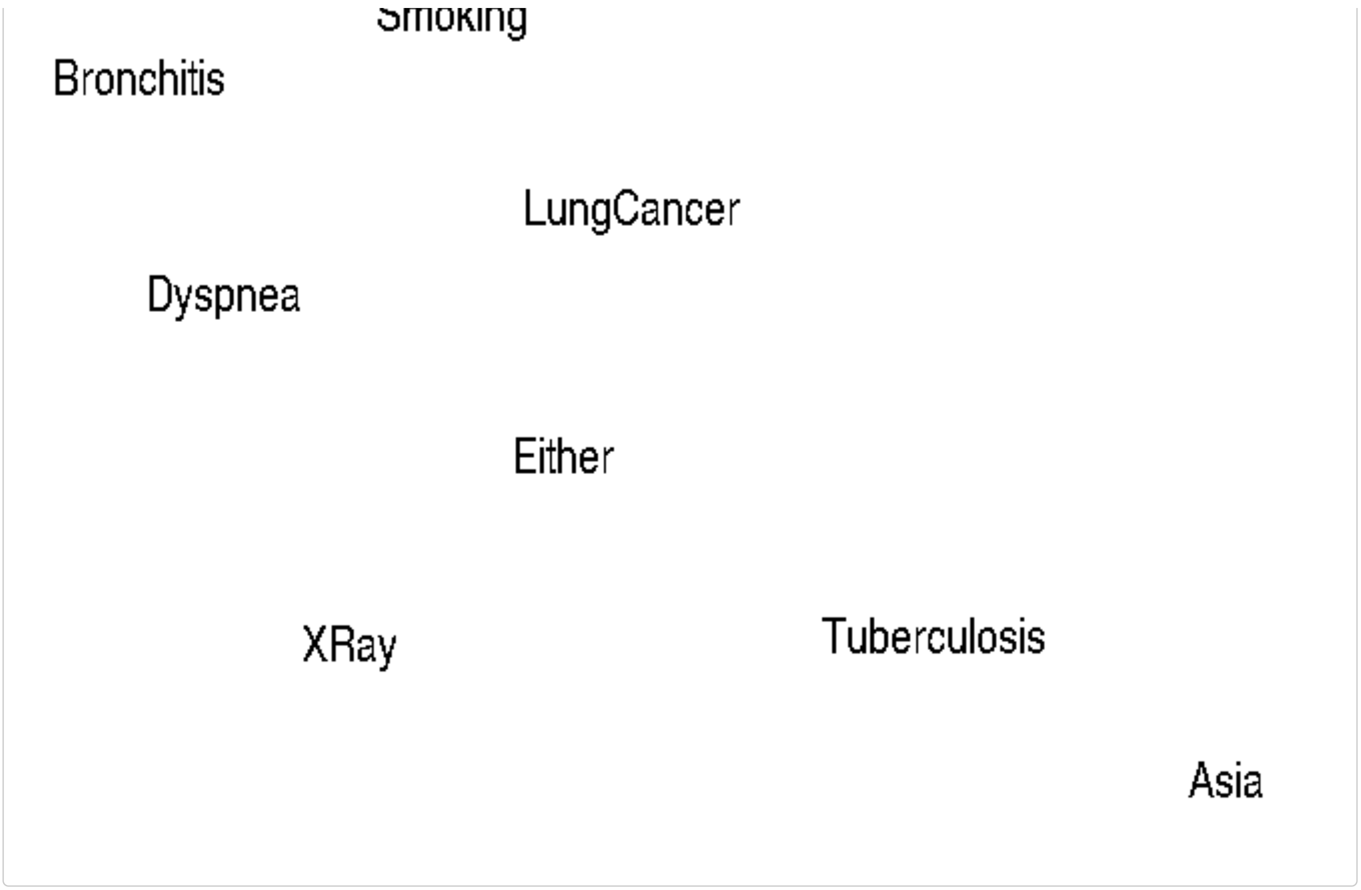
Let us start with an example from the `bnlearn` R package from Scutari (2010). It is about a small synthetic dataset from Lauritzen and Spiegelhalter (1988) about lung diseases (tuberculosis, lung cancer, or bronchitis) and visits to Asia (8 nodes and 8 arcs). The dataset is include in as `asia`.

```
library(abn) # to pre-compute the scores
library(ggplot2) # plotting
library(ggpubr) # plotting
library(cowplot) # plotting
library(ggdag) # to plot the DAG

#plot the BN as described in the paper
dag  <- dagify(Tuberculosis~Asia,
               Either~Tuberculosis,
               XRay~Either,
               Dyspnea~Either,
               Bronchitis~Smoking,
               LungCancer~Smoking,
               Either~LungCancer,
               Dyspnea~Bronchitis)

ggdag_classic(dag, size = 6) + theme_dag_blank()
```





One needs to pre-compute a cache of scores. We use the R package `abn` to do it. For the first search, it is advised to limit the number of parents per node to 2, which is the maximum number of parents of the network.

```
#loglikelihood scores
abnCache.2par.asia <- buildScoreCache(data.df = asia,
                                      data.dists = dist.asia,
                                      max.parents = 2)
```

We use the `mcmcabn()` function on the cache of pre-computed networks scores. One needs to define the type of score used (here is the marginal likelihood `mlik` other choice are: `bic`, `aic` or `mdl`). The maximum number of parents per node (same as the one used in `buildScoreCache()`). The MCMC learning scheme: number of MCMC samples, number of thinned sampled (to avoid autocorrelation), and the length of the burn-in phase. Here we choose: 1000 returned MCMC steps and 99 thinned steps in between. It means that for each returned MCMC move, 99 have been thinned. The burn-in phase is set to 10000 steps. We decide to initiate the algorithm with a random DAG `start.dag = "random"`, which means that the function randomly selects a valid DAG. In this context, valid means no cycle and an appropriate maximum number of parents. The frequencies of selecting a REV or the MBR jumps are set to 3%. It is usually a good idea to keep those frequencies low. Indeed, REV and MBR are efficient in producing high scoring structure but computationally costly compared to classical MCMC jumps. We select the Koivisto prior `prior.choice = 2`.

```
mcmc.2par.asia <- mcmcabn(score.cache = abnCache.2par.asia,
                          score = "mlik",
                          data.dists = dist.asia,
                          max.parents = 2,
                          mcmc.scheme = c(1000,99,1000),
                          seed = 42,
                          verbose = FALSE,
                          start.dag = "random",
                          prob.rev = 0.03,
                          prob.mbr = 0.03,
                          prior.choice = 2)
```

The function `mcmcabn()` returns a list with multiple entries:

- *dags* is the list of sampled DAGs.
- *scores* is the list of DAGs scores.
- *alpha* is the acceptance probability of the Metropolis-Hasting sampler.
- *method* is the algorithm chosen for the index MCMC step.
- *rejection* if an MCMC jumps is rejected or not.
- *iterations* is the total number of MCMC iterations.
- *thinning* is the number of thinned steps for one returned MCMC jump
- *burnin* is the length of the burn-in phase.
- *data.dist* is the list giving the distribution for each node in the network.

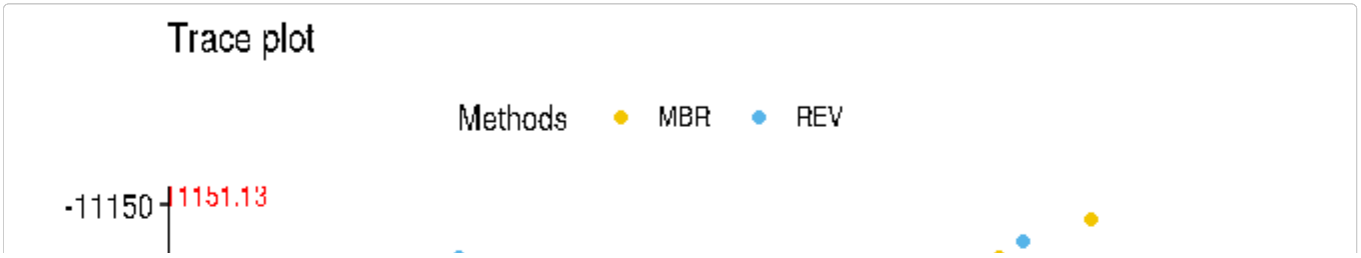
The object `mcmc.out` is of class `mcmcabn`. We can check that we reach the maximum possible posterior DAG score even with a small number of MCMC steps. To do so we use an exact search from Koivisto 2004 implemented in `mostProbable()` from `abn`.

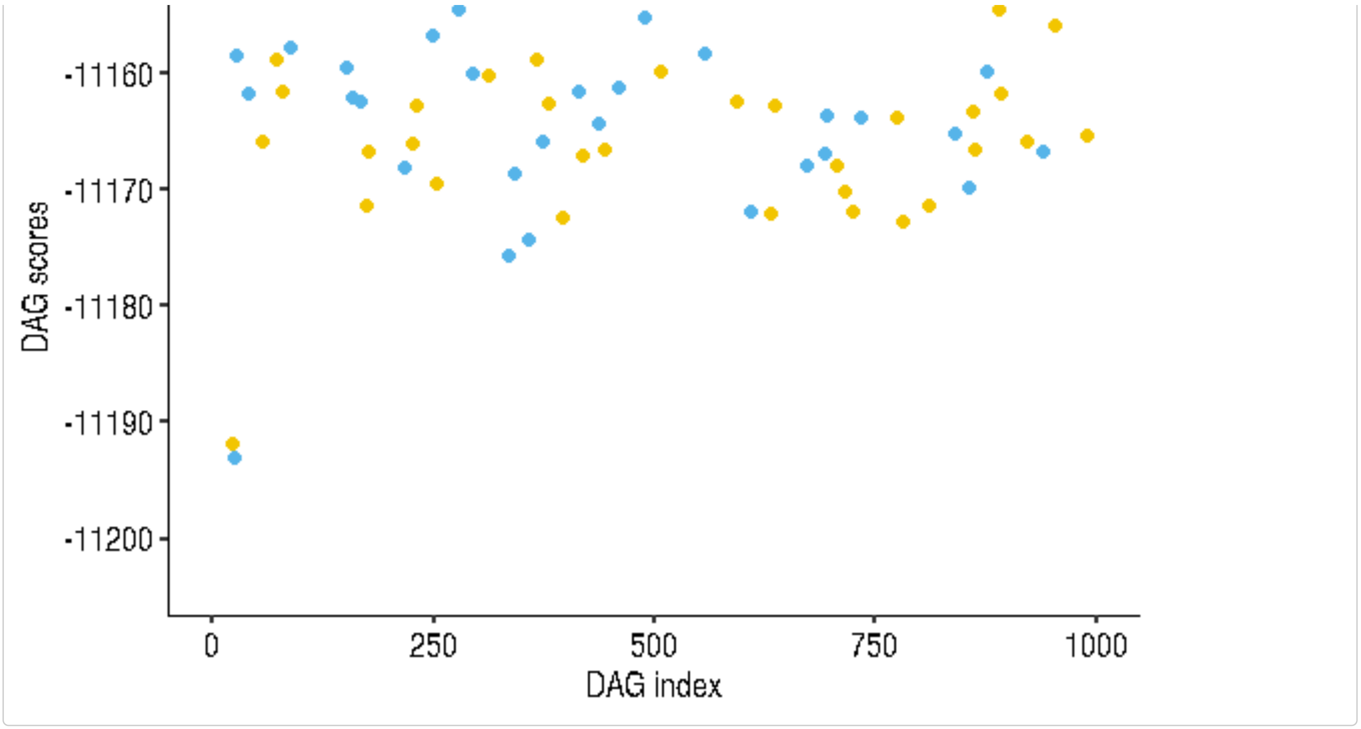
```
#maximum score get from the MCMC sampler
max(mcmc.2par.asia$scores)
#> [1] -11151

#maximum scoring network using exact search (not MCMC based)
dag <- mostProbable(score.cache = abnCache.2par.asia)
#> Step1. completed max alpha_i(S) for all i and S
#> Total sets g(S) to be evaluated over: 256
fitAbn(object = dag,data.df = asia, data.dists = dist.asia)$mlik
#> [1] -11151
```

One can plot the output using the *plot()*. On the graph below, one can see the trace plot of the posterior structure score. The dashed red line is the maximum reached score (as expected = -11151). The colored dots on the trace plot indicate when different methods have been used. The densities on the left represent the relative occurring frequencies of the methods.

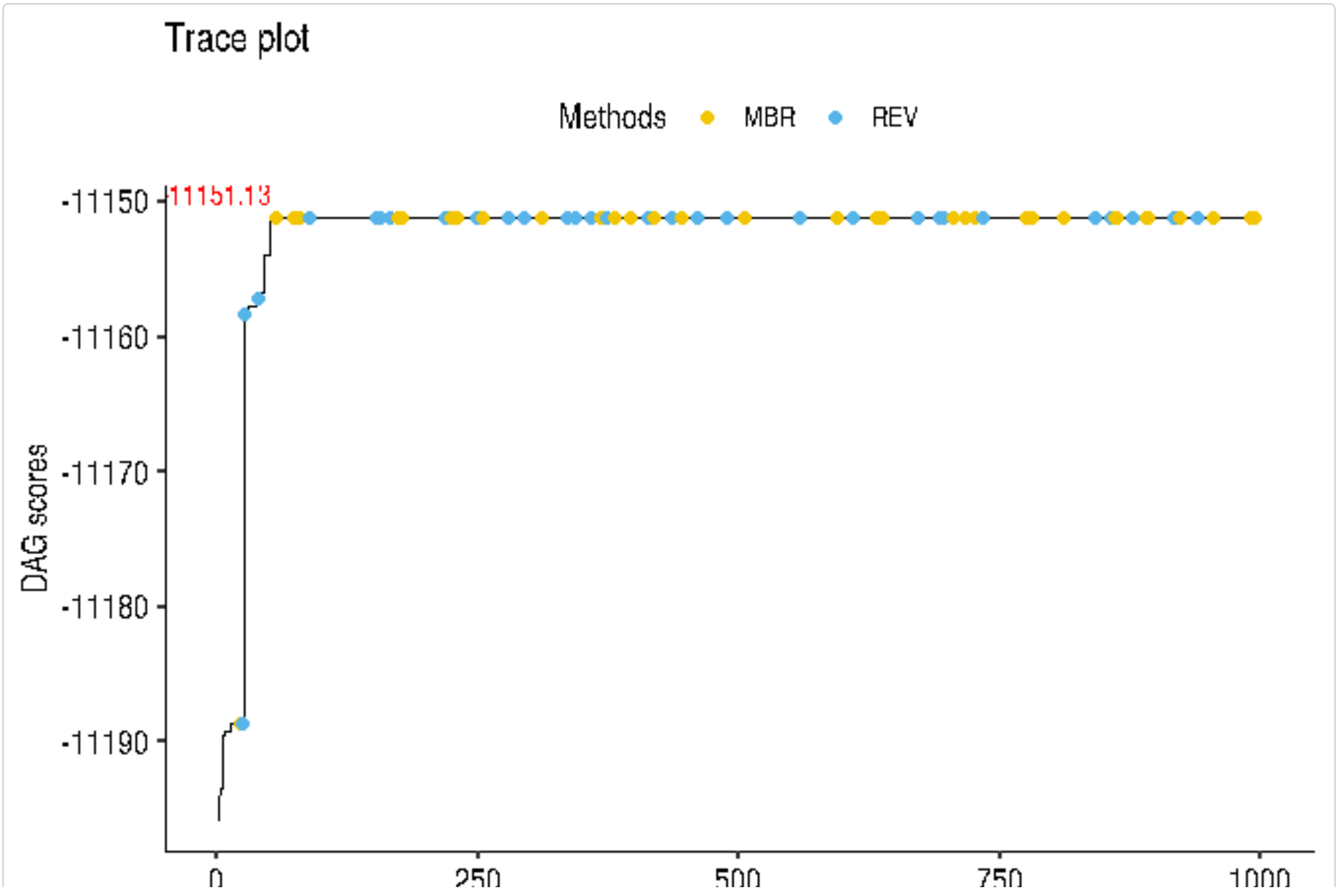
```
plot(mcmc.2par.asia)
```

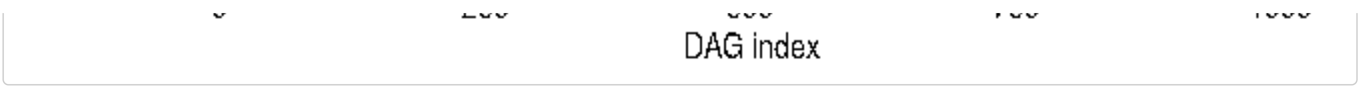




One can also plot the cumulative best score over the MCMC steps. On this graph, one can see the cumulated maximum posterior structure's score (as expected = -11151). The dashed red line is the maximum reached score. The colored dots on the trace plot indicate when different methods have been used.

```
plot(mcmc.2par.asia, max.score = TRUE)
```





One can also print the summary of the MCMC run:

```
summary(mcmc.2par.asia)
#> MCMC summary:
#> Number of burn-in steps: 1000
#> Number of MCMC steps: 1e+05
#> Thinning: 99
#>
#> Maximum score: -11151
#> Empirical mean: -11165
#> Empirical standard deviation: 7.41
#> Quantiles of the posterior network score:
#>           0.025   0.25    0.5   0.75  0.975
#> BN score -11189 -11168 -11164 -11160 -11154
#>
#>
#> Global acceptance rate: 0.205
#>     Accepted Rejected
#> MBR        9       24
#> MC3      186      752
#> REV       10       20
#>
#>
#> Sample size adjusted for autocorrelation: 51.6
#>
#> Autocorrelations by lag:
#>     0     1     2     3     4     5     6     7     8    9    10
#> acf 1 0.568 0.562 0.523 0.478 0.459 0.457 0.419 0.402 0.37 0.335
```

This function return some MCMC diagnostics and descriptive metrics.

The main advantage of computing MCMC samples from the posterior is that we can structurally query them. Hence we account for the uncertainty in the structures through Bayesian model averaging. In practice, we compute the probabilities of the arcs in the DAG or the probability of a given part of a structure. Below are the individual frequencies of the arcs.

```
query(mcmcabn = mcmc.2par.asia)
#>                Asia Smoking Tuberculosis LungCancer Bronchitis Either   XRay
#> Asia         0.0000  0.1698       0.1908      0.158      0.175 0.1229 0.1399
#> Smoking      0.0829  0.0000       0.0799      0.440      0.424 0.0360 0.0230
#> Tuberculosis 0.1319  0.1319       0.0000      0.277      0.123 0.1818 0.0390
#> LungCancer   0.0649  0.5125       0.2198      0.000      0.146 0.1499 0.0440
#> Bronchitis   0.0809  0.5764       0.1389      0.149      0.000 0.0539 0.0360
#> Either       0.0170  0.0529       0.8182      0.850      0.022 0.0000 0.0889
#> XRay         0.0649  0.0460       0.3047      0.271      0.032 0.7922 0.0000
#> Dyspnea      0.0050  0.0000       0.0270      0.172      0.968 0.7552 0.0270
#>              Dyspnea
#> Asia         0.13786
#> Smoking      0.00000
#> Tuberculosis 0.18681
#> LungCancer   0.03996
#> Bronchitis   0.03197
#> Either       0.00899
#> XRay         0.03596
#> Dyspnea      0.00000
```

The `query()` function takes formula statements. So it is possible to explicitly query: **what is the probability of LungCancer node being children of the Smoking node?**

```
query(mcmcabn = mcmc.2par.asia, formula = ~ LungCancer|Smoking)
#> [1] 0.512
```

This means that an arrow from Smoking to LungCancer appears in 51.249% of the sampled DAGs.

One can also ask more complicated structural requests. If we want to know **what is the probability of Smoking node being parent of both LungCancer and Bronchitis node?**

```
query(mcmcabn = mcmc.2par.asia, formula = ~ LungCancer|Smoking + Bronchitis|Smoking)
#> [1] 0.271
```

It means that an arrow from Smoking to LungCancer **and** an arrow from Smoking to Bronchitis appears in 27.073% of the sampled DAGs. This probability cannot be read from the matrix above. Indeed, it is a structural statement.

If one wants to query positive and negative statements such as **What is the probability of the previous statement when there is no arc from Smoking to Tuberculosis and no arc from Bronchitis to XRay?**

```
query(mcmcabn = mcmc.2par.asia ,formula = ~LungCancer|Smoking + Bronchitis|Smoking -
        Tuberculosis|Smoking - XRay|Bronchitis)
#> [1] 0.002
```

So essentially zero!

Formula statement for DAG specifications

The **formula** statement has been designed to ease querying over the MCMC samples. Hence, without explicitly writing an adjacency matrix (which can be painful when the number of variables increases). The `formula` argument can be provided using a formula alike:

`~ 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 parents' names have to match those given in `data.dist` exactly. `:` is the separator between either children or parents, `|` separates children (left side), and parents (right side), `+` separates terms, `.` replaces all the variables in name. Then, the arrows go from the left to the right side of the formula. Additionally, when one wants to exclude an arc put `-` in front of that statement. Then a formula like: `~`

`-node1|parent1` excludes all DAGs that have an arc between parent1 and node1. Alternatively, one can query using an adjacency matrix. This matrix should have only: 0,1 and -1. The 1 indicates the requested arcs, the -1 the excluded, and the 0 all other entries that are not subject to constraints. The rows indicate the set of parents of the index nodes. The order of rows and columns should be the same as the ones used in the `mcmcabn()` function in the `data.dist` argument. The matrix should not be named, but it should be squared.

Technical foundations

Note: The terms: Bayesian networks, networks, structures, and graphs are considered as synonyms.

This part aimed at describing the implemented methodology in `mcmcabn`. The rationale for *mcmcabn* methodology is that MCMC over DAGs allows the user to query the family of high scoring DAGs and then draw a more general conclusion than using only one DAG. The rationale for using structural algorithms for sampling the DAG space and not the classical node ordering MCMC described by Friedman and Koller (2003) is that it is biased. Additionally, it is not transparent regarding the prior. One characteristic of the systems epidemiology dataset is that data are scarce. Thus, in a Bayesian setting, the prior play a significant role. Thus it is of high importance to be transparent about our prior belief. Finally, the structural approach allows user-defined a tunable prior, which is of great interest in an applied perspective.

The three main problems addressed by this methodology are:

Selecting the most probable structure based on a cache of pre-computed scores

MCMC over DAGs produces estimates of high scoring structures. The actual alternatives are the `mostProable()` (for exact search) or `search.heuristic()` (for approximate search) R functions in the `abn` R package, for example. The former function performs an exact search, but it is limited to 25 nodes DAGs. The latter performs approximate inference.

Controlling for overfitting

Classically, one wants to select the single most robust DAG, which can then be used for prediction or inference. A major concern in Bayesian network modeling is controlling overfitting. To this end, it is advised to perform parametric or non-parametric bootstrapping in generating samples from a chosen DAG structure or the data and then determining which subset of the selected structure is supported by the observed data. One can control for overfitting using `mcmcabn` R package. Indeed, it returns the set of highly supported structures with their individual support. Inspecting them allows the user to make a general claim about individual arc support.

Marginalizing relationship of interest over nuisance dependencies

Sampling from the posterior distribution of structure could be beneficial in an applied perspective to avoid reducing the richness of Bayesian network modeling to only **one** structure. Indeed, it allows the user to quantify the marginal impact of relationships of interest by marginalizing out over structures or nuisance dependencies, hence accounting for the uncertainty in the structures through Bayesian model averaging. Structural MCMC seems an elegant and natural way to estimate the *true* marginal impact, so one can determine if it’s magnitude is big enough to consider as a worthwhile intervention.

Posterior distribution of the structures

Form a mathematical perspective, one wants to sample over all possible structures in moving from structures to structures according to its support by the data. The posterior distribution of the structures is given by:

$$p(G|D) = \frac{p(G,D)}{p(D)} = \frac{p(D|G)p(G)}{z^*}$$

Where G is a Bayesian network, D the data, and z^* is a normalization factor. In a score-based approach, the likelihood i.e., $p(D|G)$ are precisely the pre-computed scores. In `mcmcabn` they have been pre-computed using `abn` and stored in a cache of scores. Once one get an estimate of the posterior distribution, one can compute the posterior probability of any structural feature (f) from an MCMC sample by

$$E[f|D] \approx \frac{1}{S}\sum_{s=1}^{S} f(G^s), \text{ where } G^s \sim p(G|D)$$

where f is any structural query (`formula` statement in the `query()` function) and S is the set of visited structures G.

Classically, structural MCMC is done using the algorithm described by Madigan and York (1995) and Giudici and Castelo (2003). It is called the Monte Carlo Markov Chain Model Choice ((MC)^3). It is an algorithm based on a single edge move: deletion, addition, or reversal. This algorithm is known to be slow in mixing and easily stuck in local maxima. Multiple updates have been proposed. The idea is to perform a more drastic change in the proposed DAG to cross low probability regions more efficiently.

Friedman and Koller (2003) proposed the order MCMC. This MCMC scheme acts on the node order, improving a lot the convergence of the algorithm. However, this algorithm is biased as a given DAG may belong to a different order. A solution has been proposed by Kuipers and Moffa (2017). It acts on the ordered partitions, which makes DAG having a unique representation. This algorithm is implemented in the R package BiDAG. Goudie and Mukherjee (2016) proposed a Gibbs sampling approach. Finaly, Grzegorczyk and Husmeier (2008) and Su and Borsuk (2016) proposed two drastic MCMC moves. They are implemented in this R package.

Algorithms

Two structurally, prior transparent, large scale moves have been proposed: the new edge reversal move (REV) and the Markov blanket resampling (MBR). The former advocates to make a reversal move in resampling the parent set. The classical reversal move depends on the global configuration of the parents and children and then fails to propose MCMC jumps that produces valid but very different DAGs in a unique move. It is known to be unbiased, but the assumption of ergodicity does not necessarily hold. Then it has to be mixed with (MC)^3 moves. In the MBR the same idea is applied but to the entire Markov blanket of a randomly chosen node.

We believe that having those three algorithms in a unique function with user-adjustable relative frequencies leads to better results than individual implementations. Those three algorithms work very differently. The (MC)^3 is very stable and samples efficiently the nearby region. The REV and MBR potentially produce large scale MCMC jumps but differently and possibly complementary.

The general method is the Metropolis Hasting algorithm. Essentially, it is a sequential application of two steps:

1. a new DAG is proposed from some proposal distribution Q
2. the proposed DAG is accepted with some acceptance probability A

The distribution Q has to be smart. The closer it is from the posterior, the faster will the algorithm converges. To compute the acceptance probability, one needs to compute the posterior probability of each DAG. In this step, the prior on the index structure plays a significant role.

Priors

Three priors are implemented in the `mcmcabn` R package. The parameter `prior.choice` determines the prior used within each node for a given choice of parent combination. In Koivisto and Sood (2004) p.554, a form of prior called the Koivisto prior, is used, which assumes that the prior probability for parent combinations comprising the same number of parents are all equal.

Explicitly, the Koivisto prior is given by

$$p(G) = \frac{1}{z} \prod_{n=1}^{N} \binom{N-1}{|G_n|}^{-1}$$

where N is the total number of nodes and $|G_n|$ is the cardinality of the nth node (i.e. the number of parents). As we are in a MCMC setting and we will compute an Hasting ratio, the normalizing constant z will cancel out.

Note that this prior favors parent combinations with either very low or very high cardinalities, which may not be appropriate. This prior is used when `prior.choice = 2`.

When `prior.choice = 1`, an uninformative prior is used where parent combinations of all cardinalities are equally likely.

When `prior.choice = 3` a user-defined prior is used, defined by `prior.dag`. It is given by an adjacency matrix (squared and same size as number of nodes) where entries ranging from zero to one give the user prior belief. An hyperparameter defining the global user belief in the prior is given by `prior.lambda`. In the vein of Werhli and Husmeier (2007) we defined the biological prior knowledge matrix B where the rows are the parents set and the columns are the children, each entry $B_{ij} \in [0, 1]$ is such that:

- B_{ij} = 0.5, indicates no prior knowledge about the presence or absence of an arc between nodes i and j.
- $B_{ij} \in [0, 0.5[$ indicates a prior belief that there is no arc between node i and j. The belief gets stronger as B_{ij} gets closer to 0.
- $B_{ij} \in]0.5, 1]$ indicates a prior belief that there is an arc between nodes i and j. The belief gets stronger as B_{ij} gets closer to 1.

Such a matrix is still not a prior as we need to introduce a proper normalization procedure. To do so, let us define the energy of a structure as

$$E(G) = \sum_{i,j=1}^{N} |B_{i,j} - G_{i,j}|$$

The energy E is zero for a perfect match between the prior knowledge B and the actual network G, while increasing values of E indicates an increasing divergence between B and G. Following, Imoto et al. (2003) we define the prior belief on graph G by

$$p(G) = \frac{\exp(-\lambda E(G))}{z^*}$$

where z^* is a normalizing constant, and λ is a hyperparameter defined by `prior.lambda`. In an MCMC setting the normalizing constant will canceled out in the Hasting ratio.

In statistical physics, in a Gibbs distribution, the hyperparameter λ is the inverse of temperature. It can be interpreted as a proxy indicating the strength of the influence of the prior over the data. Thus the strength of the user belief in the given prior. For $\lambda \to 0$, the prior distribution becomes flatter then uninformative about the structure. Inversely, for $\lambda \to \infty$, the prior distribution becomes sharply peaked at the network with the lowest energy.

References

- Kratzer, Gilles, et al. (2020). “Bayesian Networks modeling applied to Feline Calicivirus infection among cats in Switzerland.” Frontiers in Veterinary Science 7: 73.
- Madigan, D. and York, J. (1995) “Bayesian graphical models for discrete data”. International Statistical Review, 63:215–232.
- Giudici, P. and Castelo, R. (2003). “Improving Markov Chain Monte Carlo model search for data mining”. Machine Learning, 50:127–158.
- Kratzer, G. and Furrer, R. (2019) “Is a single unique Bayesian network enough to accurately represent your data?”. arXiv preprint arXiv:1902.06641.
- Friedman, N. and Koller, D. (2003). “Being Bayesian about network structure. A Bayesian approach to structure discovery in Bayesian networks.” Machine Learning, 50:95–125, 2003.
- Grzegorczyk, M. and Husmeier, D. “Improving the structure MCMC sampler for Bayesian networks by introducing a new edge reversal move”, Machine Learning, vol. 71(2-3), pp. 265, 2008.
- Su, C. and Borsuk, M. E. “Improving structure MCMC for Bayesian networks through Markov blanket resampling”, The Journal of Machine Learning Research, vol. 17(1), pp. 4042-4061, 2016.
- Koivisto, M. V. (2004). Exact Structure Discovery in Bayesian Networks, Journal of Machine Learning Research, vol 5, 549-573.
- Werhli, A. V., and Husmeier, D. (2007). “Reconstructing gene regulatory networks with Bayesian networks by combining expression data with multiple sources of prior knowledge”. Statistical Applications in Genetics and Molecular Biology, 6 (Article 15).
- Imoto, S., Higuchi, T., Goto, T., Tashiro, K., Kuhara, S., and Miyano, S. (2003). Using Bayesian networks for estimating gene networks from microarrays and biological knowledge. In Proceedings of the European Conference on Computational Biology.
- Goudie, R. J., and Mukherjee, S. (2016). A Gibbs Sampler for Learning DAGs. Journal of machine learning research: JMLR, 17(30), 1-39.
- Kuipers, J. and Moffa, G. (2017). Partition MCMC for Inference on Acyclic Digraphs, Journal of the American Statistical Association, 112:517, 282-299, DOI: 10.1080/01621459.2015.1133426
- Scutari, M. (2010). Learning Bayesian Networks with the bnlearn R Package. Journal of Statistical

Software, 35(3), 1-22.