Research Master's Thesis

Occam's window and Bayesian model averaging for graphical models

David Coba
St. no. 12439665
Research Master's Psychology
University of Amsterdam
Psychological Methods

Supervised by:
Maarten Marsman
16th of August, 2022

Abstract

When we select a statistical model and use it to make inferences about its parameters, we usually ignore the uncertainty derived from the model selection process, leading to overconfident inferences. There are techniques that address this, like Bayesian model averaging (BMA). However, when the space of possible models is vast, such as with graphical models that are popular in psychology, it is not evident how to efficiently find the most relevant models to average over with BMA. Occam's window is a model search algorithm that explores the space of possible models, and it is designed to generate a set of candidate models for BMA.

The goal of this project was to assess in broad terms if Occam's window is a suitable method to explore the model space, specifically in the context of graphical models. To this end we developed an Occam's window implementation, and conducted a simulation study exploring how the algorithm performs under different conditions and how it compared to other alternative model search techniques. Our results show that in its current form, our algorithm underperforms compared to the alternatives. We discuss potential avenues for improving its performance, but it is not clear whether it might be worth to develop it further.

Keywords: Bayesian inference, Bayesian model averaging, model selection, Occam's window, Gaussian graphical model

Research Master's Thesis

Occam's window and Bayesian model averaging for graphical models

Introduction

When we perform statistical inferences, such as hypotheses tests about the inclusion of a parameter in a model or whether a parameter lays within an interval, we typically select a statistical model and then use that model to perform the inference. However, this single-model approach underestimates the total uncertainty in our inferences, since it ignores the uncertainty derived from the model selection process. And, ignoring this uncertainty, leads to overconfident conclusions (Leamer, 1978; Draper et al., 1987; Hoeting et al., 1999; for a recent review of the issue see Kaplan, 2021). The aim of this project in general terms was explore whether an algorithm called Occam's window can be useful to deal with the issue of single-model inference. We wanted to know whether it can produce results that are good enough to be used, while also being able to run in an adequate time frame. To his end, we have developed an implementation of Occam's window that can work with any statistical model and we have conducted a preliminary simulation study to benchmark its performance. Specifically, we were motivated by the issue of deciding whether to include or not particular edges in graphical models that are popular in psychology. The number of possible graphical models grows exponentially with the number of variables, and current approaches to multi-model inference struggle because of the size of the model space.

Different Bayesian solutions have been proposed that allow us to model the uncertainty of the model selection process, which can be categorized into two groups. The first group is using mixture models that encompass all possible models. To estimate the joint posterior distribution of all possible models researchers usually employ simulation based methods like Markov chain Monte Carlo model composition (MC^3, Madigan & York, 1995) or reversible jump Markov chain Monte Carlo (Green, 1995). These algorithms explore the joint posterior distribution by drawing samples from the space of all possible parameter values of all possible models. However, it is often impossible to implement simulation based methods that produce good results in realistic time frames, and they tend to have stability issues (Yao et al., 2018). The second group of approaches to multi-model inference is to only combine the information from a set of pre-specified candidate models $\mathcal{A}$, instead of using the whole model space. To combine a set of candidate models we can use Bayesian model averaging (BMA, Hinne et al., 2020; Hoeting et al., 1999; Leamer, 1978). This approach allows to separate the use of multiple models into two steps: first identifying a set of candidate models $\mathcal{A}$ and then combining the uncertainty from those models. With BMA, the posterior probability of our target inference (Δ, e.g. whether a parameter is included in the model or not) given the observed data, $p(\Delta|D)$, is the weighted average of that inference across all candidate models $p(\Delta|M_k, D),\ M_k \in \mathcal{A}$. BMA uses the posterior probability of candidate models $p(M_k|D)$ as model weights, and our target inference $p(\Delta|D)$ becomes

$$p(\Delta|D) = \sum_{\forall M_k \in \mathcal{A}} p(\Delta|\mathcal{M}_k, D) p(M_k|D).$$

For example, the posterior probability that a specific parameter is included in a model becomes the sum of the posterior probabilities of all models that include that parameter.

From Bayes theorem we know that the posterior probability of a model is the product of the prior probability of that model $p(M_k)$ times the marginal likelihood of the data under that model $p(D|M_k)$, divided by the total probability of the data $p(D)$. Moreover, the total probability of the data $p(D)$ is the weighted sum of the marginal probability of the data under each model (the likelihood of the data, $p(D|M_k)$), using the prior probability of each model $p(M_k)$ as weights. The full expression for the posterior probability of a model becomes:

$$p(M_k|D) = \frac{p(D|M_k)p(M_k)}{\sum_{\forall M_l \in \mathcal{A}} p(D|M_l)p(M_l)}.$$

Note, that if we assume a uniform prior across the model space, the posterior model probabilities become proportional to their marginal likelihoods. And, to calculate the marginal likelihoods we need to integrate the product of the likelihood function of each model $p(D|\theta_k, M_k)$ and the prior distribution of the model parameters $p(\theta_k|M_k)$ over the whole parameter space of that model:

$$p(D|M_k) = \int p(D|\theta_k, M_k) p(\theta_k|M_k) d\theta_k.$$

This is often not possible to do analytically, but there are multiple ways of approximating the marginal likelihood of different models. We will expand more about this in a latter section.

Ideally, we would like to use domain knowledge and theoretical arguments to specify the set of candidate models $\mathcal{A}$ to average with BMA. However, when that is not realistic, we can use model search algorithms instead, to avoid having to consider the whole model space. One possible algorithm is the topic of this thesis: Occam's window, which was introduced by Madigan and Raftery (1994) and is based on Occam's razor principle.

Occam's window as a concept

Occam's razor—also known as the law of parsimony—states that when one is presented with competing hypotheses that explain equally well a particular phenomena, one should choose the simplest one. In general terms, Occam's window approach first selects a set of models that fit the data reasonably well, and then discards from that set all models that have simpler counterparts that fit the data equally well. The final result is the set of simplest models that explain the data well enough, and that set of models is the set of candidate models $\mathcal{A}$ that is considered during BMA.

Formally, the first step equals constructing the set of models

$$\mathcal{A}' = \left\{ M_k : \log \frac{\max\left[p(M_l|D)\right]}{p(M_k|D)} \leq O_L \right\}$$

with posterior probabilities $p(M_k|D)$ not significantly lower than that of the model with highest posterior probability $M_l \in \mathcal{A}'$. The constant O_L specifies the range of posterior probabilities that are acceptable, the size of the window of models that fit well enough. Under the special case when $O_L = 1$, $\mathcal{A}'$ would only include the single model with greatest posterior probability. For the second step, the algorithm identifies a set of models to discard from $\mathcal{A}'$:

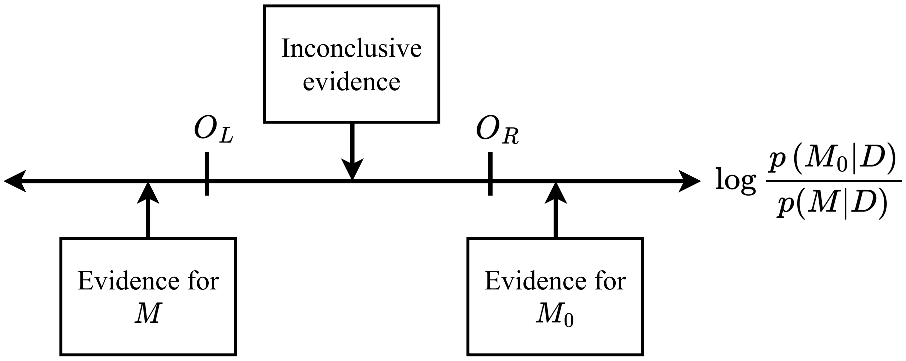
$$\mathcal{B} = \left\{ M_k : \exists M_l \in \mathcal{A}', M_l \subset M_k, \log \frac{p(M_l|D)}{p(M_k|D)} > O_R \right\}$$

that have at least one submodel M_l in $\mathcal{A}'$ with significantly greater posterior probability, and the range of acceptable differences is controlled by the constant O_R. Under the special case when $O_R = 1$, $\mathcal{B}$ would include all models that have a simpler submodel with higher posterior probability, which implies that in the final set of candidate models there would not be any model that is a submodel of other included model. The final set of candidate models is the set of models in the first set that are not present in the second $\mathcal{A} = \mathcal{A}' \setminus \mathcal{B}$. That is, the set of models that fit the data reasonably well and that do not have simpler counterparts that fit the data well enough.

Both constants, O_L and O_R, determine the size of the window—hence the name of the algorithm—of acceptable posterior probabilities to consider, depicted in Figure 1. For any pair of models M and M_0, where M_0 is a submodel of M, we would expect the set of candidate models $\mathcal{A}$ to include either of them, or both of them, depending on the ratio of their posterior probabilities. When the ratio of posterior probabilities lies inside the window, both M and M_0 are included in $\mathcal{A}$.

Figure 1

The "window" from Occam's window, which width is determined by the constants O_L and O_R. M_0 is a submodel of M. Figure adapted from Madigan and Raftery (1994).



However, if we were to translate the formal definition that I have described in this section into a practical algorithm, we would quickly run into issues. Even though the idea behind Occam's window is to avoid having to consider the whole model space for BMA, applying the formal definition literally step-by-step would mean that we have to explore the whole model space anyway. In the next section I will go over two different algorithms that implement Occam's window in a practical way.

Occam's window algorithms

The first algorithm (Algorithm 1) is described in Madigan and Raftery (1994), and it is a deterministic greedy search over the model space. The algorithm starts from a set of initial candidate models $\mathcal{C}$, and performs two passes iteratively deleting (in the first pass) or adding (in the second pass) parameters. In their examples they initialize $\mathcal{C}$ with a single saturated model that includes all possible parameters. They showcase the algorithm with Gaussian graphical models (GGM), where they propose an analytical computation of the marginal likelihood that allows to re-use computations across models, which is ideal for model search algorithms. However, their approach is restricted to chordal graphical structures and they do not report results about its performance, since they only show applied empirical examples and not simulations results.

Algorithm 1 Occam's window algorithm from Madigan and Raftery (1994). An immediate submodel M_0 or supermodel M_1 means that the models differ from the original model M by a single parameter.

```
Require: $\mathcal{C}$
$\mathcal{A} \leftarrow \emptyset$
repeat {Down pass}
    Select a model $M$ from $\mathcal{C}$.
    $\mathcal{C} \leftarrow \mathcal{C} \setminus \{M\}$;  $\mathcal{A} \leftarrow \mathcal{A} \cup \{M\}$
    for inmediate submodel $M_0$ of $M$ do
        Compute $B = \log\left[p\left(M_0|D\right)/p(M|D)\right]$
        if $B > O_R$ then
            $\mathcal{A} \leftarrow \mathcal{A} \setminus \{M\}$
            if $M_0 \notin \mathcal{C}$ then
                $\mathcal{C} \leftarrow \mathcal{C} \cup \{M_0\}$
        else if $O_L \leq B \leq O_R$ then
            if $M_0 \notin \mathcal{C}$ then
                $\mathcal{C} \leftarrow \mathcal{C} \cup \{M\}$
until $\mathcal{C}$ is empty.
$\mathcal{C} \leftarrow \mathcal{A}$;  $\mathcal{A} \leftarrow \emptyset$
repeat {Up pass}
    Select a model $M$ from $\mathcal{C}$.
    $\mathcal{C} \leftarrow \mathcal{C} \setminus \{M\}$;  $\mathcal{A} \leftarrow \mathcal{A} \cup \{M\}$
    for inmediate supermodel $M_1$ of $M$ do
        Compute $B = \log\left[p\left(M|D\right)/p(M_1|D)\right]$
        if $B < O_L$ then
            $\mathcal{A} \leftarrow \mathcal{A} \setminus \{M\}$
            if $M_1 \notin \mathcal{C}$ then
                $\mathcal{C} \leftarrow \mathcal{C} \cup \{M_1\}$
        else if $O_L \leq B \leq O_R$ then
            if $M_1 \notin \mathcal{C}$ then
                $\mathcal{C} \leftarrow \mathcal{C} \cup \{M_1\}$
until $\mathcal{C}$ is empty.
return $\mathcal{A}$
```

The second algorithm (Raftery, 1995; Raftery et al., 1997), Algorithm 2, is implemented for linear, logistic and multinomial-logit regression models in the R packages BMA and mlogitBMA (Raftery et al., 2015), and is a significant deviation from the original approach. This version treats the set of initial candidate models $\mathcal{C}$ as a proxy of the whole model space, and does not fit new submodels or supermodels during the model search. Therefore, we have to use another method to generate $\mathcal{C}$, and then the algorithm eliminates models from that set. The R package BMA uses a leaps-and-bounds algorithm (Furnival & Wilson, 1974) to generate $\mathcal{C}$. The leaps-and-bounds is extremely performant in the linear regression case (less than six floating point operations per model), because it re-uses most calculations across models.

Algorithm 2 Occam's window as implemented in BMA. An immediate submodel M_0 means that the model differ from the original model M by a single parameter.

```
Require: $\mathcal{C}$
for $M|M \in \mathcal{C}$ do
    Compute $B_{max} = \log \frac{\{\max\left[p\left(M_l|D\right)|\forall M_l \in \mathcal{C}\right]}{p(M|D)}$
    if $B_{max} > O_L$ then
        $\mathcal{C} \leftarrow \mathcal{C} \setminus \{M\}$
    else
        for inmediate submodel $M_0$ of $M$ $|M_0 \in \mathcal{C}$ do
            Compute $B = \log\left[p\left(M_0|D\right)/p(M|D)\right]$
            if $B > O_R$ then
                $\mathcal{C} \leftarrow \mathcal{C} \setminus \{M\}$
            else if $B < O_L$ then
                $\mathcal{C} \leftarrow \mathcal{C} \setminus \{M_0\}$
return $\mathcal{A} \leftarrow \mathcal{C}$
```

Lastly, there is a third algorithm, an extension of Occam's window that allows to perform BMA on streams of data that become available sequentially (Onorante & Raftery, 2016), but dealing with streams of data is not relevant for our goals, so we have not considered this algorithm in this project.

Marginal likelihood approximations

Since Occam's window uses marginal likelihoods to compare models many times during the model search, we need efficient ways of approximating them. The first and crudest approximation is to use the Bayesian information criterion (BIC, Schwarz, 1978; Kass & Raftery, 1995). The BIC of a model M_k is defined as

$$\text{BIC}(M_k) = -2 \log p \left(D | \widehat{\theta}, M_k \right) + d_{M_k} \log n,$$

where $p \left(D | \widehat{\theta}, M_k \right)$ is the likelihood function evaluated at the maximum likelihood estimates of the model's parameters, d_{M_k} is the number of parameters in the model and n is the sample size. Kass and Raftery (1995) show that the logarithm of the marginal likelihood of a model can be approximated as

$$\log p \left(D | M_k \right) \approx \log p \left(D | \widehat{\theta}, M_k \right) - \frac{1}{2} d_{M_k} \log n,$$

which means that

$$\log p \left(D | M_k \right) \approx \frac{\text{BIC}(M_k)}{-2}$$

and that the ratio of marginal likelihoods between two models—the Bayes factor—is

$$2 \log B_{ij} = -\text{BIC}(M_i) + \text{BIC}(M_j).$$

This is the approach used by the BMA R package (Raftery et al., 2015). Bridge sampling offers another approach to approximate the marginal likelihood (Bennett, 1976; Gronau et al., 2017). Bridge sampling generally provides accurate approximations of marginal likelihoods, but is also computationally demanding and not usable with a model search algorithm that has to calculate (at least) thousands of marginal likelihoods. This is the case because it is a simulation based method that works drawing samples from the posterior distribution of the parameters of each model, and it requires a high number of samples to accurately estimate the marginal likelihood. A method between BIC and bridge sampling in terms of accuracy and computational demands is the Laplace approximation (Kass & Raftery, 1995; LeCam, 1953). This method approximates the posterior distribution with a normal distribution centered around the posterior mode, which can be estimated using expectation-maximization algorithms. The standard Laplace approximation is accurate to the second moment of the posterior distribution, but it is possible to extend it to get more accurate approximations at the cost of more computational resources or further assumptions (Hubin & Storvik, 2016; Rue et al., 2009; Ruli et al., 2016; Tierney & Kadane, 1986; Tierney et al., 1989). Lastly, for some models, depending on the prior choices there are analytical solutions of the marginal likelihood. Also, note that in the context of Occam's window and BMA, it is possible to use a faster but less accurate approximation during model search, and use a slower but more accurate approximation during the BMA step.

In practice, instead of using posterior probabilities, Occam's window implementations assume an uniform prior across the model space and use the marginal likelihood as a proxy for the posterior model probability. Comparisons based on Bayes factors already penalize models of higher complexity that have similar predictive performance (Kass & Raftery, 1995), but it is also possible to implement different model priors without much computational cost.

Alternative approaches

Under a Bayesian framework, the most common alternative model search algorithm designed to be used for BMA of (generalized) linear models is Bayesian adaptive sampling (BAS, Clyde et al., 2011), which samples without replacement from the space of possible models and uses the marginal likelihoods of the sampled models to iteratively estimate the marginal likelihoods of the models that remain unsampled. BAS is available as an R package (Clyde, 2021).

For GGMs, there are no implementations available of a model search algorithm designed to be used for BMA. However, there are other model search approaches. One of them is birth-death Markov chain Monte Carlo (BDMCMC), which samples from the joint posterior space of all possible models, and uses a Poisson process to model the rate at which the Markov chains jump from one model to another (Mohammadi & Wit, 2015; Mohammadi et al., 2017). BDMCMC is available in the R package BDGraph (Mohammadi & Wit, 2019) for graphical models. Another popular approach is to use Bayes Factors to test whether to include or not specific parameters in a model (Williams & Mulder, 2020a), which is implemented in the R package BGGM (Williams

& Mulder, 2020b). However, this approach is not attempting to explore the model space, but doing pair-wise hypothesis tests for every parameter.

Our Occam's window implementation

During my thesis I have implemented a general version of Occam's window, based on Algorithm 1, in the Julia programming language (Bezanson et al., 2017). Because of Julia's multiple dispatch system, it is possible to use the program with any marginal likelihood approximation and with any statistical model. The only requirements are that the model parameters can be represented as a vector of bits, and that the user defines a function to calculate the marginal likelihood of a model. The way the program is designed allows to easily cache results, and to implement sequential calculations that re-use the computations from one model to the next. Moreover, because of Julia's virtually-zero-cost abstractions, the program can be almost as performant as an implementation in a traditionally compiled language, as long as the marginal likelihood computation is defined in an efficient way. By default the program supports linear regression models and GGMs, both using the BIC approximation to the marginal likelihood. To obtain the maximum likelihood estimates of linear regression models we use Cholesky-decomposed ordinary least squares, and estimation-maximization as implemented in the R package *psychonetrics* for the GGMs. Therefore, the current implementation does not reuse computations across models, although it caches results from previously explored models.

The program also allows to specify different sets of initial candidate models $\mathcal{C}$. The current implementation allows starting from a single saturated model that includes all parameters, a single random model or multiple random models. Additionally, for linear regression models, it also supports the leaps-and-bounds algorithm (Furnival & Wilson, 1974) to generate a set of starting candidate models.

Simulation study

The goal of the simulation study was to assess in broad terms whether Occam's window is a potentially usable approach to Bayesian variable selection in the context of

GGMs. In this section I will describe the procedure we have used and the results we have obtained. However, the current implementation is not the most optimal possible, there is room for improvement and there are multiple modifications that could improve its performance. I will elaborate on the limitations of this study and about potential modifications in the discussion section.

Procedure

Data generation

The simulation study is divided in two parts, simulations for linear regression models and for GGMs.

We used a 4-way design for the linear regression simulations with 3x3x3x6 conditions. The possible number of total variables were $\{5, 10, 20\}$, the number of observations per variable $\{10, 20, 100\}$ and the proportion of the total variables used in the data-generating models $\{1/4, 1/2, 1\}$. If we ended up with a fractional number we rounded it to the lowest integer. For example, if the number of total variables in the simulated data set was 10 and the proportion of total variables was 1/4, the true data-generating model used 2 variables. For each simulation condition, we generated data drawing predictor samples from a normal distribution with $\mu = 0$ and $\sigma = 1$, drawing regression parameters from a normal distribution with $\mu = 0$ and $\sigma = 10$, and adding normal noise to the dependent variables with $\mu = 0$ and $\sigma = 1$.

For each simulated dataset, we used the R package BAS, the R package BMA and our Occam's window implementation. The R package BMA uses default values $O_R = \log(20)$ and $O_L = 1$ during Occam's window, but we noticed during the development of our implementation that sometimes we would obtain better results with both constants being one. This implies that the *window* of Occam's window collapses and that the algorithm becomes a simple greedy search that just selects the model with higher posterior probability at every comparison. To assess this, we did run our implementation with both sets of constants. Additionally, the R package BMA uses leaps-and-bounds to generate a model set that they use as a proxy of the whole model space, and we also wanted to compare how using this set of initial candidate models $\mathcal{C}$ impacts Algorithm 1 performance versus using a single saturated model as recommended by Madigan and Raftery (1994). In total, this leaves us with 6 different model specifications, which are depicted in Table 1. We ran each condition 20 times.

We also used a 4-way design for the GGM simulations, in this case with 2x2x2x3 conditions. The total number of possible variables were $\{5, 10\}$, sample sizes of $\{500, 2000\}$, and $\{0.25, 0.75\}$ as the proportion of sparsity in the data-generating networks—the proportion of omitted edges. To generate positive definite precision matrices we used the procedure followed by Epskamp et al. (2017) and Yin and Li (2011). First, we generate a network structure without weights, just choosing which edges include or not at random. Next, we draw weights from a uniform distribution between 0.5 and 1, and we make them negative half of the time. Then, we set the diagonal elements of the precision matrix to 1.5 times the sum of the absolute values of each row. Finally, we divide each element by the diagonal value of the corresponding row and average the upper and lower triangular matrices to assure that the precision matrix is symmetric. This creates partial correlation networks in which the non-zero edges have a mean of 0.33 and a standard deviation of 0.04. For each simulated dataset, we used the R package BGGM *explore()* function with its default values, the R package BDgraph, also with its default values, and our Occam's window implementation with $O_R = \log(20)$ and $O_L = 1$, as shown in Table 1. We ran each condition 15 times.

The number of simulations per condition is constrained by the available computational resources. All simulations were run in a laptop with an Intel i7-7700HQ CPU processor under less-than-ideal thermal conditions.

Analysis

All models, except BGGM, return the posterior probability of a parameter being present or not in the data-generating model. BGGM returns instead the Bayes factor of a model with that parameter included against a similar model without that parameter. Having those values, and knowing the true parameters included in the data generating models, we can use a decision threshold (e.g. posterior probability greater than 0.5 or

Table 1

Models used in the simulation study.

Model	Algorithm	$\mathcal{C}$	O_R & O_L
Linear regression			
BAS	BAS	-	-
BMA	Occam's window 2	Leaps & bounds	log(20) & 1
Ours	Occam's window 1	Leaps & bounds	log(20) & 1
"	"	"	1 & 1
"	"	Saturated model	log(20) & 1
"	"	"	1 & 1
GGM			
BGGM	Pairwise BF	-	-
BDgraph	BDMCMC	-	-
Ours	Occam's window 1	Saturated model	log(20) & 1

Bayes factor greater than 3) to calculate the sensitivity (i.e. proportion of true positives) and specificity (i.e. proportion of true negatives) of each algorithm. Our plan was to plot sensitivity versus specificity curves using multiple thresholds, and calculate the area under those curves as an indicator of performance. However, the output from all algorithms was very bimodal, with the posterior probabilities of different parameters being very close to 0 or very close to 1—or very close to 0 or in the 10^{10} order of magnitude for the Bayes factors reported by BGGM. Therefore, we have chosen to use a simple decision threshold (posterior probability of 0.5 or Bayes factor of 3) to calculate the sensitivity and specificity of each algorithm.

We also planned on reporting the average time per run of each algorithm as a measure of computational speed. However, there are two issues with this metric. First, the laptop used to run the simulations experienced significant thermal throttling in some cases. Second, because of computational constrains we had to set a time limit to our implementation of Occam's window with GGMs. If a single run of the algorithm would still be running after 1 hour, it wold be a timeout, and the program would return the current set of accepted models $\mathcal{A}$. We kept track of which runs timed-out, and we indented to present their results separated from those of runs that executed in less than 1 hour. However, when a simulation timed out, all simulations of the same condition ended up timing-out as well. We mark in the results sections which conditions produced time-outs. Therefore, because of these two limitations any comparison across the algorithms' runtime would be unfair. However, their runtime are in different orders of magnitude altogether, so a rough approximation of their runtime should not be less informative that an accurate one.

Results

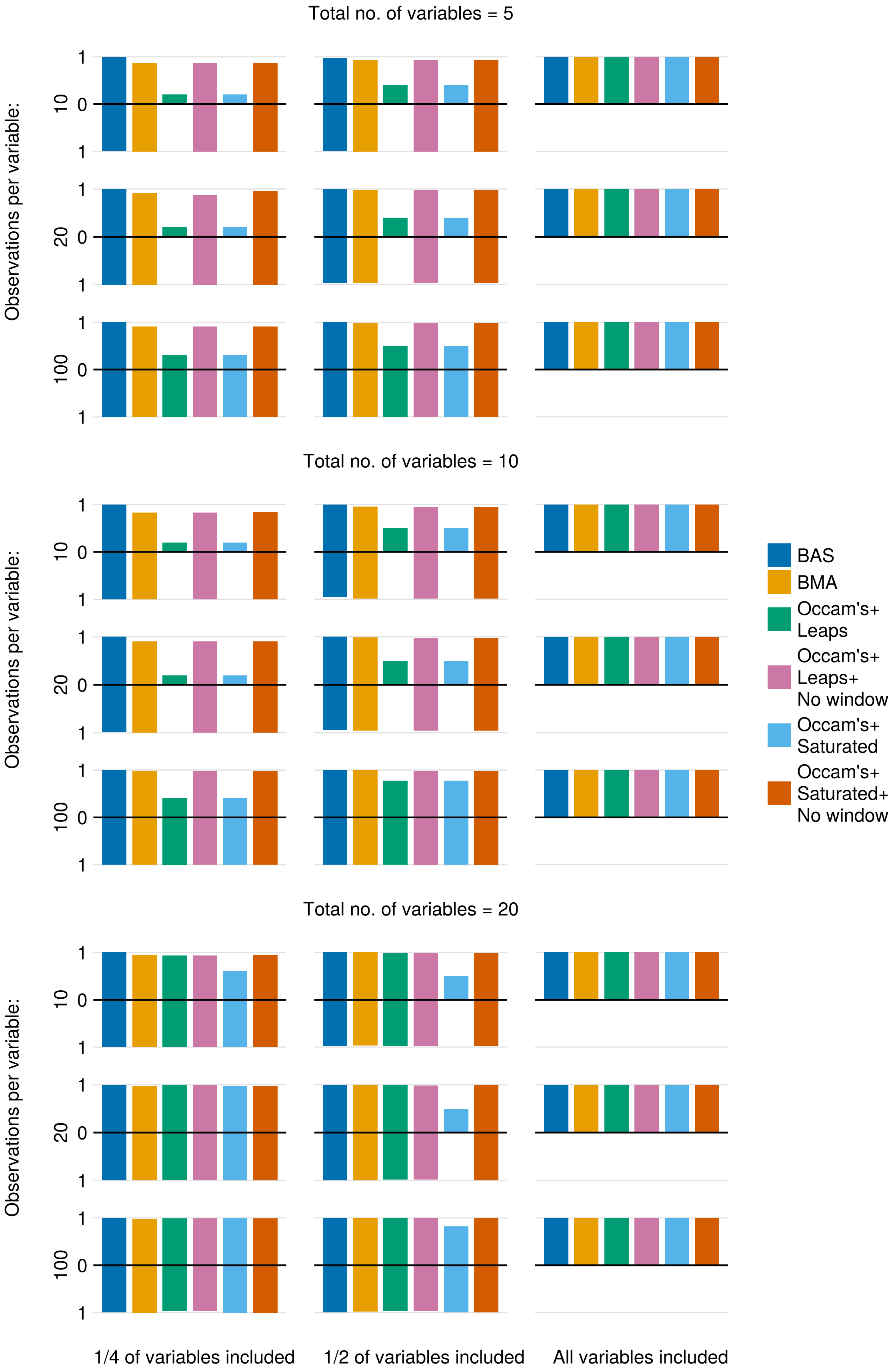
The bar plots of every simulation condition represent the sensitivity (top bar) and specificity (bottom bar) of each algorithm. If both sides of a bar are full it means that the algorithm performed very well under those conditions, and the shorter they are, the worse it performed.

Figure 2 shows the results for the linear regression simulations. We can see that both BMA and BAS performed very well, since they achieved sensitivities and specificities of (almost) 1 under every condition. This was also the case for our implementation of Occam's window when we set both constants to 1 and Occam's window *collapses* into a simple greedy step-wise search. On the other hand, our Occam's window implementation achieved very poor results under most conditions when used as intended (i.e. with the constants specifying a window of accepted models). Lastly, note that all algorithms identified correctly when the true data-generating model was the saturated model, as shown in the last column of Figure 2. In this case, all algorithms had a sensitivity of 1, and a specificity of 0, since there were no true missing edges. Our implementation performed better when the total number of variables considered by the algorithm was highest, when using a *good* set of initial candidate models (i.e. the one generated by the leaps-and-bounds algorithm), and, predictably, when the sample sizes were greater.

In terms of running time, both BAS and BMA outperformed our implementation. BAS executed typically in less than 2 seconds; BMA in a fraction of a second, often less than 100 milliseconds; and our implementation run for a few seconds, usually less than 5.

Figure 2

Simulation results of the linear regression models. The top of each bar is the sensitivity and the bottom the specificity of each algorithm.

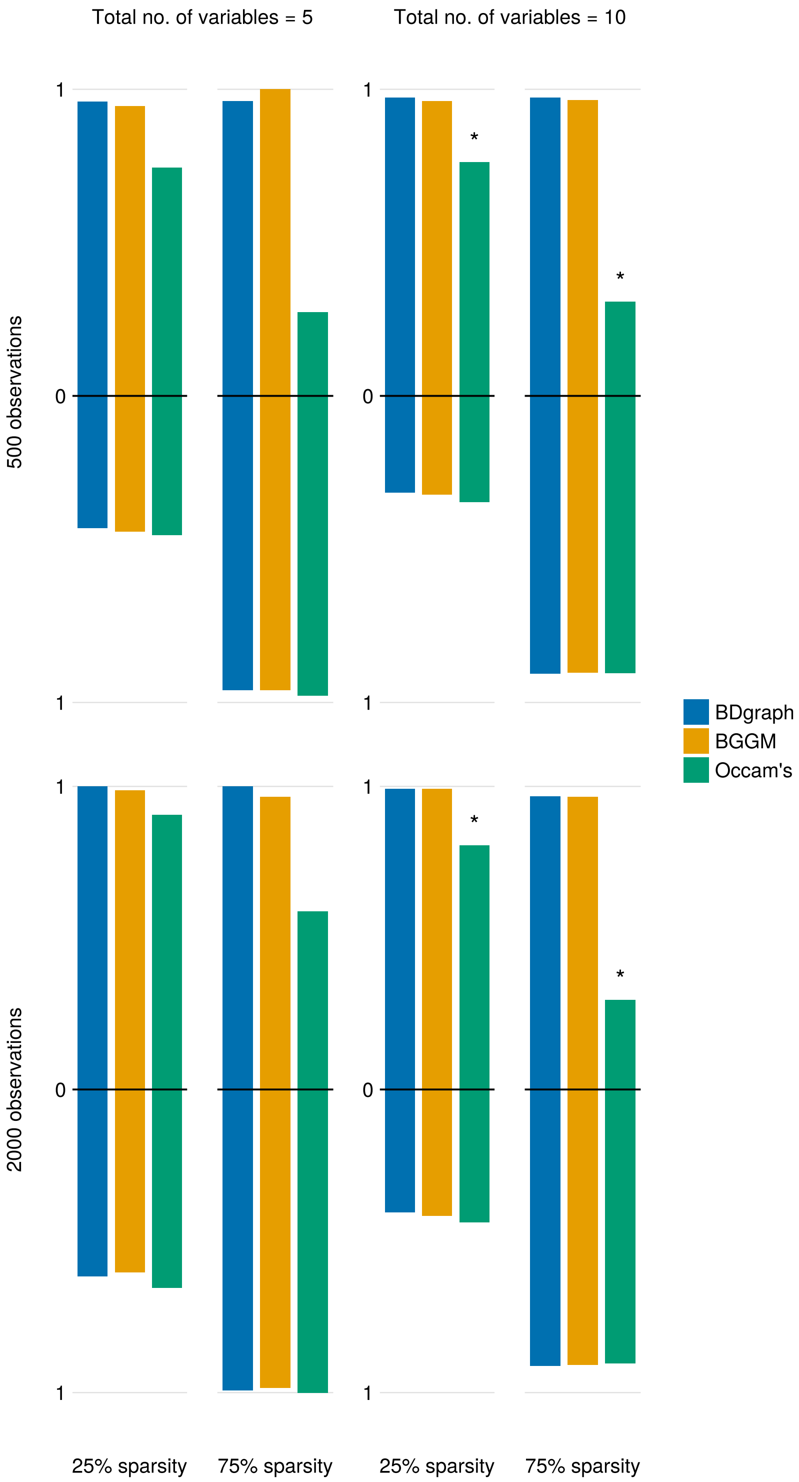


In figure 3 we can see the results from the graphical models simulations. In conditions with high sparsity both BDgraph and BGGM performed very well, while our implementation tended to include parameters that were not present in the data-generating models. However, in conditions of lower sparsity all algorithms had low specificity and the differences in performance were less pronounced, although our implementation was still outperformed by the alternatives.

On the other hand, when considering execution time, the differences across the three algorithms were more pronounced than in the linear regression case. BGGM executed in a fraction of a second and BDgraph in a few seconds, but usually less than 5 seconds. However, our implementation took over 30 minutes for every run, even exceeding 1 hour of runtime and timing-out in half of the simulation conditions, those where the number of variables was 10.

Figure 3

Simulation results of the GGMs. The top of each bar is the sensitivity and the bottom the specificity of each algorithm. Bars marked with an asterisk correspond to simulations that timed-out.



Discussion

The goal of this thesis project was to asses whether Occam's window is a potentially useful algorithm to explore the space of possible models, specially in the case of graphical models. Our results show that, at least for now, our implementation of Occam's window is not capable of matching the performance of alternative algorithms. Neither in terms of accuracy of the results nor in terms of computational speed. Our implementation runs in an acceptable time frame when dealing with linear regression models, but its performance is worse than the alternatives. In the case of GGMs the performance of our implementation was closer to that of the alternatives, albeit still significantly underperforming, but it took a prohibitively amount of time to execute.

The algorithms we have benchmarked our implementation against, BGGM and BDgraph, work very well when dealing with simple cases (i.e. continuous, normally distributed data), but their performance is not as good under more complex conditions. These method are essentially as optimized as they can be, while our implementation is still very bare-bones. In this sense, Occam's window still has the potential of outperforming the alternatives in more complex scenarios. There are two main avenues for improving the performance of our implementation that we think have the most potential. The first one, would be to implement a method of calculating the marginal likelihoods that allows to re-use computations across different models, to improve its execution speed. The standard way of obtaining the maximum likelihood estimates required to calculate the BIC in the case of the GGMs is to use estimation-maximization methods, which make it difficult to re-use computations across models. We could instead use pseudolikelihoods functions like those proposed in Pensar et al. (2017) or Mohammadi et al. (2017), for which deriving sequential calculations might be possible. The second avenue to improve Occam's window performance is to use a method to find a good set of initial candidate models. One of the keys of the excellent performance of the BMA R package is the quality of the set of initial candidate models generated by the leaps-and-bounds algorithm. In our simulations with GGMs we just used a single saturated model as a starting point, and using a better set of initial models might significantly improve Occam's window performance—although that was not always the case in our linear regression results. If the quality of the initial set was good enough to act as a proxy of the full model space, it would even be an option to use Algorithm 2 instead of Algorithm 1. A potential method of reducing the model space that Occam's window has to explore is the approach proposed by Marsman et al. (2022), which allows to discard edges from the total set of possible edges in graphical models. However, although this approach will limit the space of models to explore, it will not necessarily help in finding a set of good initial candidate models from that reduced model space.

Our results do not show that Occam's window is a very promising method, but they do not show either that it is clearly an impractical algorithm to use with graphical models. Whether it is worth pursuing the development of an implementation of Occam's window for graphical models is not a question that we can answer in absolute terms. It depends on our personal trade-offs between the perceive importance of the issue of variable selection in graphical models, and the potential increase in performance versus alternative approaches.

Materials

All project materials are available in the following GitHub repository: https://github.com/cobac/resma-thesis . It includes the code of our Occam's window implementation, code for all simulations and analyses, simulations raw output and intermediary analysis results.

References

Bennett, C. H. (1976). Efficient estimation of free energy differences from monte carlo data. *Journal of Computational Physics*, *22*(2), 245–268. https://doi.org/10.1016/0021-9991(76)90078-4

Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. (2017). Julia: A fresh approach to numerical computing. *SIAM Review*, *59*(1), 65–98. https://doi.org/10.1137/141000671

Clyde, M. A. (2021). *BAS: Bayesian Variable Selection and Model Averaging using Bayesian Adaptive Sampling.* https://cran.r-project.org/package=BAS

Clyde, M. A., Ghosh, J., & Littman, M. L. (2011). Bayesian adaptive sampling for variable selection and model averaging. *Journal of Computational and Graphical Statistics*, *20*(1), 80–101. https://doi.org/10.1198/jcgs.2010.09049

Draper, D., Hodges, J. S., Leamer, E. E., Morris, C. N., & Rubin, D. B. (1987). A research agenda for assessment and propagation of model uncertainty. *Rand Corporation*, *Report N-2683-RC*.

Epskamp, S., Rhemtulla, M., & Borsboom, D. (2017). Generalized network psychometrics: Combining network and latent variable models. *Psychometrika*, *82*(4), 904–927. https://doi.org/10.1007/s11336-017-9557-x

Furnival, G. M., & Wilson, R. W. (1974). Regressions by leaps and bounds. *Technometrics*, *16*(4), 499–511. https://doi.org/10.1080/00401706.1974.10489231

Green, P. J. (1995). Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, *82*(4), 711–732. https://doi.org/10.1093/biomet/82.4.711

Gronau, Q. F., Sarafoglou, A., Matzke, D., Ly, A., Boehm, U., Marsman, M., Leslie, D. S., Forster, J. J., Wagenmakers, E.-J., & Steingroever, H. (2017). A tutorial on bridge sampling. *Journal of Mathematical Psychology*, *81*, 80–97. https://doi.org/10.1016/j.jmp.2017.09.005

Hinne, M., Gronau, Q. F., van den Bergh, D., & Wagenmakers, E.-J. (2020). A conceptual introduction to Bayesian model averaging. *Advances in Methods and Practices in Psychological Science*, *3*(2), 200–215. https://doi.org/10.1177/2515245919898657

Hoeting, J. A., Madigan, D., Raftery, A. E., & Volinsky, C. T. (1999). Bayesian model averaging: A tutorial [with comments by M. Clyde, David Draper and EI George, and a rejoinder by the authors]. *Statistical Science*, *14*(4), 382–417. https://doi.org/10.1214/ss/1009212519

Hubin, A., & Storvik, G. (2016). Estimating the marginal likelihood with integrated nested Laplace approximation (INLA). http://arxiv.org/abs/1611.01450v1

Kaplan, D. (2021). On the quantification of model uncertainty: A Bayesian perspective. *Psychometrika*, *86*(1), 215–238. https://doi.org/10.1007/s11336-021-09754-5

Kass, R. E., & Raftery, A. E. (1995). Bayes factors. *Journal of the American Statistical Association*, *90*(430), 773–795. https://doi.org/10.1080/01621459.1995.10476572

Leamer, E. E. (1978). *Specification searches: Ad hoc inference with nonexperimental data*. Wiley.

LeCam, L. (1953). On some asymptotic properties of maximum likelihood estimates and related Bayes estimates. *University of California Publication in Statististics*, *1*(11), 277–330.

Madigan, D., & Raftery, A. E. (1994). Model selection and accounting for model uncertainty in graphical models using Occam's window. *Journal of the American Statistical Association*, *89*(428), 1535–1546. https://doi.org/10.1080/01621459.1994.10476894

Madigan, D., & York, J. (1995). Bayesian graphical models for discrete data. *International Statistical Review*, *63*(2), 215–232. https://doi.org/10.2307/1403615

Marsman, M., Huth, K., Waldorp, L., & Ntzoufras, I. (2022). Objective bayesian edge screening and structure selection for ising networks.

Mohammadi, A. R., & Wit, E. C. (2015). Bayesian structure learning in sparse Gaussian graphical models. *Bayesian Analysis*, *10*(1), 109–138. https://doi.org/10.1214/14-BA889

Mohammadi, A. R., Massam, H., & Letac, G. (2017). Accelerating Bayesian structure learning in sparse Gaussian graphical models. http://arxiv.org/abs/1706.04416v3

Mohammadi, A. R., & Wit, E. C. (2019). BDgraph: An R package for Bayesian structure learning in graphical models. *Journal of Statistical Software*, *89*(3), 1–30. https://doi.org/10.18637/jss.v089.i03

Onorante, L., & Raftery, A. E. (2016). Dynamic model averaging in large model spaces using dynamic Occams window. *European Economic Review*, *81*, 2–14. https://doi.org/https://doi.org/10.1016/j.euroecorev.2015.07.013

Pensar, J., Nyman, H., Niiranen, J., & Corander, J. (2017). Marginal pseudo-likelihood learning of discrete Markov network structures. *Bayesian Analysis*, *12*(4). https://doi.org/10.1214/16-ba1032

Raftery, A. E., Madigan, D., & Hoeting, J. A. (1997). Bayesian model averaging for linear regression models. *Journal of the American Statistical Association*, *92*(437), 179–191. https://doi.org/10.1080/01621459.1997.10473615

Raftery, A. E. (1995). Bayesian model selection in social research. *Sociological methodology*, 111–163.

Raftery, A. E., Hoeting, J., Volinsky, C., Painter, I., & Yeung, K. Y. Y. (2015). *BMA: Bayesian model averaging*. https://cran.r-project.org/package=BMA

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 (Statistical Methodology)*, *71*(2), 319–392. https://doi.org/10.1111/j.1467-9868.2008.00700.x

Ruli, E., Sartori, N., & Ventura, L. (2016). Improved Laplace approximation for marginal likelihoods. *Electronic Journal of Statistics*, *10*(2), 3986–4009. https://doi.org/10.1214/16-EJS1218

Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics*, 461–464. http://www.jstor.org/stable/2958889

Tierney, L., & Kadane, J. B. (1986). Accurate approximations for posterior moments and marginal densities. *Journal of the American Statistical Association*, *81*(393), 82–86.

Tierney, L., Kass, R. E., & Kadane, J. B. (1989). Fully exponential Laplace approximations to expectations and variances of nonpositive functions. *Journal of the American Statistical Association*, *84*(407), 710–716. https://doi.org/10.1080/01621459.1986.10478240

Williams, D. R., & Mulder, J. (2020a). Bayesian hypothesis testing for gaussian graphical models: Conditional independence and order constraints. *Journal of*

Mathematical Psychology, *99*, 102441. https://doi.org/https://doi.org/10.1016/j.jmp.2020.102441

Williams, D. R., & Mulder, J. (2020b). Bggm: Bayesian gaussian graphical models in r. *Journal of Open Source Software*, *5*(51), 2111. https://doi.org/10.21105/joss.02111

Yao, Y., Vehtari, A., Simpson, D., & Gelman, A. (2018). Using stacking to average Bayesian predictive distributions [with Discussion]. *Bayesian Analysis*, *13*(3), 917–1007. https://doi.org/10.1214/17-BA1091

Yin, J., & Li, H. (2011). A sparse conditional gaussian graphical model for analysis of genetical genomics data. *The Annals of Applied Statistics*, *5*(4), 2630. https://doi.org/10.1214/11-AOAS494