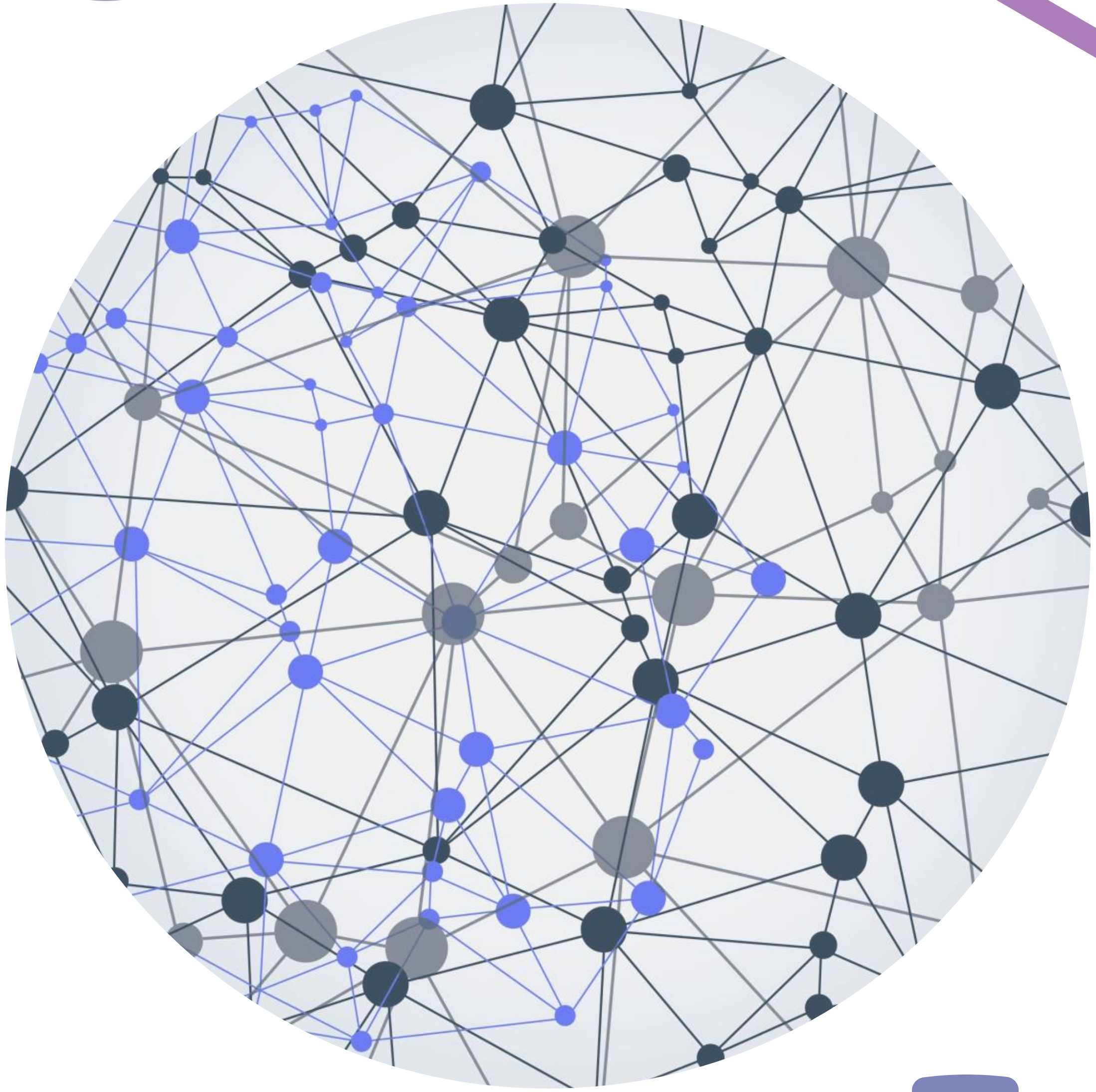


Occam's window and Bayesian model averaging for graphical models

Research Master's Thesis

David Coba

22 of July of 2022

Supervised by:
Maarten Marsman

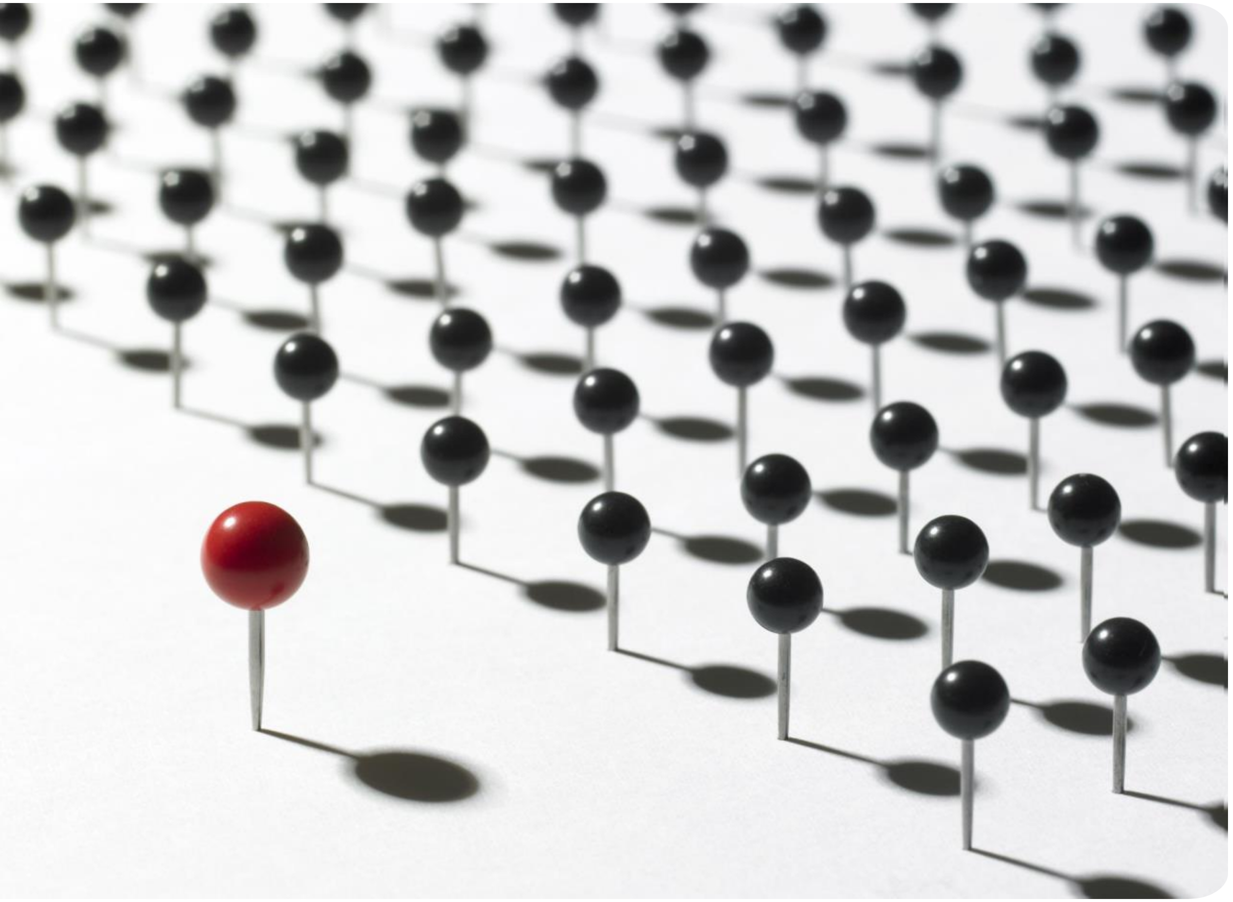
Overview

- The problem of single model inference and Bayesian model averaging
- The idea behind Occam's window
- Occam's window algorithms
  - (Including mine!)
- Simulation study
- Discussion

One question.

What is the probability that a parameter is included in a statistical model?

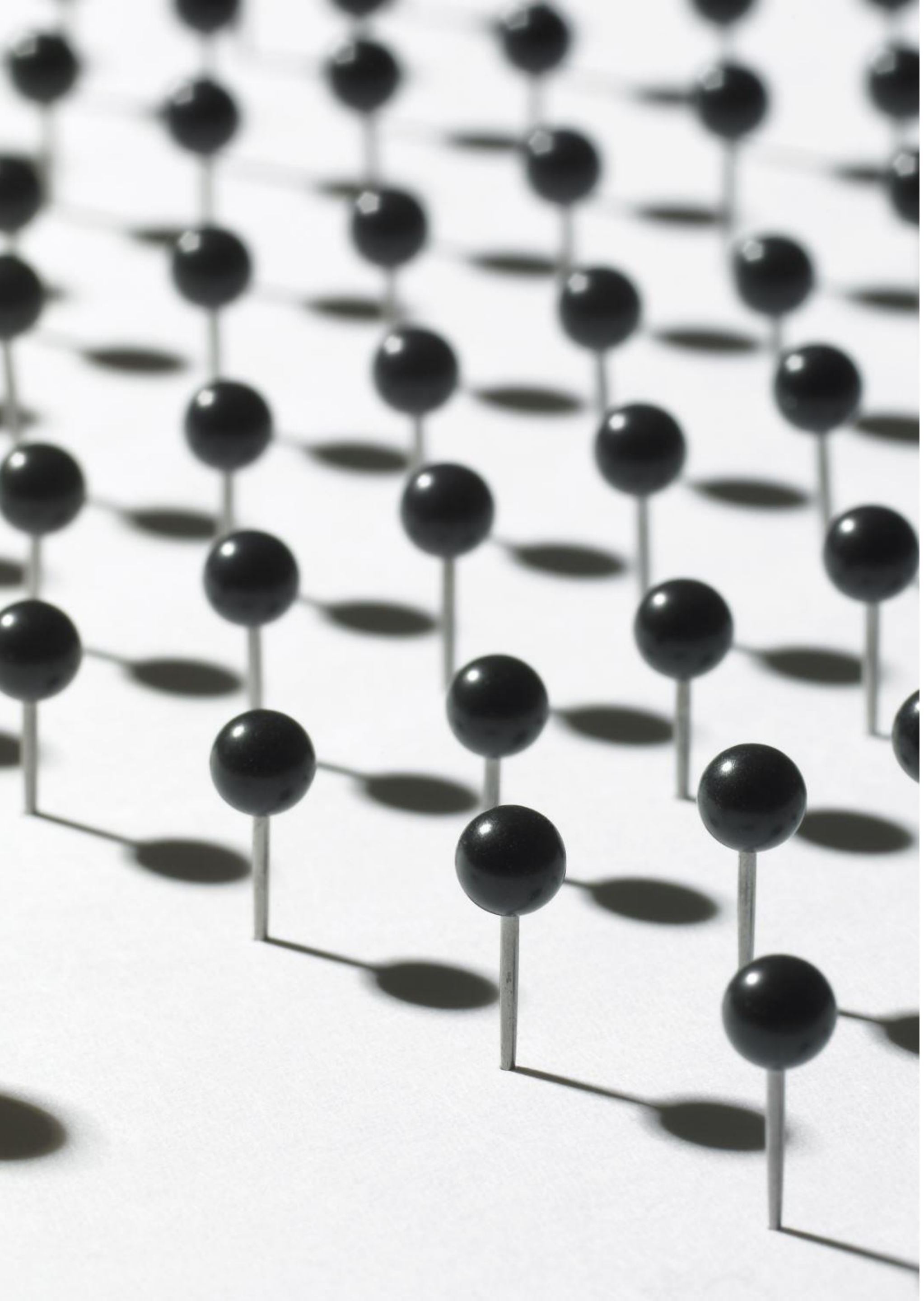
...which is a hard question, particularly when the number of possible models increases exponentially with the number of variables, like with graphical models.



Select a statistical model

Use that model to perform the inference

Is the value of the parameter close to 0?
Compare vs null model, CIs, the other CIs, etc.



Issues

- Ignore the uncertainty derived from the model selection process
  - overconfident inferences
- The parameter estimates are always conditioned on the rest of the parameters of the model

Solution

Don't use a single model, but instead model the uncertainty derived from the model selection process

Two approaches under a Bayesian framework

Big mixture models

- They consider the full model space
- Simulation based (Markov chain Monte Carlo) methods (MC^3, RJMCMC)
- Their solutions tend to be unstable, hard to implement efficiently, high computational cost

Bayesian model averaging (BMA)

- Combine the uncertainty from (only) a set of candidate models

$$P(\text{Parameter}) = \sum_{\text{For all candidate models}} P(\text{Parameter} \mid \text{Model}) \cdot P(\text{Model} \mid \text{Data})$$

Bayesian model averaging (BMA)

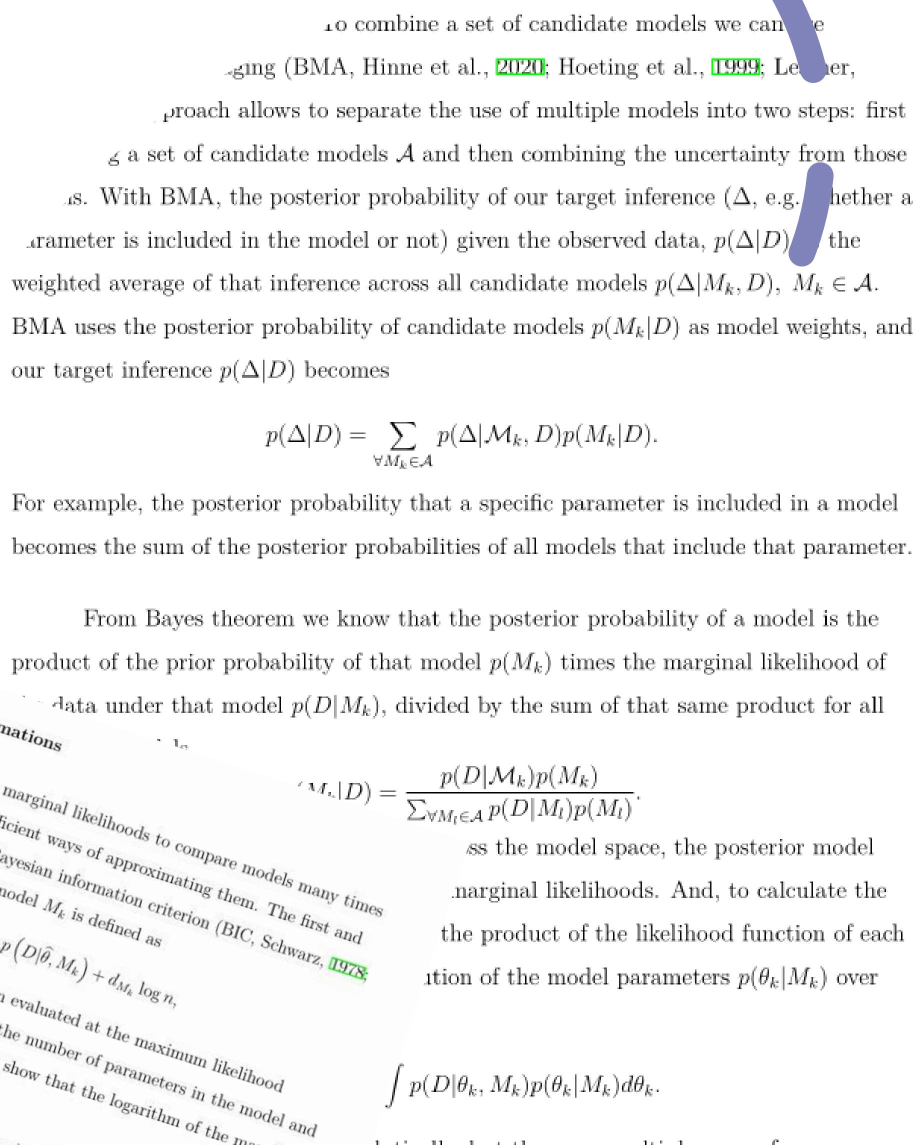
- Combine the uncertainty from (only) a set of candidate models

$$P(\text{Parameter}) = \sum_{\text{For all candidate models}} P(\text{Parameter} \mid \text{Model}) \cdot P(\text{Model} \mid \text{Data})$$

Posterior probability of a model, P(Model | Data)

- Depends on the marginal likelihood of the data under that model
- Solving an integral over the whole parameter space

- Not tractable, we need to use approximations



o combine a set of candidate models we can e

ging (BMA, Hinne et al., 2020; Hoeting et al., 1999; Le er,

proach allows to separate the use of multiple models into two steps: first

a set of candidate models $\mathcal{A}$ and then combining the uncertainty from those

s. With BMA, the posterior probability of our target inference (Δ, e.g. hether a

arameter is included in the model or not) given the observed data, $p(\Delta|D)$, 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

data under that model $p(D|M_k)$, divided by the sum of that same product for all

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

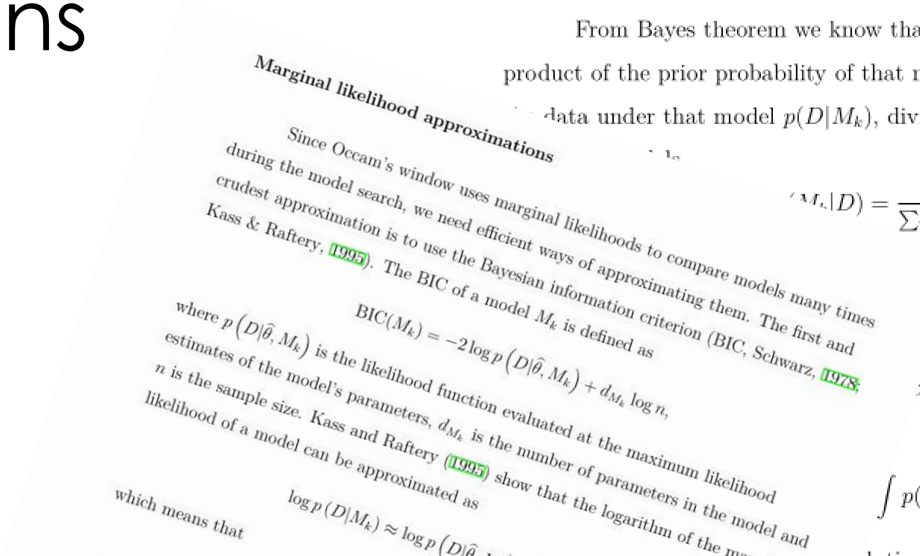
ss the model space, the posterior model

narginal likelihoods. And, to calculate the

the product of the likelihood function of each

ıtion of the model parameters $p(\theta_k|M_k)$ over

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



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

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

where $p\left(D|\hat{\theta}, M_k\right)$ 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 ma likelihood of a model can be approximated as

$$\log p(D|M_k) \approx \log p(D|\hat{\theta}$$

which means that

Bayesian model averaging (BMA)

- Combine the uncertainty from (only) a set of candidate models

$$P(\text{Parameter}) = \underbrace{\sum}_{\text{For all candidate models}} P(\text{Parameter} \mid \text{Model}) \cdot P(\text{Model} \mid \text{Data})$$

How to generate a set of candidate models?

- Ideally, domain knowledge and theoretical arguments.

- Not realistic in a lot of scenarios, so…

Model search algorithms, like

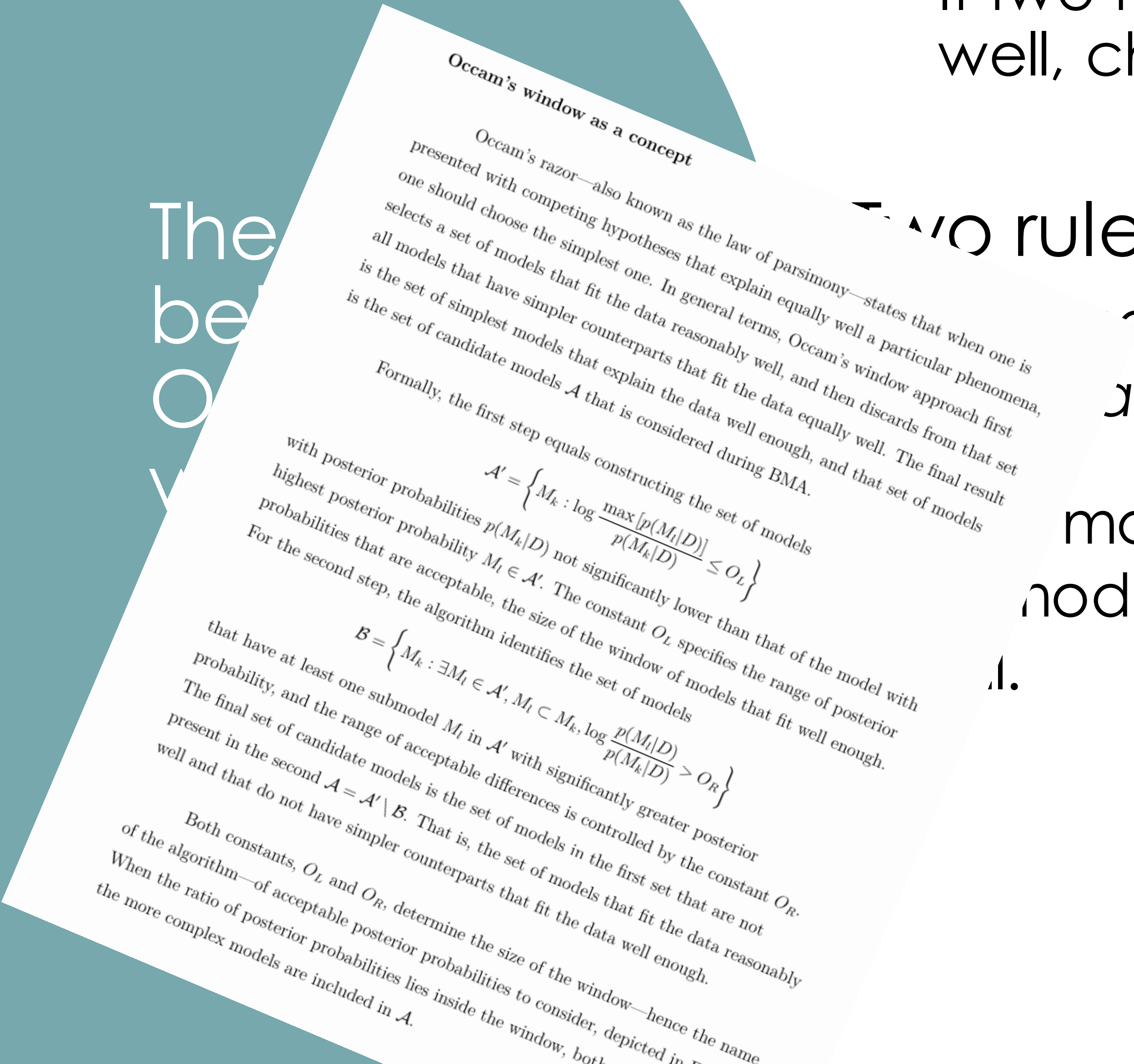
Occam's window

Is Occam's window a potentially useful algorithm to explore the model space of GGMs?

Does it produce OK results?
Does it work in a reasonable timeframe?

The idea behind Occam's window

The
be
O
w



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

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

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.

For the second step, the algorithm identifies the set of models

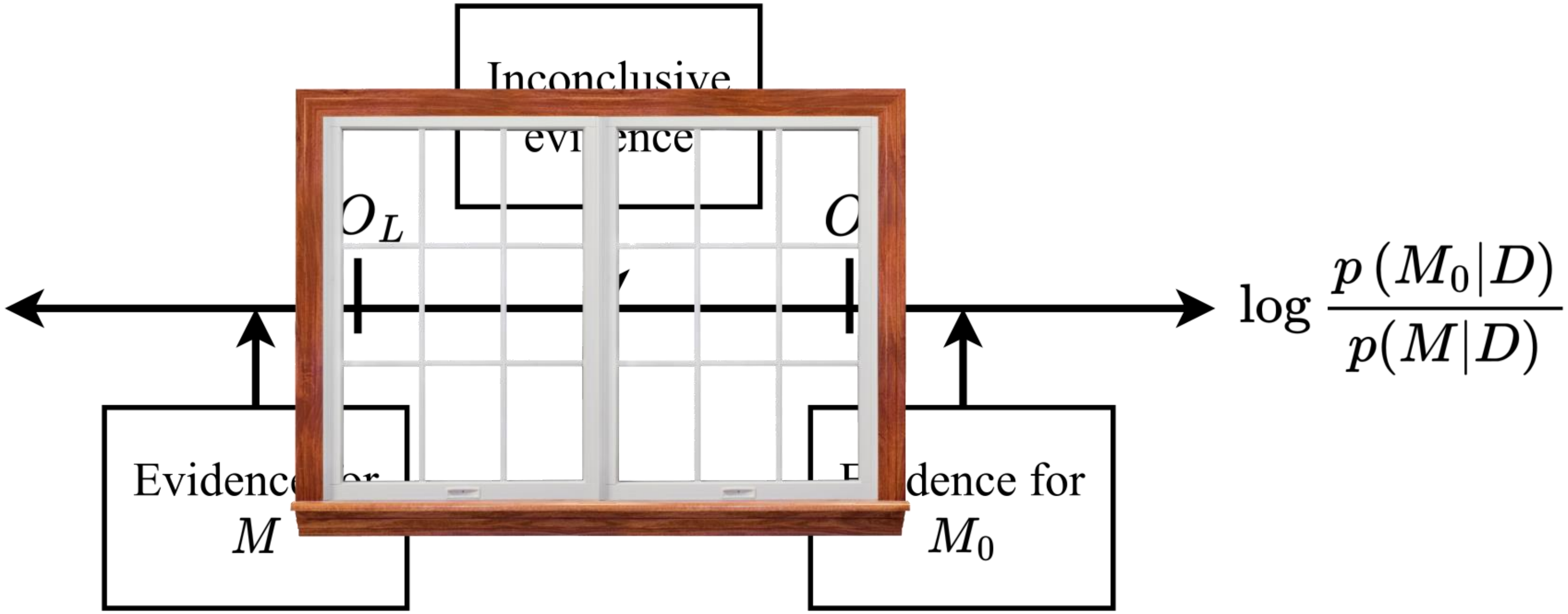
$$\mathcal{B} = \left\{ M_k : \exists M_l \in \mathcal{A}', M_l \subset M_k, \log \frac{p(M_l|D)}{p(M_k|D)} > O_R \right\}$$

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. 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 F… When the ratio of posterior probabilities lies inside the window, bot… the more complex models are included in $\mathcal{A}$.

- Generalization of Occam's razor
  - If two models explain the data equally well, choose the simplest one.

- …o rules to consider a model:
  - …odel should explain the data …antly worse than the best model.
  - … model should have simpler …nodels that explain the data *equally* …l.


Inconclusive
O_L
$\log \frac{p(M_0|D)}{p(M|D)}$
Evidence
M
dence for
M_0

Occam's window algorithms

Algorithm 1 (original from Raftery & Madigan, 1994)

- Start from a set of initial candidate models
  - Single saturated model

- Deterministic step-wise greedy search testing whether models fall inside the window or not

Algorithm 2 (R package BMA)

- Start from a set of initial candidate models as a proxy of the **full** model space.
  - Super-fast leaps-and-bounds
  - Only linear (and logistic) regression models

- Check which models are in the window
  - Does not require fitting new models!

Occam's window algorithms

Algorithm 1 (original from Raftery & Madigan, 1994)

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 $\mathcal{A}$

Algorithm 2 (R package BMA)

Algorithm 2 Occam's window as implemented in BMA.

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 $\mathcal{A} \leftarrow \mathcal{C}$

Our Occam's window implementation

Based on Algorithm 1

- Because of Julia's multiple dispatch
  - Works with any statistical model
  - The model parameters can be represented as a vector of bits
  - The user implements a marginal likelihood calculation

- Because of Julia's virtually-zero-cost abstractions
  - *Potentially* as fast as an implementation in a compiled language
  - If the marginal approximation is implemented in an efficient way

- BIC as an approximation to the marginal likelihood, like BMA.

Simulation study

4-way design

Linear regression

- No. of variables = {5, 10, 20}
- No. of observations per variable = {10, 20, 100}
- Variables used in the data-generating model = {¼, ½ , 1}
- Predictors generated from N(0, 1), parameters from N(0, 10) and noise from N(0, 1)
- 20 datasets per condition.

GGM

- No. of variables = {5, 10}
- No. of observations = {500, 2000}
- Sparsity = {25%, 75%}
- Precision matrices generated like Epskamp et al. (2017) & Yin and Li (2011) do.
- 15 datasets per condition.

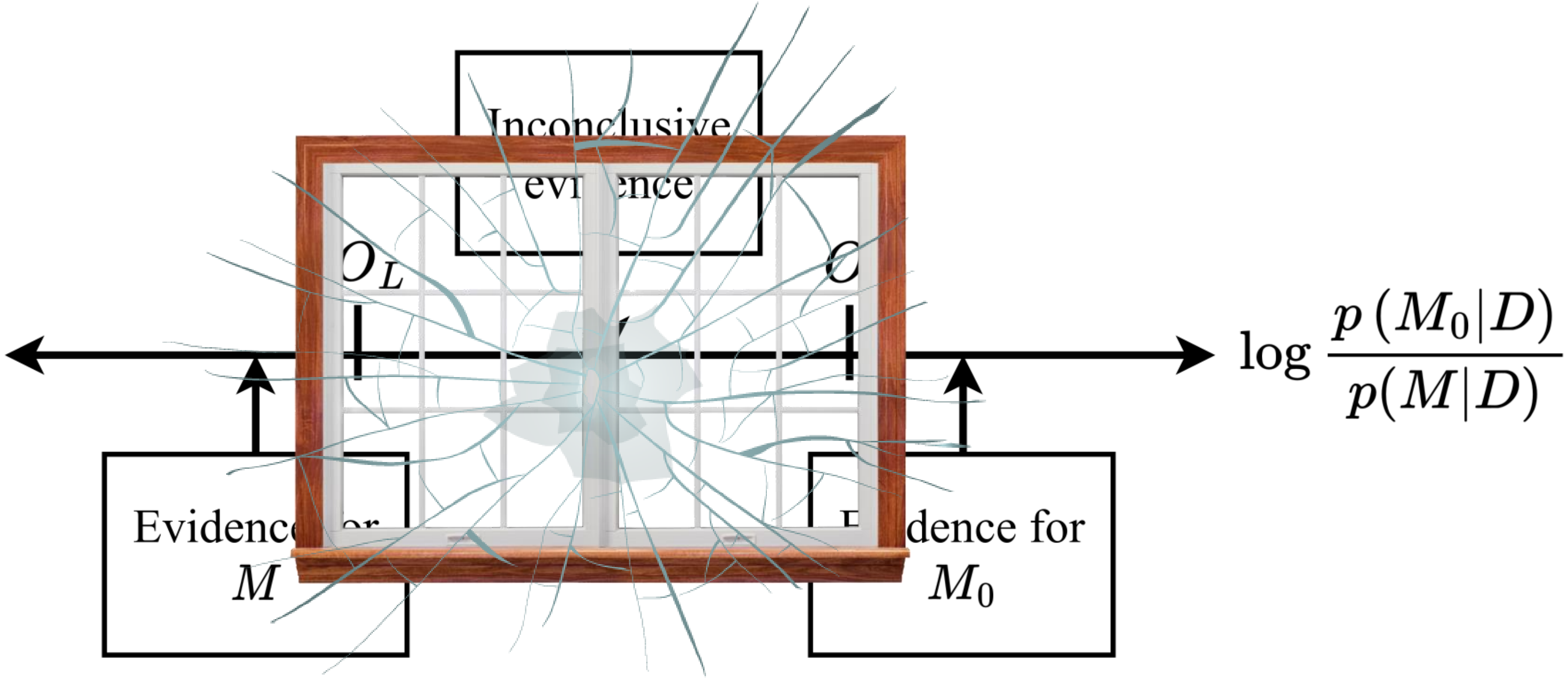
Simulation study

4-way design

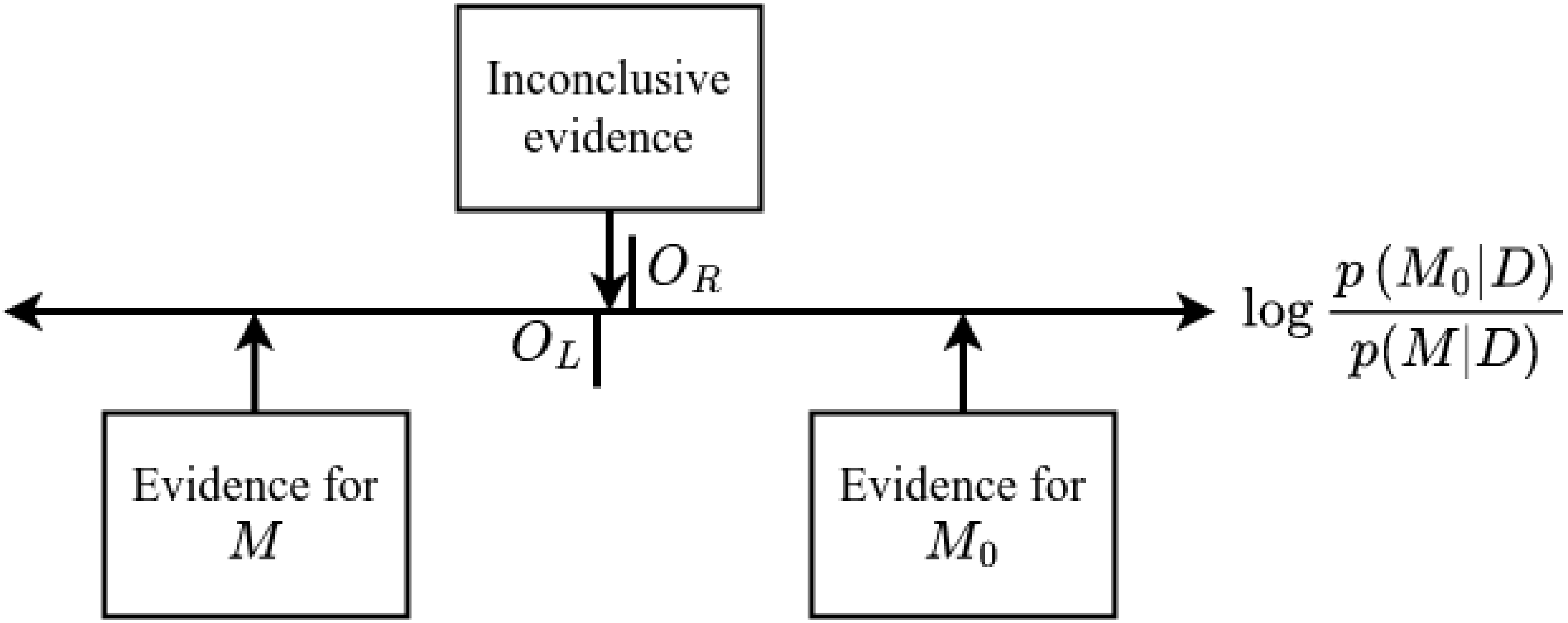
Linear regression

Model	Algorithm	Starting models	Constants
BAS	BAS	-	-
BMA	Occam's window 2	Leaps & Bounds	Normal
Ours	Occam's window 1	Leaps & Bounds	Normal
"	"	"	Collapsed
"	"	Saturated model	Normal
"	"	"	Collapsed

Collapsed constants?



Collapsed constants?



Just a step-wise search that selects the model with higher posterior probability at every iteration.

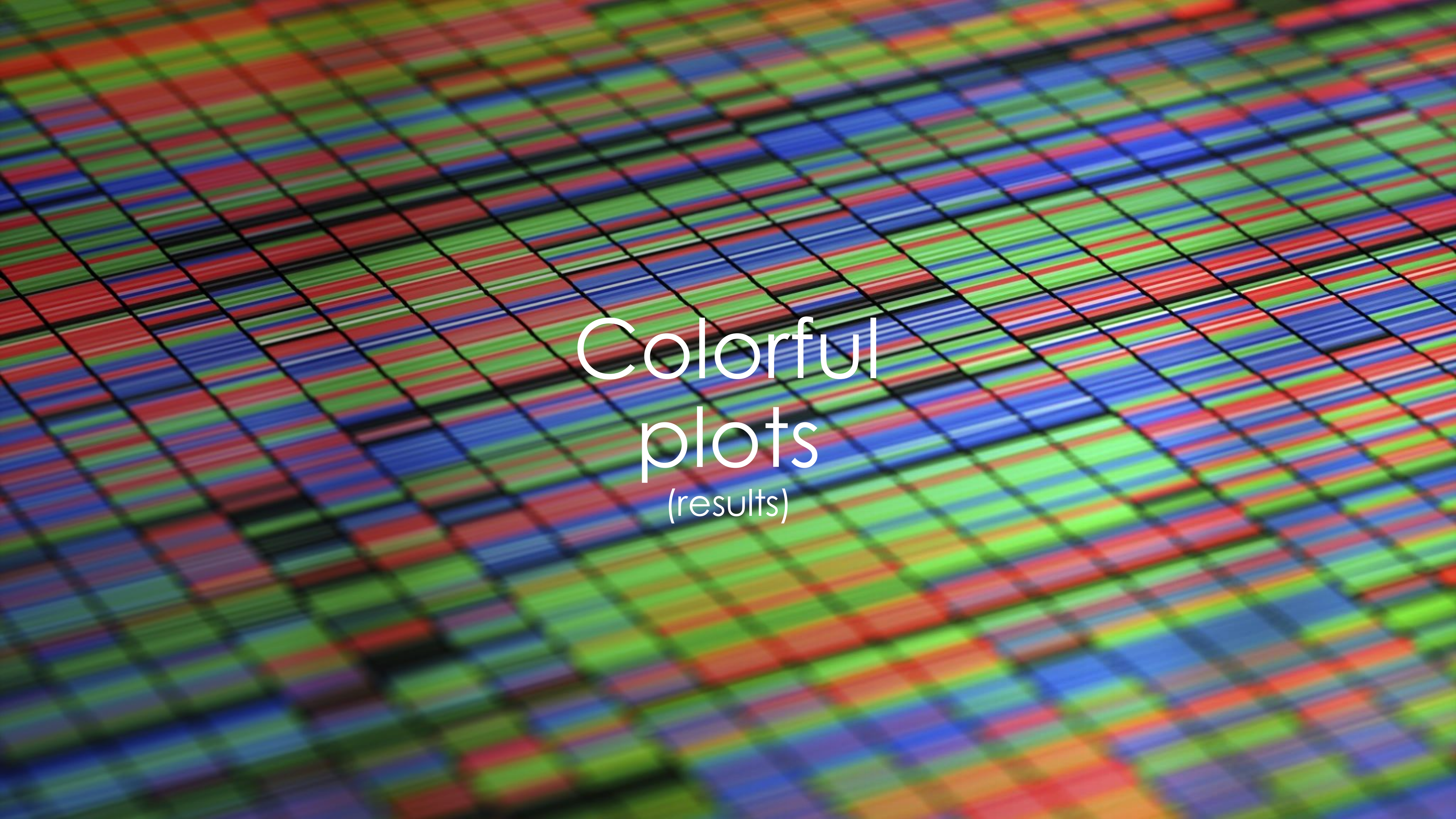
Simulation study

4-way design

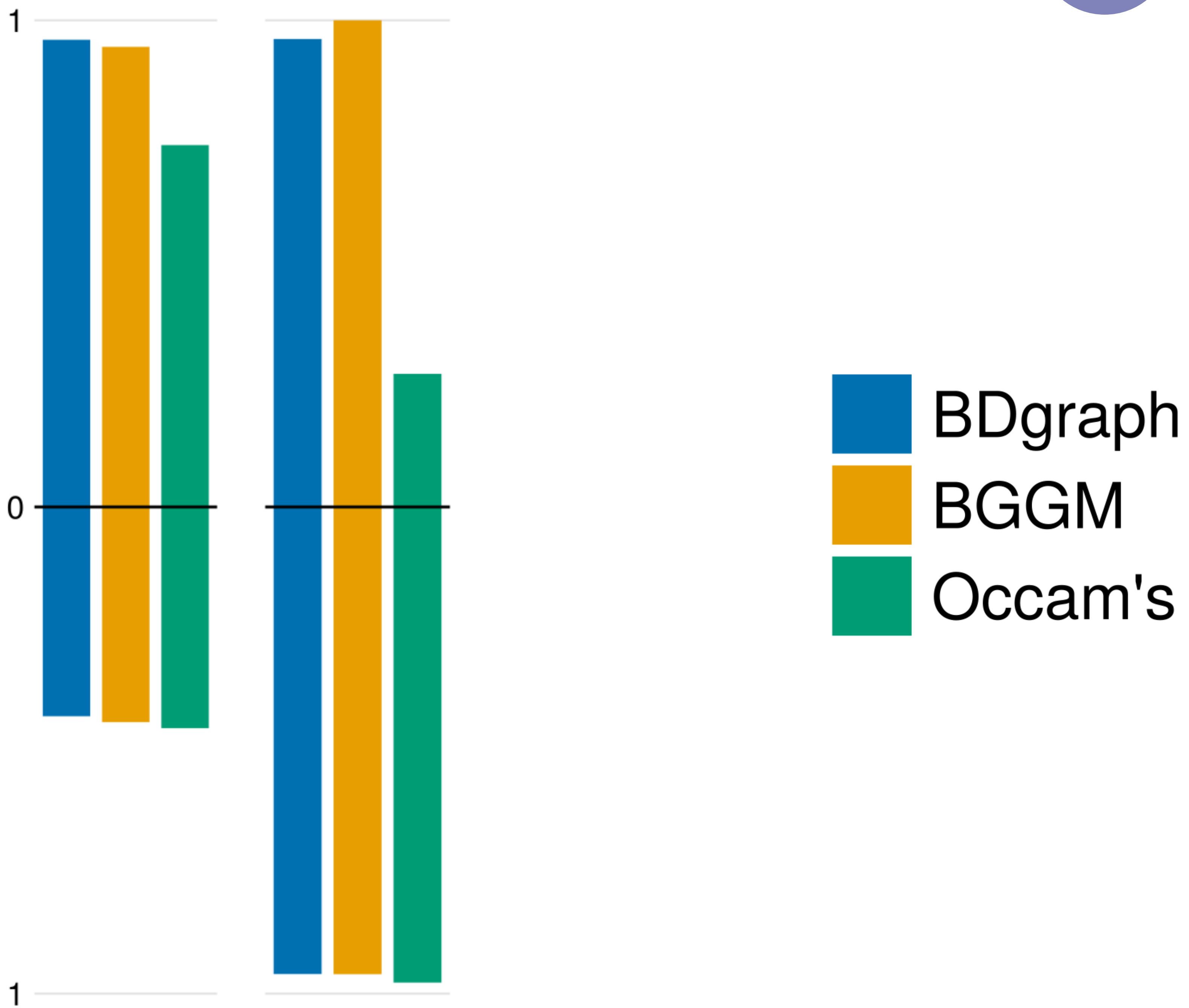
GGM

Model	Algorithm	Starting models	Constants
BGGM	Pairwise BF	-	-
BDgraph	(BD)MCMC	-	-
Ours	Occam's window 1	Saturated	Normal

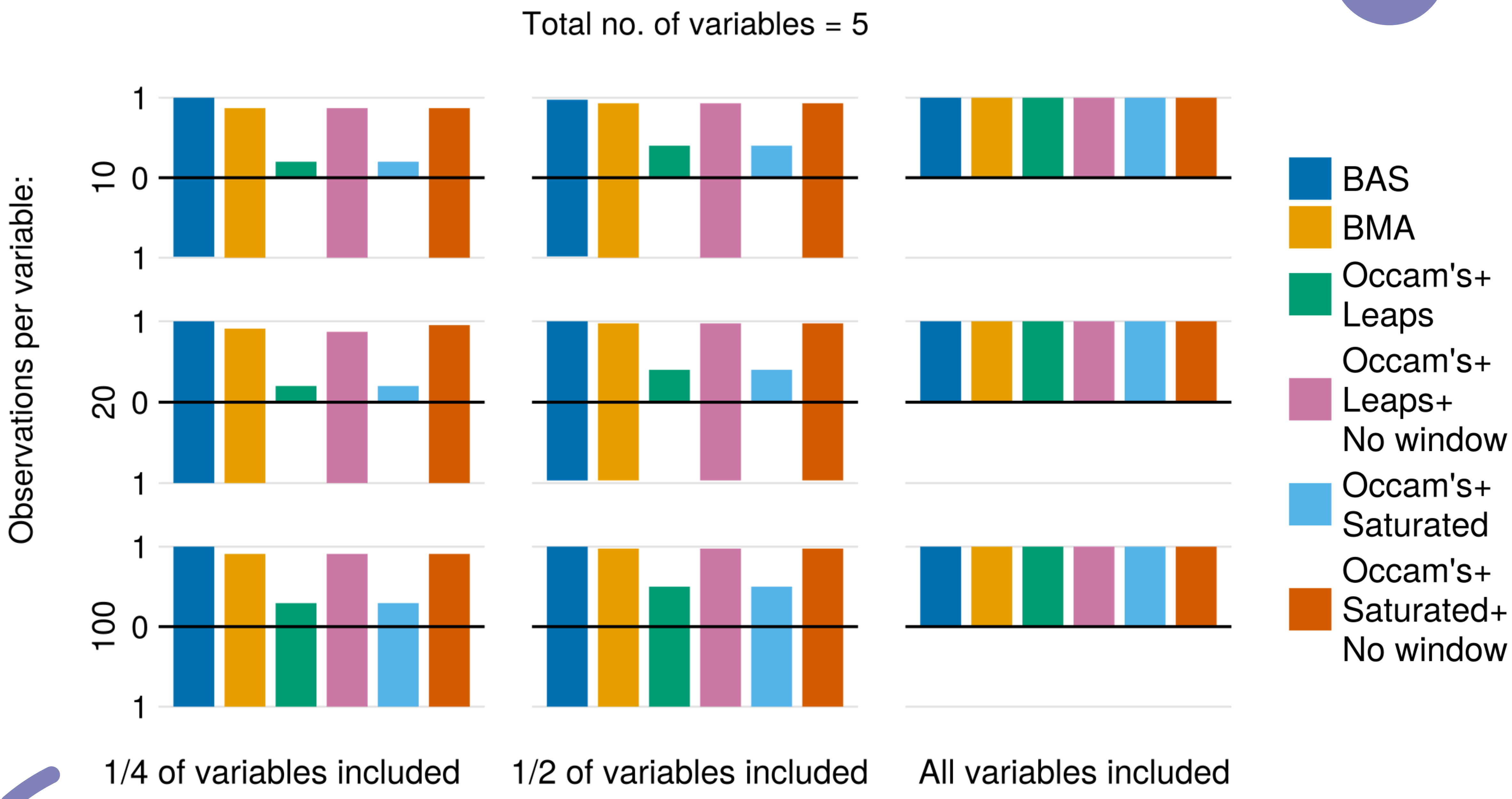
$ killall julia
$ killall julia
$ killall julia
$ killall julia
Temperatures
acpitz 25°C
ath10k_hwmon 56°C
coretemp_core0 99°C
coretemp_core1 97°C
Multiprocessing vs Multithreading
CPU1
Registers
Cache
CPU1
Registers
Cache
CPU1
Registers
Cache
Code Data Files
registers stack
Thread
Code Data Files
registers stack registers stack registers stack
Threads
Progress: 6%| | ETA: 14:34:37
Progress: 100%| | Time: 16:00:39
[Info: Completed in 57639.626967191s


Colorful
plots
(results)

Sensitivity
True **positive** rate
How good the algorithm is at identifying the true edges

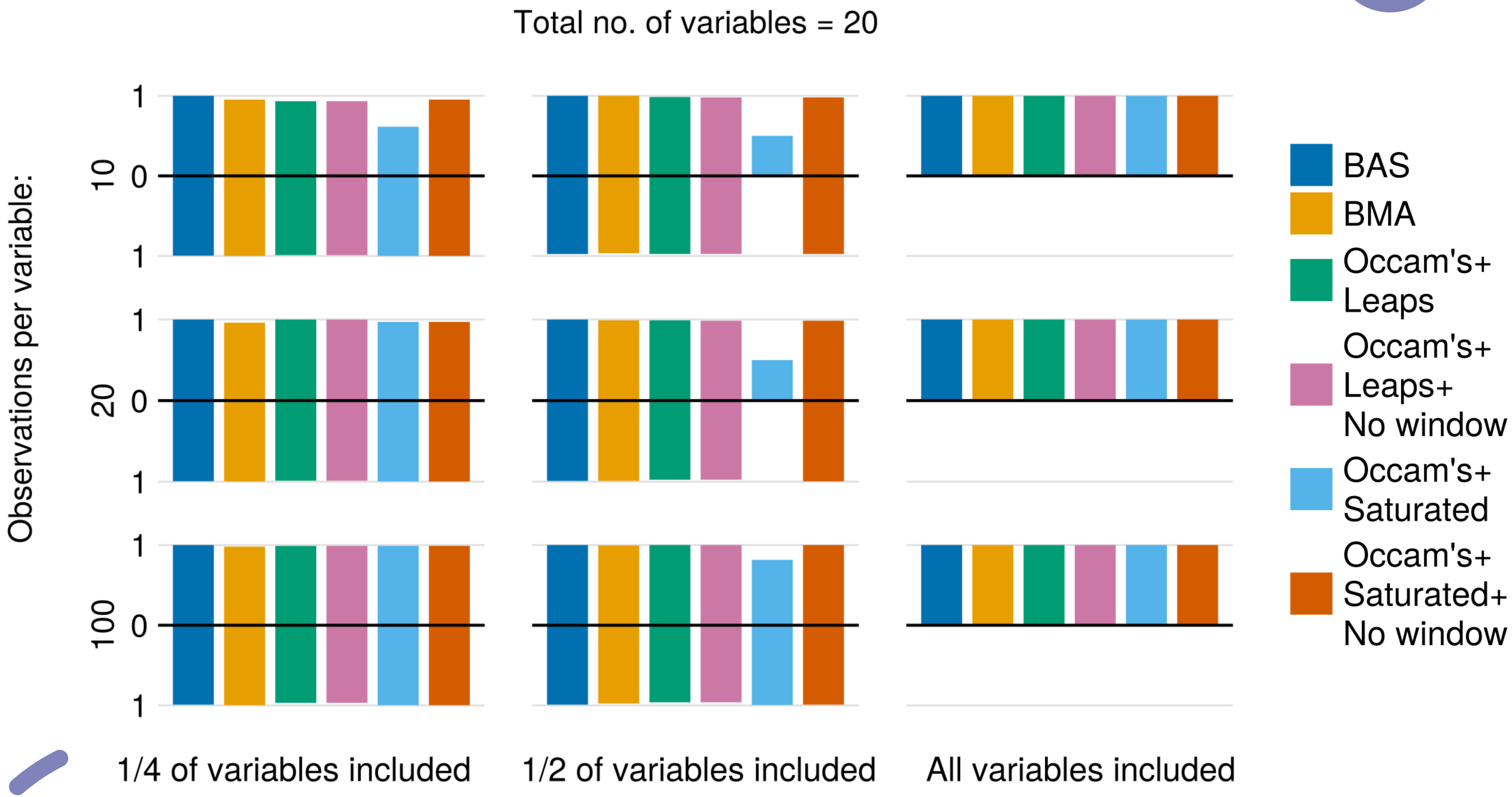


Specificity
True **negative** rate
How good the algorithm is at identifying the missing edges



Total no. of variables = 5
Observations per variable:
10
20
100
1
0
1
1/4 of variables included
1/2 of variables included
All variables included
BAS
BMA
Occam's+ Leaps
Occam's+ Leaps+ No window
Occam's+ Saturated
Occam's+ Saturated+ No window

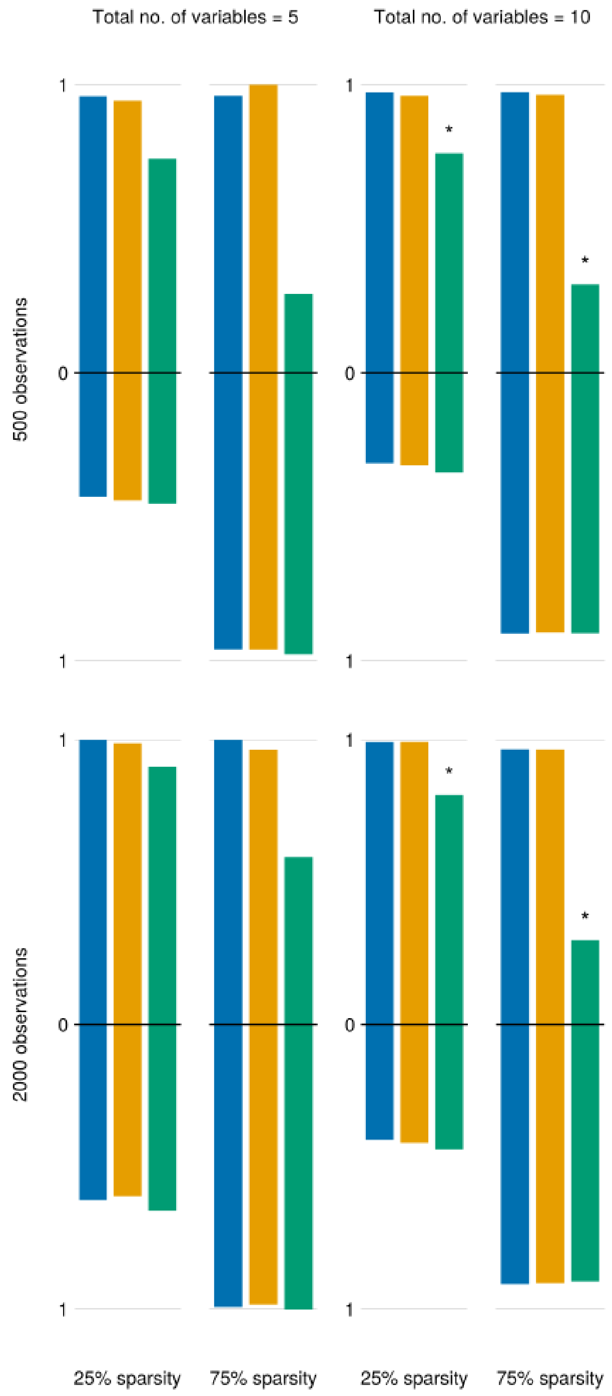

Total no. of variables = 10
Observations per variable:
10
20
100
1
0
1
1/4 of variables included
1/2 of variables included
All variables included
BAS
BMA
Occam's+ Leaps
Occam's+ Leaps+ No window
Occam's+ Saturated
Occam's+ Saturated+ No window



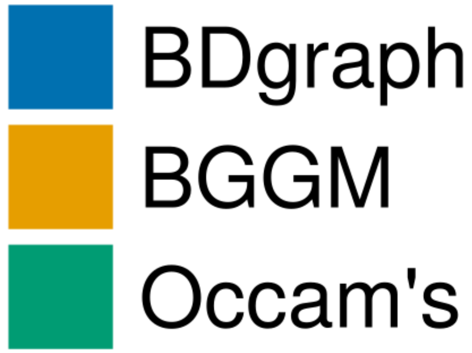
Total no. of variables = 20
Observations per variable:
10
20
100
1
0
1
1/4 of variables included
1/2 of variables included
All variables included
BAS
BMA
Occam's+ Leaps
Occam's+ Leaps+ No window
Occam's+ Saturated
Occam's+ Saturated+ No window

Results linear regression

- BMA & BAS just work great
- Occam's window Algorithm 1 performs worse
  - Especially worse when the algorithm is used as indented (with the window)
  - Tendency to over-include parameters
- Run-time
  - BAS took ~1.5 s
  - BMA a fraction of a second
  - Occam's a few seconds, less than 5



Total no. of variables = 5
Total no. of variables = 10
500 observations
2000 observations
1
0
1
*
25% sparsity
75% sparsity



BDgraph
BGGM
Occam's

Results GGM

- BGGM & BDgraph work better
- All struggle on low sparsity conditions & all tend to over-include parameters
- Less differences than in the linear regression case
- Run-time
  - BGGM runs in a fraction of a second
  - BDgraph a few seconds
  - Occam's +30 min, over 1h sometimes

Is Occam's window a potentially useful algorithm to explore the model space of GGMs?

- Does it produce OK results?
- Does it work in a reasonable timeframe?

Discussion

- BGGM and BDgraph work fine in the simple (continuous normal data) case
  - We know their performance is much worse in other cases
- All other algorithms use very optimized calculations, ours is a bare-bones implementation

Discussion

- How to improve speed?
  - Use some sort of sequential calculation for the marginal likelihood
  - Possibly a pseudo-likelihood like Pensar et al. (2017) or Mohammadi et al. (2017).
- How to improve the results?
  - Occam's window Algorithm 2 is the key for BMA's performance
  - Requires a *good* reduced set of candidate models as a proxy for the whole model space
  - Maybe Marsman et al. (2022), only exploring models with promising edges

Is Occam's window a potentially useful algorithm to explore the model space of GGMs?

Is it worth to invest more time pursuing a better implementation of Occam's window for GGMs?

- Eh... idk.
- Depends on your personal trade-off between the importance of the problem and the potential increase in performance vs other solutions.
- Not for me (:
- But maybe it is for Maarten.


Thank you.