

Bayesian Structure Learning in Undirected Gaussian Graphical Models: Literature Review with Empirical Comparison

Lucas Vogels, Reza Mohammadi, Marit Schoonhoven and Ş. İlker Birbil
Faculty of Economics and Business, University of Amsterdam

March 26, 2024

Abstract

Gaussian graphical models provide a powerful framework to reveal the conditional dependency structure between multivariate variables. The process of uncovering the conditional dependency network is known as structure learning. Bayesian methods can measure the uncertainty of conditional relationships and include prior information. However, frequentist methods are often preferred due to the computational burden of the Bayesian approach. Over the last decade, Bayesian methods have seen substantial improvements, with some now capable of generating accurate estimates of graphs up to a thousand variables in mere minutes. Despite these advancements, a comprehensive review or empirical comparison of all recent methods has not been conducted. This paper delves into a wide spectrum of Bayesian approaches used for structure learning and evaluates their efficacy through a simulation study. We also demonstrate how to apply Bayesian structure learning to a real-world data set and provide directions for future research. This study gives an exhaustive overview of this dynamic field for newcomers, practitioners, and experts.

Keywords: Markov chain Monte Carlo; Bayesian model selection; covariance selection; Markov random fields; Link prediction.

1 Introduction

One can depict conditional dependencies between a large number of variables using undirected graphical models. Using data to recover the structure of a graphical model is called *structure learning*. There are many applications of undirected graphical models. In biology, they are used to recover gene networks (Bhadra & Mallick 2013, Li et al. 2019a, Chandra et al. 2022); in neuroscience, graphical models can discover the connectivity of the brain (Hinne et al. 2014, Dyrba et al. 2020); in economics, they map the relationships between purchases of individual customers (Giudici & Castelo 2003); in finance, they discover how the credit risks of financial institutions are related (Wang 2015); and in psychology, they map the relationships between psychological variables (Epskamp et al. 2018, Waldorp & Marsman 2022).

Within the context of structure learning, Bayesian methods offer several advantages over frequentist approaches. First, Bayesian methods facilitate the integration of prior beliefs, which can potentially enhance the precision of learning the structure. For example, one can have an idea about the risk factors of a disease before seeing any patient data. Second, Bayesian methods estimate the entire probability distribution of parameters, whereas frequentist methods only provide their point estimates. This allows Bayesian methods to incorporate model uncertainty. Despite these advantages, frequentist methods in structure learning are more popular than Bayesian methods due to their computational efficiency and simplicity. However, in the last decade, new Bayesian methods have been proposed that are computationally efficient and accurate, even for large-scale graphs. Meanwhile, the development of several software packages has made Bayesian structure learning accessible and practical; see, for example, the R packages BDgraph (Mohammadi et al. 2022), ssgraph (Mohammadi 2022), BGGM (Williams & Mulder 2019) and baygel (Smith et al. 2023a).

Despite the recent developments in Bayesian structure learning, there is no comprehen-

sive review or empirical comparison of all methods. For newcomers to the field, this review provides a clear introduction. For practitioners, we discuss typical challenges when applying Bayesian methods to real data and show the benefits of Bayesian structure learning in practice. For experts, we provide an in-depth overview of Bayesian methods coupled with an extensive simulation study.

In this article, we consider general undirected graphs. Moreover, we assume Gaussian data, *i.e.*, we work with Gaussian graphical models (GGMs). GGMs are the most widely researched undirected graphical models due to their relative simplicity and their variety of applications. The focus of this review is on Bayesian methods in GGMs. However, we do include a short overview of the relevant frequentist methods as they form the starting point for some of their Bayesian counterparts. Moreover, the simulation study contains a frequentist approach to show how Bayesian algorithms perform relative to the better-known frequentist alternative. Our main focus is on the Bayesian methods of the last two decades since these contributed most to the progress in Bayesian structure learning. GGMs form the foundation of several sub-fields that we briefly touch upon too.

We continue in Section 2 with a general introduction to frequentist and Bayesian structure learning in GGMs. Section 3 provides a review of the recent literature, and Section 4 contains a simulation study comparing the performance of seven state-of-the-art Bayesian algorithms and one frequentist algorithm. In Section 5, we apply Bayesian algorithms to a gene data set and discuss typical challenges that arise in real data application of Bayesian structure learning. We end with a short overview of related sub-fields and give future perspectives and recommendations in Section 6.

2 Structure Learning

This section introduces structure learning. We start with the problem and notation in Subsection 2.1, briefly discuss frequentist approaches in Subsection 2.2, and end with an introduction to Bayesian structure learning in Subsection 2.3.

2.1 Problem and Notation

Let (X_1, X_2, \dots, X_p) be a vector of random variables, and X_{-ij} denote this vector without the variables X_i and X_j with $i, j = 1, 2, \dots, p$ and $i \neq j$. We say that the variables X_i and X_j are conditionally independent, when

$$P(X_i|X_j, X_{-ij}) = P(X_i|X_{-ij}).$$

In other words, when the values of the variables X_{-ij} are given, knowing the value of X_j does not change the probability distribution of X_i . When two variables X_i and X_j are conditionally (in)dependent, they are also referred to as partially (un)correlated.

We aim to estimate the conditional dependencies between any pair of variables in an undirected graph $G = (V, E)$. Here, every node in the node set $V = \{1, \dots, p\}$ corresponds to a random variable, and the edge set E is defined by $E = \{(i, j) \in V \times V : i < j \text{ and } X_j \text{ and } X_i \text{ are conditionally dependent}\}$. We slightly abuse notation by saying $(i, j) \in G$ when we mean $(i, j) \in E$. We denote the true graph with $G^* = (V, E^*)$ and any other graph with G . In general, we do not know the true graph. We only observe a sample of n observations of the variables X_1, \dots, X_p . Let x_{li} denote the value of the random variable X_i for observation l with $i = 1, \dots, p$ and $l = 1, \dots, n$. We denote all samples of the i th variable as $\mathbf{x}_i = (x_{1i}, \dots, x_{ni})^T$. Lastly, $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_p)$ denotes the $n \times p$ matrix containing all observations of all variables. We assume that all observations are drawn from a multivariate normal distribution with mean $\mathbf{0}$ and an unknown $p \times p$ covariance matrix Σ^* . That is, $(x_{l1}, x_{l2}, \dots, x_{lp}) \sim \mathcal{N}(\mathbf{0}, \Sigma^*)$ for all $l = 1, \dots, n$.

The problem of finding the graph G^* was introduced by Dempster (1972) and coined as covariance selection. It is also referred to as structure learning or graphical model determination. The goal of structure learning is to use the sample \mathbf{X} to infer the properties of G^* . Let $\mathbf{K}^* = \Sigma^{*-1}$ with elements k_{ij}^* , $i, j = 1, 2, \dots, p$, be the true and unknown precision matrix or concentration matrix. The precision matrix turns out to be a helpful tool in unraveling the structure of the true graph, since

$$\rho_{ij} = -\frac{k_{ij}^*}{\sqrt{k_{ii}^* k_{jj}^*}}, \quad (1)$$

where ρ_{ij} is the partial correlation between variables X_i and X_j . For a proof, see Lauritzen (1996). Due to (1), we obtain the following relationship between \mathbf{K}^* and G^* :

$$k_{ij}^* = 0 \iff (i, j) \notin G^*. \quad (2)$$

In what follows \mathbf{K}^* and Σ^* denote the true and unknown precision and covariance matrix, respectively. Likewise, \mathbf{K} and Σ denote any other precision and covariance matrix.

2.2 Frequentist Structure Learning

This section gives a short overview of frequentist methods and their connections to their Bayesian counterparts. Most frequentist methods in structure learning for GGMs obtain an estimate $\hat{G} = (V, \hat{E})$ of the true graph G^* by either solving multiple regressions or employing the penalized likelihood (Liang & Jia 2023, Chapter 2.2). The regression-based approaches start with the observation that we can express each variable X_i in terms of the other variables X_{-i} as follows:

$$X_i = \sum_{j \neq i} \beta_{ij} X_j + \epsilon_i, \quad i = 1, 2, \dots, p, \quad (3)$$

where ϵ_i is uncorrelated with X_{-i} and $\beta_{ij} = \frac{k_{ij}^*}{k_{ii}^*}$ (Lauritzen 1996). Finding the regression coefficients β_{ij} in Equation (3) is thus enough for structure learning, since $\beta_{ij} = 0 \iff k_{ij}^* = 0 \iff (i, j) \notin G^*$. In their neighbourhood selection method Meinshausen &

Bühlmann (2006) use the Lasso regression by Tibshirani (1996) to obtain estimates for all regression coefficients by solving a sequence of optimization problems given by

$$\min_{\beta_{ij}, i \neq j} \|\mathbf{X}_i - \sum_{i \neq j} \beta_{ij} \mathbf{X}_j\|^2 + \lambda \sum_{i \neq j} |\beta_{ij}|, \quad i = 1, \dots, p, \quad (4)$$

where the first term of the objective function is the residual term and the second term (with the tuning parameter $\lambda \geq 0$) is the ℓ_1 -norm regularization term enabling sparse solutions. Next, they include an edge (i, j) , if $\beta_{ij} \neq 0$. Similar approaches have been proposed by Yuan (2010), Sun & Zhang (2012), and Liu & Wang (2017), each using different methods to find the coefficients β_{ij} in (3). The Bayesian regression approaches discussed in Section 3.3 also use Equation (3), but use Bayesian methods to estimate the regression coefficients.

The penalized likelihood approach starts with the observation that the log-likelihood can be written as

$$\log P(\mathbf{X}|\mathbf{K}) = C + \frac{n}{2} \log |\mathbf{K}| + \frac{n}{2} \text{tr}(\mathbf{S}\mathbf{K}),$$

where C is a constant, $\mathbf{S} = \mathbf{X}^T \mathbf{X}$ is the sample covariance matrix, and $\text{tr}(\mathbf{A})$ and $|\mathbf{A}|$ are the trace and determinant of a square matrix \mathbf{A} , respectively. Penalized likelihood methods aim to find the maximum likelihood estimator (MLE), *i.e.*, the positive definite precision matrix \mathbf{K} that maximizes the log-likelihood. However, this MLE is generally not sparse. Penalized likelihood methods therefore add a penalty function $p(\mathbf{K})$ as a regularization term and minimize the negative penalized log-likelihood:

$$\min_{\mathbf{K} \in \mathcal{K}} -\frac{n}{2} \log |\mathbf{K}| - \frac{n}{2} \text{tr}(\mathbf{S}\mathbf{K}) - \lambda p(\mathbf{K}), \quad (5)$$

where $\lambda \geq 0$ is the regularization parameter and \mathcal{K} is the set of all $p \times p$ positive definite matrices. In the case of ℓ_1 -norm regularization, the penalty term is given by $p(\mathbf{K}) = \sum_{i,j, i \neq j} |k_{ij}|$ and the optimization problem (5) becomes a convex program that Yuan & Lin (2007) solve using the interior point algorithm (Vandenbergh et al. 1998). More efficient, however, is the block coordinate descent approach introduced by Banerjee et al.

(2008) and improved by Friedman et al. (2008), who coined it the graphical Lasso (glasso) algorithm. To this day, the glasso algorithm remains the most popular in structure learning and is therefore included in the simulation study in Section 4. Performance bounds of the graphical lasso estimator have been given by Rothman et al. (2008), Lam & Fan (2009), and Ravikumar et al. (2011). Several other penalty functions have been proposed, see, for example, Fan et al. (2009). The tuning parameter λ directly impacts the sparsity of the estimate $\hat{\mathbf{K}}$. Several strategies have been proposed for the choice of λ , including the extended Bayesian information criterion (Foygel & Drton 2010), the rotation information criterion (Zhao et al. 2012), and k -fold cross validation (Bien & Tibshirani 2011). The Bayesian methods discussed in Section 3.1.3 have a direct connection to these frequentists penalized likelihood approaches: their Bayesian estimate of the precision matrix is equal to the one obtained by solving the minimization problem given by (5).

2.3 Bayesian Structure Learning

In contrast to frequentist methods that aim to provide an estimate of the true graph, Bayesian methods strive to uncover the probability distribution of the true graph given the sample \mathbf{X} . This is called the posterior distribution and is denoted with $P(G|\mathbf{X})$. This is the probability that, given the data \mathbf{X} , the true graph is equal to some graph G . It is given by Bayes' rule:

$$P(G|\mathbf{X}) = \frac{P(\mathbf{X}|G)P(G)}{P(\mathbf{X})}. \quad (6)$$

Here, $P(\mathbf{X}|G)$ is the likelihood and denotes the probability of observing the data \mathbf{X} when the true graph is equal to G . For each G , the prior $P(G)$ describes the given belief that G is the true graph. Lastly, $P(\mathbf{X})$ is the normalizing constant given by $P(\mathbf{X}) = \sum_{G \in \mathcal{G}} P(\mathbf{X}|G)P(G)$, where \mathcal{G} denotes the set of all graphs with p nodes. To evaluate the likelihood $P(\mathbf{X}|G)$ in

Equation (6), one could use

$$P(\mathbf{X}|G) = \int_{\mathbf{K} \in \mathcal{K}} P(\mathbf{X}|G, \mathbf{K})P(\mathbf{K}|G)d\mathbf{K}. \quad (7)$$

Often this integral is hard to evaluate. Other Bayesian methods therefore focus on the joint posterior distribution of the true graph and precision matrix given by

$$P(G, \mathbf{K}|\mathbf{X}) = \frac{P(\mathbf{X}|\mathbf{K}, G)P(\mathbf{K}|G)P(G)}{P(\mathbf{X})}. \quad (8)$$

Here, $P(\mathbf{X}|\mathbf{K}, G)$ is the joint likelihood and denotes the probability density of n samples from a multivariate normal distribution with mean $\mathbf{0}$ and covariance matrix \mathbf{K}^{-1} . It is given by

$$P(\mathbf{X}|\mathbf{K}, G) = (2\pi)^{-pn/2} |\mathbf{K}|^{n/2} \exp \left\{ \frac{-1}{2} \text{tr}(\mathbf{K}\mathbf{S}) \right\}. \quad (9)$$

$P(\mathbf{K}|G)$ is the prior distribution on the precision matrix. For each G and \mathbf{K} , it describes the belief that when the true graph is equal to G , the precision matrix is equal to \mathbf{K} .

3 Review of Methodology

The history of Bayesian structure learning is one of progress. From inefficient approaches restricted to small and decomposable graphs in the 1990s and 2000s to the efficient, accurate, and generally applicable methods of today. This chapter presents the story of this progress. We briefly cover the most relevant approaches before 2010 before we give a detailed outline of all recent methods.

Bayesian approaches in structure learning aim to uncover the posterior distribution $P(G|\mathbf{X})$ of the graph. For that, one could either evaluate the posterior (6) or the joint posterior (8). For both, the prior distribution $P(\mathbf{K}|G)$ is required. Early Bayesian structure learning focuses on proposing this prior. Ideally, this prior is conjugate, *i.e.*, the prior and posterior follow the same distribution family. Dawid & Lauritzen (1993) define such prior and call it the Hyper Inverse-Wishart (HIW) prior. They show that, apart from

conjugacy, this prior also satisfies the so-called weak and strong hyper Markov property. Using the HIW prior, Giudici & Green (1999) and Carvalho et al. (2007) design algorithms for Bayesian structure learning that output the posterior for small graphs. The HIW prior, however, is only defined on decomposable graphs, which form a small subset of the whole graph space. Roverato (2002) solves this by extending the HIW to general graphs, a distribution that Atay-Kayis & Massam (2005) later coin the G-Wishart distribution. Its density is given by

$$P(\mathbf{K}|G) = \frac{1}{I_G(b, \mathbf{D})} |\mathbf{K}|^{\frac{b}{2}-1} \exp\left\{-\frac{1}{2}\text{tr}(\mathbf{K}\mathbf{D})\right\} I(\mathbf{K} \in \mathcal{P}_G), \quad (10)$$

where $\mathcal{P}_G \subset \mathcal{K}$ denotes the set of symmetric positive definite matrices \mathbf{K} that have $k_{ij} = 0$ when $(i, j) \notin G$. Here, $I(\mathbf{K} \in \mathcal{P}_G)$ is an indicator function that is equal to one, if $\mathbf{K} \in \mathcal{P}_G$, and zero otherwise. The symmetric positive definite matrix \mathbf{D} and the scalar $b > 2$ are called the scale and shape parameters of the G-Wishart distribution. They are mostly set to $b = 3$ and $\mathbf{D} = \mathbf{I}$. The normalizing constant

$$I_G(b, \mathbf{D}) = \int_{\mathbf{K} \in \mathcal{P}_G} |\mathbf{K}|^{\frac{b-2}{2}} \exp\left\{-\frac{1}{2}\text{tr}(\mathbf{K}\mathbf{D})\right\} d\mathbf{K} \quad (11)$$

ensures that $\int_{\mathbf{K} \in \mathcal{P}_G} P(\mathbf{K}|G) d\mathbf{K} = 1$. The normalizing constant is troublesome to evaluate, but Roverato (2002), Atay-Kayis & Massam (2005), and Lenkoski & Dobra (2011) propose approximations. Using the G-Wishart prior, the marginal likelihood (7) becomes

$$P(\mathbf{X}|G) = (2\pi)^{-np/2} \frac{I_G(b + n, \mathbf{D} + \mathbf{X}^T \mathbf{X})}{I_G(b, \mathbf{D})}. \quad (12)$$

Using the approximations for the normalizing constants in this ratio, one can approximate $P(\mathbf{X}|G)$. Jones et al. (2005) and Lenkoski & Dobra (2011) both design stochastic search algorithms that in each iteration favour a move to a graph with a high posterior probability $P(G|\mathbf{X}) \propto P(\mathbf{X}|G)P(G)$. However, the approximation of the normalizing constants in (12) is unstable and its accuracy can be hard to access. Moreover, the approximations are computationally expensive, limiting structure learning to small graphs with less than 20 nodes.

That is why others focus on inference on the joint posterior (8). For this and for inference on the precision matrix in general, it is required to sample from the G-Wishart distribution (10), which is challenging due to the normalizing constant (11). This led to a string of articles on efficient sampling from the G-Wishart distribution, see Wang & Li (2012) for a review of those. However, sampling from the G-Wishart distribution remained computationally inefficient and unreliable.

The period after 2010 saw several discoveries accelerating Bayesian structure learning. They include solutions to the challenges discussed before: computationally efficient approximation of the normalizing constant (11) and computationally efficient G-Wishart sampling. They also contain entirely new perspectives and approaches, enabling the creation of new methods that push the boundaries of accuracy and computational efficiency. These approaches form the topic of this chapter. We group them into four different categories: Markov chain Monte Carlo (MCMC), Expectation Conditional Maximization (ECM), regression, and hypothesis approaches. The most common Bayesian methods in each category are listed in Table 1. Next, we provide an overview of the workings and theoretical properties of these methods.

Category	Method name	Reference	Included in sim.
MCMC on $\mathcal{G} \times \mathcal{K}$	RJ - O	Dobra et al. (2011)	
	RJ - WL	Wang & Li (2012)	
	RJ - CL	Cheng & Lenkoski (2012)	
	RJ - D	Lenkoski (2013)	
	RJ - DCBF	Hinne et al. (2014)	
	RJ - A	Mohammadi et al. (2022)	
	RJ - WWA	Van den Boom et al. (2022a)	✓
	RJ - UB	Van den Boom et al. (2022b)	
	BD - O	Mohammadi & Wit (2015)	
	BD - A	Mohammadi et al. (2023a)	✓
	SS	Wang (2015)	✓
MCMC on \mathcal{G}	G - Stingo	Stingo & Marchetti (2014)	
	G - MPLRJ	Mohammadi et al. (2023b)*	
	G - MPLBD	Mohammadi et al. (2023b)*	✓
MCMC on \mathcal{K}	K - BGLasso1	Wang (2012)	
	K - BGLasso2	Khondker et al. (2013)	
	K - RIW	Kundu et al. (2018)	
	K - Horseshoe	Li et al. (2019a)	✓
	K - Horseshoe-like	Sagar et al. (2024)	
	K - LRD	Chandra et al. (2022)*	
	K - BAGR	Smith et al. (2023b)*	
ECM	ECM - EMGS	Li & McCormick (2019)	
	ECM - BAGUS	Gan et al. (2019)	✓
	ECM - HL	Sagar et al. (2024)	
Regression	R - BLNRE	Li & Zhang (2017)	
	R - P	Williams et al. (2018)	
Hypothesis	H - LR	Leday & Richardson (2018)	
	H - BGGM	Williams & Mulder (2020)	✓

Table 1: *Overview of all the state-of-the-art methods for Bayesian structure learning in Gaussian Graphical Models. They are listed per category and chronologically. The last column indicates whether the method is included in the simulation study in Section 4. Papers marked with * have not yet been published.*

3.1 MCMC-based Approaches

This section gives a short introduction to Markov chain Monte Carlo (MCMC) sampling before describing the main MCMC-based approaches in Bayesian structure learning.

Most methods in Bayesian structure learning use MCMC sampling. MCMC is popular because of the variety of inferences it provides. Specifically, when using MCMC sampling, one can approximate the expected value of $h(G^*)$ for any information function $h : \mathcal{G} \rightarrow \mathbb{R}$. For example, one can choose $h(G)$ equal to 1 when an edge e is in G , and 0 otherwise. This will give the posterior edge inclusion probabilities

$$p_e := P(e \in G^* | \mathbf{X}) = E((h(G^*) | \mathbf{X}). \quad (13)$$

Similarly, with other information functions h , one can evaluate different posterior probabilities. For example, the probability that the true graph will have more than 10 edges, the probability that the true graph is connected, the probability that the true graph contains a cycle, and so on. However, this flexibility comes at a cost: MCMC algorithms are computationally demanding because they iteratively explore large parameter spaces.

Let $P(\mathbf{X} | \boldsymbol{\theta})$ be the likelihood conditional on some parameter vector $\boldsymbol{\theta} \in \mathcal{T}$. Let $\boldsymbol{\theta}^*$ be the true and unknown parameter vector. We want to perform inference on the posterior $P(\boldsymbol{\theta} | \mathbf{X})$. We are interested in obtaining $E(h(\boldsymbol{\theta}^*) | \mathbf{X}) = \int_{\boldsymbol{\theta}} h(\boldsymbol{\theta}) P(\boldsymbol{\theta} | \mathbf{X}) d\boldsymbol{\theta}$ for some information function $h : \mathcal{T} \rightarrow \mathbb{R}$. MCMC algorithms start at an initial realization of the parameter vector $\boldsymbol{\theta}^{(0)}$ and then jump to $\boldsymbol{\theta}^{(1)}$, $\boldsymbol{\theta}^{(2)}$, all the way to $\boldsymbol{\theta}^{(S)}$ with $S \in \mathbb{N}$. We call the resulting sequence $(\boldsymbol{\theta}^{(0)}, \dots, \boldsymbol{\theta}^{(S)})$ a Markov chain. For every $s = 1, \dots, S$, $\boldsymbol{\theta}^{(s)}$ is called a state of the Markov chain. All states are elements of the state space \mathcal{T} . A Markov chain is defined by its transition kernel $P(x, A)$. It denotes the probability that the next state will be part of the set $A \subset \mathcal{T}$, given that the current state is $x \in \mathcal{T}$. By carefully selecting the transition kernel, one can ensure that the chain converges to the so-called stationary distribution. There are three conditions sufficient for this convergence (Tierney 1994, Theorem 1), the

most crucial being the balance condition. This condition states that for any set $A \subset \mathcal{T}$, the rate at which the chain enters A needs to be the same as the rate at which it leaves A . The balance condition is often replaced by a stricter condition called the detailed balance condition, see Equation 1 in Green (1995). Now, if the balance condition holds and if the stationary distribution is equal to the posterior distribution, the distribution of our Markov Chain will get arbitrarily close to the posterior as we increase the chain length S . This holds no matter the starting position $x = \boldsymbol{\theta}^{(0)}$ of our Markov Chain. We can then get arbitrarily close to our desired expected value using

$$E(h(\boldsymbol{\theta}^*)|\mathbf{X}) = \lim_{S \rightarrow \infty} \frac{1}{S} \sum_{s=1}^S h(\boldsymbol{\theta}^{(s)}) \approx \frac{1}{S} \sum_{s=1}^S h(\boldsymbol{\theta}^{(s)}). \quad (14)$$

3.1.1 MCMC-based approaches on the joint space

Most of the MCMC approaches in Bayesian structure learning create a Markov Chain $((G^{(0)}, \mathbf{K}^{(0)}), \dots, (G^{(S)}, \mathbf{K}^{(S)}))$ over the joint space of graphs and precision matrices that converges to the joint posterior (8). Using Equation (14) these methods can perform inference on almost all characteristics of both the graph and the precision matrix. We call these methods joint MCMC methods. They come in three types: reversible jump (RJ), birth-death (BD), and spike-and-slab (SS). All joint MCMC methods are listed in Table 2.

RJ methods are the first MCMC-based methods in Bayesian structure learning. They use the G -Wishart prior given by (10). RJ methods are a variant of Metropolis-Hastings methods that were introduced by Green (1995). Algorithm 1 represents the pseudo-code for this class of methods.

Algorithm 1: Reversible Jump (RJ) algorithm

Input: Data \mathbf{X} and an initial graph $G^{(0)}$.

for *state* $s = 1, \dots, S$ **do**

 Set $G = G^{(s-1)}$;

 Propose a new graph G' by adding or deleting one edge from G ;

 Accept the proposal with acceptance probability $\alpha(G', G)$;

 Set $G^{(s)} = G'$ if accepted, set $G^{(s)} = G$ if rejected;

 Sample $\mathbf{K}^{(s)}$ from the $G^{(s)}$ -Wishart distribution;

Output: Samples $(G^{(0)}, \mathbf{K}^{(0)}), \dots, (G^{(S)}, \mathbf{K}^{(S)})$ from the joint posterior (8)

One can define the acceptance probabilities $\alpha(G', G)$ such that the balance condition holds and the resulting MCMC chain converges to the joint posterior. The first three RJ methods to use this strategy are the RJ-Original (RJ-O) method by Dobra et al. (2011), the RJ-Wang Li (RJ-WL) method by Wang & Li (2012), and the RJ-Cheng Lenkoski (RJ-CL) method by Cheng & Lenkoski (2012). These methods form the foundation for all later reversible jump methods. Their approach to sample from the G-Wishart distribution, however, is not scalable to bigger graphs and lacks a theoretical guarantee of convergence. To address these issues, Lenkoski (2013) introduces an efficient exact sampler from the G-Wishart distribution. The resulting RJ-Double (RJ-D) method thereby becomes the first joint method to achieve detailed balance, and hence, theoretical convergence. Hinne et al. (2014) combine several existing techniques to improve the overall efficiency in the RJ-Double Continuous Bayes Factor (RJ-DCBF) method. Despite these improvements, RJ methods are still too computationally demanding for graphs with more than 50 variables. One of the reasons is that the acceptance probabilities contain a ratio of normalizing constants that is time-consuming to evaluate. Mohammadi et al. (2023a) tackles this by developing a closed-form approximation of the ratio used in the RJ - Approximation (RJ-A) method by Mohammadi & Wit (2019). Due to the approximation, the RJ-A method loses

theoretical convergence but gains computational efficiency. The RJ-Weighted Proposal (RJ-WWA) method by Van den Boom et al. (2022a) combines the best of both worlds: a method that has both theoretical convergence and computational efficiency. Their improved strategy starts with two observations. First, at each iteration, all RJ algorithms propose a new graph that adds or removes an edge at random. This results in lots of graphs with low posterior probabilities that are proposed and subsequently rejected, thereby wasting computing power. The RJ-WWA algorithm is more likely to propose graphs with higher acceptance probabilities. The second observation is that the expensive calculation of the exact acceptance probability $\alpha(G', G)$ wastes computing power, especially if the proposed graph G' gets rejected. Therefore, they introduce a pre-acceptance step, in which the acceptance probabilities are approximated using the computationally efficient approximation of Mohammadi et al. (2023a). Only when pre-accepted, the exact acceptance probabilities are calculated. This so-called delayed acceptance requires the expensive calculation of exact acceptance probabilities only for G' that are likely to be accepted, hence minimizing the lost computational effort. The most recent reversible jump method is the RJ-Unbiased (RJ-UB) method by Van den Boom et al. (2022b). The RJ-UB method introduces unbiased estimation to Bayesian structure learning by applying a technique of Glynn & Rhee (2014). It requires the construction of two Markov Chains $(G^{(0)}, \dots, G^{(S)})$ and $(\bar{G}^{(0)}, \dots, \bar{G}^{(S)})$ that meet at some time $\tau < \infty$, that is, $G^{(s)} = \bar{G}^{(s)}$ for all $s \geq \tau$. Now, the quantity

$$\hat{h}_q = h(G^{(q)}) + \sum_{s=q+1}^{\tau} [h(G^{(s)}) - h(\bar{G}^{(s)})] \quad (15)$$

is an unbiased estimator of $E(h(G^*)|\mathbf{X})$. The main challenge in obtaining the unbiased estimator \hat{h}_q is the construction of the two Markov chains that meet at a time τ and for which the balance conditions hold. To do so, Van den Boom et al. (2022b) use a combination of a sequential Monte Carlo (SMC) method and a particle-independent Metropolis-Hastings method. Van den Boom et al. (2022b) are not the first to use SMC techniques in Bayesian

structure learning. Tan et al. (2017) also use an SMC sampler. Their method, however, is computationally cumbersome and comes with no theoretical guarantees.

The informed proposal and delayed acceptance mechanisms of the RJ-WWA method are one way of dealing with the low acceptance probabilities of the reversible jump approach. Mohammadi & Wit (2015) develop a different strategy to tackle this issue and create a new class of joint MCMC methods: Birth-Death (BD) methods. Their BD-Original (BD-O) method abolishes acceptance probabilities altogether while maintaining theoretical convergence. The BD-O method designs a continuous time (CT) Markov chain. This chain spends a continuous time $W(s)$ in each state s before moving to the next state. The next state is determined by adding or removing an edge based on so-called birth-death rates that need to be recalculated at every iteration. Its pseudo-code is shown in Algorithm 2.

Algorithm 2: Birth-death algorithm

Input: Data \mathbf{X} and an initial graph $G^{(0)}$.

for *state* $s = 1, \dots, S$ **do**

Calculate the birth-death rate r_e for every edge $e \in G^{(s-1)}$;

Calculate the waiting time $W(s-1) = \frac{1}{\sum_e r_e}$;

Select one edge e with a probability proportional to its birth-death rate;

Obtain $G^{(s)}$ by flipping edge e in $G^{(s-1)}$;

Sample $\mathbf{K}^{(s)}$ from the $G^{(s)}$ -Wishart distribution;

Output: Samples from the joint posterior (8)

By defining the birth-death rates correctly one can prove that the BD-O method meets the balance condition and therefore converges to the joint posterior. One can obtain the desired expected value using

$$E(h(G^*, \mathbf{K}^*)|\mathbf{X}) \approx \frac{1}{W} \sum_{s=1}^S W(s) h(G^{(s)}, \mathbf{K}^{(s)}), \quad (16)$$

where W is the sum of the waiting times of all states. Per iteration, the BD-O algorithm is computationally costly because it calculates the birth and death rates for all edges. However, its MCMC chain needs less iterations to converge because it moves to a new graph

every iteration. Moreover, the calculation of the birth and death rates can be performed in parallel. Just like the acceptance ratio of the RJ methods, the birth and death rates of the BD-O method contain a ratio of normalizing constants. Using the approximation of this ratio by Mohammadi et al. (2023a), the BD-Approximation (BD-A) method achieves a significant reduction in computational cost. Due to this approximation the balance condition no longer holds and there is no theoretical convergence.

Despite all the mentioned improvements, RJ and BD methods still suffer from a high computational burden. This has two reasons: first, at every iteration, a new precision matrix needs to be sampled from the G -Wishart distribution, which remains, despite the direct sampler of Lenkoski (2013), computationally costly. Second, at every MCMC iteration, the graph changes by at most one edge. This leads to poor mixing and convergence of the MCMC chain, especially for large-scale graphs. Both problems were overcome by Wang (2015) with his Spike-and-Slab (SS) method. First, Wang (2015) defined a new prior on the precision matrix, called the spike-and-slab prior. Remember that the G -Wishart prior (10) sets elements k_{ij} to zero when the corresponding element $(i, j) \notin G$. The spike-and-slab (SS) prior samples these elements from a normal density with a variance v_0 . The resulting elements are therefore not exactly zero, but, by choosing v_0 to be small, they are likely to be close to zero. The other elements are sampled from a normal density with a higher variance $v_1 > v_0$. The elements of the precision matrix are still dependent, since the resulting matrix \mathbf{K} needs to be positive definite. Moreover, like the G -Wishart prior, the SS prior contains an intractable normalizing constant. The SS prior allows for the design of an efficient block Gibbs sampler that updates the precision matrix column by column. Instead of adding or deleting a single edge (like the RJ or BD algorithms), the SS algorithm samples an entirely new graph at each iteration. The Markov chain’s mixing is therefore greatly improved compared to the RJ and BD methods. The pseudo-code of the SS algorithm is shown in Algorithm 3

Algorithm 3: Spike-and-slab algorithm

Input: Data \mathbf{X} and an initial graph $G^{(0)}$.

for *state* $s = 1, \dots, S$ **do**

 Sample a new precision matrix $\mathbf{K}^{(s)}$ using block Gibbs sampling;

 Calculate the probabilities l_e for every edge e ;

 Obtain $G^{(s)}$ by including every edge e with probability l_e ;

Output: Samples from the joint posterior (8).

The probabilities l_e in Algorithm 3 are calculated using an approximation. The balance condition therefore does not hold and the SS algorithm is not guaranteed to converge to the the joint posterior (8).

Prior on \mathbf{K}	Discr/Cont.	Method name	Reference	Conv.
G-Wishart	Discrete	RJ - O	Dobra et al. (2011)	
		RJ - WL	Wang & Li (2012)	
		RJ - CL	Cheng & Lenkoski (2012)	
		RJ - D	Lenkoski (2013)	✓
		RJ - DCBF	Hinne et al. (2014)	✓
		RJ - A	Mohammadi et al. (2022)	
		RJ - WWA	Van den Boom et al. (2022a)	✓
		RJ - UB	Van den Boom et al. (2022b)	✓
		BD - O	Mohammadi & Wit (2015)	✓
	Continuous	BD - A	Mohammadi et al. (2023a)	
Spike-and-slab	Discrete	SS	Wang (2015)	

Table 2: *Overview of joint MCMC methods for Bayesian structure learning in GGMs with their corresponding prior on \mathbf{K} , type of Markov chain (discrete or continuous), name, and reference. The last column indicates whether an algorithm theoretically converges to the posterior distribution.*

3.1.2 MCMC-based approaches on the graph space

Joint MCMC methods move over the joint space of graphs and precision matrices. An additional benefit of this strategy is that with the samples $(G^{(1)}, \mathbf{K}^{(1)}), \dots, (G^{(S)}, \mathbf{K}^{(S)})$, one can perform inference on the precision matrix. However, one is often just interested in

retrieving the structure of the true graph. In that case, obtaining a new precision matrix at every iteration is a computational burden. The class of methods in this subsection overcomes this burden by only exploring the graph space. We also refer to these types of methods as G-methods.

Stingo & Marchetti (2014) introduce this strategy in their G-Stingo method. They start with the well-known result that if a graph G_d is decomposable, the joint likelihood $P(\mathbf{X}|\mathbf{K}, G_d)$ can be written as the product of p different regressions:

$$P(\mathbf{X}|\mathbf{K}, G_d) = \prod_{i=1}^p P(X_i|X_{-i}, G_d, \mathbf{K}) = \prod_{i=1}^p \sum_{j \neq i} \beta_{ij} X_j + \epsilon_i. \quad (17)$$

Note the connection with the frequentist regression approach in (3). They then observe that every general graph G is a subgraph of a decomposable graph. Every graph G can therefore be denoted by the pair (G_d, L) , where G_d is decomposable and L is a subgraph of G_d . By putting priors on β_{ij} and ϵ_i , they then construct an MCMC algorithm that moves from $(G^{(s)}, L^{(s)})$ to $(G^{(s+1)}, L^{(s+1)})$ for $s = 0, \dots, S - 1$. This Markov chain approximates the posterior distribution $P(G, L|\mathbf{X})$, but the balance conditions do not hold. Therefore, the G-Stingo method does not have theoretical convergence.

Equation (17) only holds when the graph G_d is decomposable. When G is a general graph, this no longer holds. However, we can use the same strategy to find an approximation of the joint likelihood:

$$P(\mathbf{X}|\mathbf{K}, G) \approx \prod_{i=1}^p P(X_i|X_{-i}, G, \mathbf{K}). \quad (18)$$

The right-hand side of this expression is called the pseudo-likelihood. We can use it to approximate the marginal likelihood $P(\mathbf{X}|G)$:

$$\begin{aligned} P(\mathbf{X}|G) &= \int_{\mathbf{K}} P(\mathbf{X}|G, \mathbf{K}) P(\mathbf{K}|G) d\mathbf{K} \\ &\approx \int_{\mathbf{K}} \prod_{i=1}^p P(X_i|X_{-i}, \mathbf{K}, G) P(\mathbf{K}|G) d\mathbf{K} \\ &:= \hat{P}(\mathbf{X}|G), \end{aligned} \quad (19)$$

where $\hat{P}(\mathbf{X}|G)$ is called the marginal pseudo likelihood (MPL). Leppä-aho et al. (2017) introduces the concept of marginal pseudo likelihood to GGMs. They use the Wishart prior with $b = p$ and $D = \frac{1}{n}\mathbf{X}^T\mathbf{X}$. The Wishart prior is equivalent to the G -Wishart prior (10), where G is the complete graph. Note that this prior uses the data \mathbf{X} . Such priors are called fractional priors. Leppä-aho et al. (2017) choose this prior because it leads to a closed-form expression of $\hat{P}(\mathbf{X}|G)$, and then design a hill-climbing algorithm.

The G-MPL-Reversible Jump (G-MPLRJ) method by Mohammadi et al. (2023b) introduce the closed-form expression of $\hat{P}(\mathbf{X}|G)$ in an MCMC algorithm. Their discrete time MCMC algorithm creates the chain $(G^{(0)}, \dots, G^{(S)})$. Its pseudo-code resembles Algorithm 1 with two differences. First, no precision matrix is sampled at every iteration. Second, the acceptance probabilities $\alpha(G', G)$ are calculated using the marginal pseudo-likelihood, greatly improving the computational efficiency. The G-MPLRJ algorithm meets the balance conditions and therefore converges. However, the stationary distribution is not the true posterior $P(G|\mathbf{X})$, but instead $\hat{P}(G|\mathbf{X}) := \hat{P}(\mathbf{X}|G)P(G)/P(\mathbf{X})$. The same work also presents the G-MPL-Birth Death (G-MPLBD) method. Again using the approximation in (19), this method designs a continuous time MCMC similar to the BD-O algorithm. The resulting Markov chain also converges to $\hat{P}(G|\mathbf{X})$ instead of to the true posterior $P(G|\mathbf{X})$.

3.1.3 MCMC-based approaches on the precision matrix space

Methods on the space of precision matrices remove the necessity of sampling a graph at every iteration. They create an MCMC chain $(\mathbf{K}^{(1)}, \dots, \mathbf{K}^{(S)})$ solely over the space of positive definite $p \times p$ matrices. The methods are computationally efficient and have appealing theoretical properties. We also refer to these types of methods as K-methods. K-methods do not use a prior on the graph and are, therefore, not able to incorporate a prior belief in the graphical structure. Moreover, since K-methods do not provide MCMC samples in the graph space, they still require some operation to select a graph G from the posterior

samples of the precision matrix. Two examples of such operations are thresholding and credible intervals. The thresholding method (Wang 2012) first computes the edge inclusion probability p_e of an edge e and then includes it in the estimate of the graph if $p_e > \gamma$ where $\gamma \in (0, 1)$ is some threshold. The credible intervals method (Li et al. 2019a) first constructs intervals, for every edge (i, j) , that contains $\gamma\%$ of the values $\{k_{ij}^{(1)}, k_{ij}^{(2)}, \dots, k_{ij}^{(S)}\}$ for some user-defined γ . An edge is then excluded if and only if the interval includes zero.

For K-methods, the prior on the precision matrix $P(\mathbf{K}|G)$ no longer depends on G and can therefore be written as $P(\mathbf{K})$. We call these priors K-only priors. Most K-only priors are defined by two probability distributions. One for off-diagonal elements and one for diagonal elements. The elements of K-only priors are not independent due to the positive definite constraint and these priors also contain an intractable normalizing constant. Specific choices of K-only priors enable algorithms to be computationally efficient and/or have attractive theoretical properties. All K-methods are listed in Table 3 with their corresponding priors.

All K-methods share one or more of three useful theoretical properties. First, all K-methods meet the balance condition and therefore converge to the posterior distribution of the precision matrix (Tierney 1994, Theorem 1). Second, most K-methods have a connection with the frequentist graphical lasso algorithm. Specifically, priors of K-methods are often chosen such that the precision matrix that maximizes the posterior $P(\mathbf{K}|\mathbf{X})$ is equal to the graphical lasso estimate obtained by solving the optimization problem (5) for some regularization parameter λ and penalty function $p(\mathbf{K})$. The third and last theoretical property only holds for the K-Horseshoelike and the K-Low rank and diagonal (LRD) methods and builds off a theorem originally introduced by Banerjee & Ghosal (2015). It states that the posterior probability of precision matrices \mathbf{K} that are “far away” from the true precision matrix \mathbf{K}^* goes to zero, as the number of observations goes to infinity. Or, put differently, the posterior distribution $P(\mathbf{K}|\mathbf{X})$ is concentrated around the true value \mathbf{K}^* .

(Sagar et al. 2024, Theorem 4.6). Table 3 contains all K-methods with their corresponding theoretical properties.

Reference	Name	Value of $P(k_{ij})$		Theor. Prop.		
		$i \neq j$	$i = j$	i	ii	iii
Wang (2012)	K - BGLasso1	Double exp.	Exp.	✓	✓	
Khondker et al. (2013)	K - BGLasso2	Double exp.	Exp.	✓	✓	
Kundu et al. (2018)	K - RIW	Wishart	Wishart	✓		
Li et al. (2019a)	K - Horseshoe	Horseshoe	$\propto 1$	✓		
Sagar et al. (2024)	K - Horseshoe-like	Horseshoe-like	$\propto 1$	✓	✓	✓
Chandra et al. (2022)*	K - LRD	NA	NA	✓		✓
Smith et al. (2023b)*	K - BAGR	Norm.	Trunc. norm.	✓	✓	

Table 3: *K-methods for Bayesian structure learning with their corresponding reference, name, K-only prior, and theoretical properties. Note that all marginal K-only priors are not equal to the distributions mentioned in the table due to the positive definite constraint. The horseshoe and horseshoe-like distribution are defined in the corresponding articles. The theoretical properties are i) the Markov chain converges to the posterior distribution, ii) the maximum a posteriori estimate equals the maximum likelihood estimate (5), and iii) the posterior concentrates around the true value. Papers with a * are not yet published.*

3.2 Expectation Conditional Maximization Approaches

MCMC-based approaches are popular due to the wide variety of inferences they can provide using Equation (14). They are, however, computationally demanding because of the large parameter space they explore. Expectation Conditional Maximization (ECM) methods are a more efficient alternative. They allow for the quick computation of the maximum a posteriori (MAP) estimate of the precision matrix but do not provide inference on the rest of the posterior. The three ECM methods in Bayesian structure learning are listed in Table 4.

Every iteration $s = 1, \dots, S$ of an ECM algorithm renders a new precision matrix $\mathbf{K}^{(s)}$.

After S iterations, where S is large enough, the sequence $(\mathbf{K}^{(1)}, \mathbf{K}^{(2)}, \dots, \mathbf{K}^{(S)})$ converges to the MAP estimator. In each iteration two steps are applied: the Expectation step (E-step) and the Conditional-Maximization step (CM-step). In iteration s , the E-step evaluates $Q(\mathbf{K}|\mathbf{K}^{(s-1)})$, defined as the expected value of the log posterior with respect to the current conditional distribution of G given \mathbf{K} and \mathbf{X} . In the CM-step the new precision matrix $\mathbf{K}^{(s)}$ is set to the matrix that optimizes $Q(\mathbf{K}|\mathbf{K}^{(s-1)})$. The pseudo-code is shown in Algorithm 4.

Algorithm 4: Expectation Conditional Maximization algorithm

Input: Data \mathbf{X} and an initial precision matrix $\mathbf{K}^{(0)}$.

for iteration $s = 1, \dots, S$ **do**

E-step: evaluate $Q(\mathbf{K}|\mathbf{K}^{s-1})$;

CM-step: set $\mathbf{K}^{(s)} = \arg \max_{\mathbf{K} \in \mathcal{K}} Q(\mathbf{K}|\mathbf{K}^{s-1})$;

Output: A sequence $\mathbf{K}^{(0)}, \mathbf{K}^{(1)}, \dots, \mathbf{K}^{(S)}$ that converges to the MAP estimate.

ECM methods in Bayesian structure learning differ in three aspects: their prior, their theoretical properties, and their output. See Table 4 for an overview. First, ECM methods define their prior on the precision matrix $P(\mathbf{K}|G)$ using three distributions. One for elements k_{ij} for which $(i, j) \in G$, one for elements k_{ij} for which $(i, j) \notin G$, and one for diagonal elements. Like the G-Wishart prior, SS prior, and K-only priors discussed before, the elements k_{ij} are not independent, because the matrix \mathbf{K} needs to be positive definite. The priors therefore also contain an intractable normalizing constant. Second, ECM methods provide consistent results on the graph and/or the precision matrix. For example, with their *ECM - Bayesian regularization for Graphical models with Unequal Shrinkage* (ECM-BAGUS) method Gan et al. (2019) show that the estimate of the precision matrix $\hat{\mathbf{K}}$ gets arbitrarily close to the true precision matrix \mathbf{K}^* as the number of observations goes to infinity. To be specific, under certain assumptions and for a constant C , they show that

$$\|\hat{\mathbf{K}} - \mathbf{K}^*\|_\infty \leq C \sqrt{\frac{\log p}{n}}, \quad (20)$$

where $\|\cdot\|_\infty$ denotes the l_∞ norm. The ECM-Horseshoelike (ECM-HL) method by Sagar

et al. (2024) proves a similar result. The ECM - BAGUS method is the only ECM method that also provides a consistent result for the graph. Third, like the K-methods of Section 3.1.3, ECM methods require some post-hoc operation to provide inference on the graph. The ECM-BAGUS method is thereby the only ECM method that outputs the edge inclusion probabilities (13).

A limitation of ECM methods is that they only work for specifically designed priors on the precision matrix. Sagar et al. (2023) present a method that can output a MAP estimate of the precision matrix for any prior on the precision matrix, as long as the prior densities of the off-diagonal elements are completely monotone. Completely monotone priors form a large class of priors that include the priors of the K-Horseshoe and ECM-HL methods.

Reference	Name	Value of $P(k_{ij} G)$			Theory		Output	
		$(i, j) \in G$	$(i, j) \notin G$	$i = j$	i	ii	\mathbf{K}_{MAP}	p_e
Gan et al. (2019)	ECM - BAGUS	$DE(0, v_1)$	$DE(0, v_0)$	$\text{Exp}(\eta)$	✓	✓	✓	✓
Li & McCormick (2019)	ECM - EMGS	$N(0, v_1)$	$N(0, v_0)$	$\text{Exp}(\eta)$			✓	
Sagar et al. (2024)	ECM - HL	Horseshoe-like	$\propto 1$		✓		✓	

Table 4: *ECM methods for Bayesian structure learning with their corresponding reference, name, prior on \mathbf{K} , theoretical properties, and output. $DE(a, b)$ denotes the double exponential distribution with mean a and scale parameter b . $N(a, b)$ denotes the normal distribution with mean a and standard deviation b . $\text{Exp}(\eta)$ is the exponential distribution with rate η . Note that all marginal priors are not equal to the distributions mentioned in the table due to the positive definite constraint. The theoretical properties are i) the MAP estimate is consistent, and ii) the estimate of the graph is consistent. The output is the MAP estimate of the precision matrix (\mathbf{K}_{MAP}) and/or the edge inclusion probabilities (p_e) of (13).*

3.3 Regression Approaches

In Section 2.2, we saw that structure learning is equivalent to finding the coefficients of p different regressions. That is, estimating the coefficients β_{ij} in (3) allows for direct

estimation of the graphical model, since $\beta_{ij} = \frac{k_{ij}}{k_{ii}}$ and hence $\beta_{ij} = 0 \iff (i, j) \notin G$. Regression methods use a Bayesian approach to find the coefficients β_{ij} . Regression methods offer a simple approach to Bayesian structure learning. The resulting estimates of the graph and precision matrix, however, come without any theoretical guarantees.

There are two regression methods: The R - Bayesian Lasso Neighborhood Regression Estimate (R - BLNRE) method by Li & Zhang (2017) and the R - Projection (R-P) method by Williams et al. (2018). Both put priors on each coefficient β_{ij} and create Markov chains $(\beta_{ij}^{(1)}, \dots, \beta_{ij}^{(S)})$ for each β_{ij} . These MCMC samples are approximately samples from the posterior distribution $P(\beta_{ij}|\mathbf{X})$. Like MCMC-based approaches on the space of precision matrices (Section 3.1.3), regression methods now use a post-MCMC operation to obtain an estimate of the graph and precision matrix.

The R-BLNRE method finds the β_{ij} in (3) using the Bayesian lasso, a popular Bayesian algorithm for finding regressions coefficients by Park & Casella (2008). In the Bayesian lasso the prior on each coefficient β_{ij} is set to the double exponential distribution. The mode of the resulting posterior distribution $P(\beta_{ij}|\mathbf{X})$ equals the frequentist estimate obtained by solving the optimization problems in (4).

The R-P method by Williams et al. (2018) uses the horseshoe prior for the regression coefficients. The same prior is used for the elements of the precision matrix in the K-Horseshoe method by Li et al. (2019a). Using so-called projection predictive selection, the samples from the resulting posterior $P(\beta_i|\mathbf{X})$ are used to output an estimate of the graph and precision matrix.

3.4 Hypothesis Approaches

Hypothesis methods calculate the edge inclusion probabilities in (13) directly. They thereby circumvent the time-consuming exploration of the model space necessary for MCMC-based methods. To do so, they first formulate, for every (i, j) , a null hypothesis $H_0 : k_{ij} = 0$ and

an alternative hypothesis $H_1 : k_{ij} \neq 0$. Then, they calculate the Bayes factor in favour of H_1

$$\begin{aligned} BF_{ij} &= \frac{P(\mathbf{X}|H_1)}{P(\mathbf{X}|H_0)} \\ &= \frac{\int_{\mathbf{K}} P(\mathbf{X}|\mathbf{K})P(\mathbf{K})d\mathbf{K}}{\int_{\mathbf{K}:k_{ij}=0} P(\mathbf{X}|\mathbf{K})P(\mathbf{K})d\mathbf{K}}. \end{aligned}$$

This strategy was introduced by Giudici (1995), who derived a closed-form expression of BF_{ij} when the number of observations is bigger than the number of variables, *i.e.*, $p < n$. The Hypothesis Leday and Richardson (H - LR) method by Leday & Richardson (2018) builds upon this result by deriving a closed-form expression for BF_{ij} that also holds for $p > n$ and show that it is consistent, *i.e.*, $\lim_{n \rightarrow \infty} BF_{ij} = 0$, if H_0 is true and $\lim_{n \rightarrow \infty} BF_{ij} = \infty$, if H_1 is true. This Bayes factor, however, is not scale-invariant. To correct for this, the H-LR method first scales BF_{ij} to obtain the scaled Bayes factor sBF_{ij} . Let $P(sBF_{ij} > b|H_0)$ denote the tail probability that one observes a scaled Bayes factor higher than some constant b , given that H_0 is true. When H_0 is true, sBF_{ij} follows a beta distribution. The tail distribution $P(sBF_{ij} > b|H_0)$ can thus be computed exactly. The edge inclusion probabilities now follow by setting $P((i, j) \in G) = 1 - P(sBF_{ij} > b|H_0)$. For a prior on the precision matrix they use the Wishart distribution, which is equivalent to the G-Wishart distribution $P(\mathbf{K}|G)$ in Equation (10) when G is the complete graph. The H-LR method does not use a prior on the graph and is therefore, like the K-methods, not able to incorporate a prior belief on the graphical structure.

The Hypothesis - BGGM (H-BGGM) method by Williams & Mulder (2020) formulates the hypothesis using the partial correlations ρ_{ij} in (1). It is named after their R package Bayesian Gaussian Graphical Models (BGGM). The resulting hypotheses are $H_0 : \rho_{ij} = 0$, $H_1 : \rho_{ij} > 0$, and $H_2 : \rho_{ij} < 0$. They also formulate an unrestricted hypothesis $H_u : \rho_{ij} \in (-1, 1)$. These hypotheses allow for the computation of a scale-invariant Bayes factor. The

edge exclusion probability $P((i, j) \notin G)$ can now be calculated as follows:

$$\begin{aligned}
P((i, j) \notin G) &= P(H_0 | \mathbf{X}) \\
&= \frac{P(\mathbf{X} | H_0) P(H_0)}{P(\mathbf{X})} \\
&= \frac{P(\mathbf{X} | H_0) P(H_0)}{P(\mathbf{X} | H_0) P(H_0) + P(\mathbf{X} | H_1) P(H_1) + P(\mathbf{X} | H_2) P(H_2)} \\
&= \frac{BF_{0u} P(H_0)}{BF_{0u} P(H_0) + BF_{1u} P(H_1) + BF_{2u} P(H_2)}.
\end{aligned}$$

Here, $BF_{ku} = \frac{P(\mathbf{X} | H_k)}{P(\mathbf{X} | H_u)} = \frac{P(H_k | \mathbf{X})}{P(H_k)}$ for $k = 0, 1, 2$. $P(H_k)$ is the prior distribution for accepting hypothesis H_k . The Bayes factor BF_{ku} is therefore just a ratio of the posterior and prior of the hypotheses H_k . The authors compute this ratio using three priors. First, using the Wishart prior, they show that if the precision matrix K has a Wishart prior, the prior and posterior partial correlations approximately follow normal distributions. The Bayes factors BF_{ku} can then be computed in two ways: analytically or by sampling repeatedly from the Wishart distribution and using Equation 1. The Wishart distribution, however, is not able to incorporate prior beliefs on the graph. Therefore, they propose two other types of prior distributions for the precision matrix: the Corrected Wishart (CW) prior and the matrix-F prior. For both prior distributions, they show that the Bayes factor BF_{ku} can be computed using the so-called Savage-Dickey ratio by Dickey (1971). The H-BGGM method can also be used for confirmatory hypothesis tests. These kinds of hypothesis tests involve more than one partial correlation; for example, $H_0 : \rho_{12} > \rho_{34}$ and $H_1 : \rho_{12} \leq \rho_{34}$.

4 Empirical Comparison

The simulation study in this section contributes to the literature in three ways. First, it includes large-scale instances, *i.e.*, instances with a thousand variables. Until now, Bayesian algorithms are only compared on 250 variables or less, an exception being the K-LRD algorithm by Chandra et al. (2022). Second, we report information on how long time it

	Frequentist	RJ	BD	SS	G-methods	K-methods	ECM	Regression	Hypothesis
Frequentist		<i>missing</i>	2015	2015	<i>missing</i>	2024	2024	2018	2020
RJ			2022a	<i>missing</i>	2014	<i>missing</i>	<i>missing</i>	<i>missing</i>	2018
BD				<i>missing</i>	<i>missing</i>	<i>missing</i>	<i>missing</i>	2018	2020
SS					<i>missing</i>	<i>missing</i>	<i>missing</i>	<i>missing</i>	<i>missing</i>
G-methods						<i>missing</i>	<i>missing</i>	<i>missing</i>	<i>missing</i>
K-methods							2024	2018	<i>missing</i>
ECM								<i>missing</i>	<i>missing</i>
Regression									<i>missing</i>
Hypothesis									

Table 5: *Existing and missing empirical comparisons among different classes of Bayesian methods.*

takes algorithms to achieve good results. This information is missing in Bayesian literature. Third, and most importantly, never before has such a variety of Bayesian algorithms been compared in a single simulation study. All our results can be reproduced by the specified scripts on our GitHub page¹. We discuss the simulation setup in Subsection 4.1 and the results in Subsection 4.2.

4.1 Simulation Setup

Selected algorithms: In Bayesian structure learning literature, new algorithms tend to be compared to frequentist algorithms, or Bayesian algorithms of similar nature, *e.g.*, K-algorithms are compared to K-algorithms, ECM algorithms to ECM algorithms, and so on. Comparisons between dissimilar Bayesian algorithms are often outdated or missing completely. To illustrate this, we group Bayesian methods into seven categories. Table 5 shows that comparisons between different categories are largely missing in the literature.

In this section, we complete most of this table by comparing seven Bayesian algorithms

¹<https://anonymous.4open.science/r/Review-paper-Bayesian-Structure-Learning-in-GGMs-6658>

(RJ - WWA, BD-A, SS, G-MPLBD, K-Horseshoe, ECM-BAGUS, and H-BGGM) and one frequentist algorithm (glasso). For this simulation study, Bayesian algorithms are chosen based on three criteria: i) the algorithm demonstrates outstanding performance in its category, ii) there is publicly accessible working code for the algorithm, and iii) the algorithm is documented in a published article. No regression algorithm is selected because there is no code available. As a prior for the graph we sample the edges from independent and identically distributed Bernoulli distributions. The resulting prior becomes

$$P(G) = \delta^{|E|}(1 - \delta)^{p(p-1)/2 - |E|}, \quad (21)$$

where δ is the prior sparsity of the graph. In literature this prior sparsity is often set to either the uninformative value $\delta = 0.5$ (Gan et al. 2019) or to a function that decreases with p , for example $2/(p - 1)$ (Van den Boom et al. 2022a). In this simulation study, we opt for a middle ground and set $\delta = 0.2$. Supplementary materials S1 and S2 contain the source and all other settings of the algorithms.

Performance metrics: A metric should measure the distance between an estimate and the truth. In the case of Bayesian structure learning, however, the true posterior distribution $P(G|\mathbf{X})$ is unknown. In the absence of a true posterior distribution, it is therefore common in the literature to opt for the second best choice: compare the estimate of the posterior against the true graph G^* . For lack of a better alternative, we adopt the same strategy too.

We compare the selected algorithms based on the accuracy of the edge inclusion probabilities p_e defined in (13). Let P be the matrix with elements p_e . We refer to P as the edge inclusion matrix. We use three metrics for the accuracy of the edge inclusion probabilities: AUC, Pr^+ , and Pr^- . The AUC represents the area under the Receiver Operating Characteristic (ROC) curve. It reflects to what degree the edge inclusion probabilities of links in the true graph are higher than those of links not in the true graph. It ranges from 0

(worst) to 1 (best). An AUC of 0.5 means the edge inclusion matrix is as good as a random guess. The glasso and K-Horseshoe algorithms do not output edge inclusion probabilities. To calculate the AUC of the glasso algorithm, we solve the minimization problem in (5) for different values of λ . The area under the resulting ROC curve gives us the AUC. For the K-Horseshoe algorithm we use the same strategy as Li et al. (2019a), *i.e.*, the ROC curve is generated by varying the length of posterior credible intervals from 1% to 99%.

The AUC is a measure for the ranking of the edge inclusion probabilities but does not convey their magnitude. We therefore add two metrics for the magnitude: one measuring the algorithm's ability to predict the presence of an edge (Pr^+), the other the ability to predict the absence of an edge (Pr^-). Formally, we have

$$Pr^+ = \frac{1}{|E^*|} \sum_{(i,j) \in G^*} p_e \quad (22)$$

and

$$Pr^- = \frac{1}{|E^{*-}|} \sum_{(i,j) \notin G^*} p_e, \quad (23)$$

where $|A|$ denotes the number of elements in the set A and E^{*-} denotes the set of links that are not in the true graph. Pr^+ ranges from 0 (worst) to 1 (best), whereas Pr^- ranges from 1 (worst) to 0 (best). As before, these metrics need to be redefined for the glasso and K-Horseshoe algorithms, since these algorithms do not output edge inclusion probabilities, but a point estimate of the true graph. By setting $p_e = 1$ if $e \in \hat{G}$, and zero if $e \notin \hat{G}$, we can still calculate Pr^+ and Pr^- . Note that Pr^+ (Pr^-) then becomes the true (false) positive rate of the estimate \hat{G} . Concerns might be raised about the fairness when comparing the Pr^+ and Pr^- metrics from a point estimate \hat{G} with those from the posterior edge inclusion probabilities p_e . However, in both cases, the Pr^+ and Pr^- metrics do provide valuable information and are thus included. We do recommend the reader proceed with caution when directly comparing the Pr^+ and Pr^- metrics from the glasso and K-horseshoe method with those from other algorithms.

We also measure the computational cost of each algorithm. We first run each Bayesian algorithm until visual inspection shows convergence and denote with P the final edge inclusion matrix. The computational cost T of an algorithm is then defined as the time it takes the algorithm to produce an AUC within $\epsilon = 0.01$ of its final AUC. That is,

$$T = \min(t : |AUC(P_t) - AUC(P)| < \epsilon), \quad (24)$$

where we set $\epsilon = 0.01$. Here, P_t denotes the edge inclusion matrix after t seconds of running the algorithm and $AUC(P)$ denotes the AUC corresponding to an edge inclusion matrix P . In real-world applications, the true graph is unknown and the computational cost T is not defined. For convergence assessments without knowing the true graph, we refer to Section 5. The computational cost of the H-BGGM and glasso algorithms can not be defined using (24). Instead, we define their computation cost T as the total time it takes these algorithms to provide their edge inclusion probabilities. We note that both the glasso and ECM-BAGUS algorithms require parameter tuning. We include this time in the computational cost.

Simulation instances: We include small ($p = 10$), medium ($p = 100$) and large ($p = 1000$) scale graphs with a low ($n = 20 \log p$) and high ($n = 350 \log p$) number of observations. The use of the logarithm here is motivated by the performance bound given by (20). We consider the following two graph types:

1. Random: a graph in which every edge (i, j) is drawn from an independent Bernoulli distribution with probability δ .
2. Cluster: a graph with two clusters for $p \in \{10, 100\}$ and eight clusters for $p = 1000$.

Each cluster has the same structure as the random graph.

The value of δ is chosen so that the resulting edge density in the graphs is 10% for $p = 10$, 1% and 10% for $p = 100$, and 0.1% for $p = 1000$. For every generated graph G^* , the

corresponding precision matrix \mathbf{K}^* is then sampled from the G -Wishart distribution. That is $\mathbf{K}^* \sim W_{G^*}(3, \mathbf{I})$. For $p \in \{10, 100, 1000\}$, each n , each graph type, and each density, we repeat the process 16 times which is generally enough to reach low (< 0.025) standard errors on AUC, Pr^+ and Pr^- . For each generated G^* and \mathbf{K}^* , we then sample n observations from the p -variate normal distribution with covariance matrix $\Sigma^* = \mathbf{K}^{*-1}$ and mean zero.

4.2 Results

The results on the computational costs T , AUC, Pr^- and Pr^+ , and are shown, respectively, in Tables 6, 7, 8, and 9. If there is one conclusion to be drawn from these results, then it is this: the idea that Bayesian algorithms lack the computational advantage of their frequentist counterparts is outdated. Whether with 10, 100, or 1000 variables, one or more Bayesian algorithms can estimate the full posterior distribution or any function thereof at the same time that the glasso algorithm can provide a point estimate. Moreover, there is no evidence of the glasso algorithm outperforming Bayesian algorithms on AUC, Pr^+ , or Pr^- either. It underlines the staggering progress Bayesian algorithms have made over the last decade. This progress is illustrated too in Figure 1, which shows the convergence of performance metrics for each algorithm on a large scale instance ($p = 1000$): the AUC, Pr^+ and Pr^- values of the G-MPLBD, SS-O, and K-Horseshoe algorithms converge in approximately the same time as the glasso algorithm.

Now, let us look into the results in more detail, starting with the computational cost in Table 6. Unsurprisingly, the glasso algorithm is fast. It provides a point estimate of the graph within seconds for small and medium-sized problems and within 10 minutes for large-scale problems. The majority of these 10 minutes is spent on tuning the regularization parameter λ in (5). More surprising is the outstanding performance of the G-MPLBD algorithm that allows inference on the entire posterior within the same time as the glasso provides a point estimate. The most time-consuming algorithms are the RJ-WWA and

BD-A algorithms, which are slow to run on large-scale instances. This is due to the computationally expensive sampling from the G-Wishart distribution at every iteration and the slow edge-by-edge exploration of the state space. Moreover, both algorithms benefit from parallel computing but are run on one core in this simulation study. The H-BGGM algorithm gives errors for the high dimensional case $n < p$ and for the case $p = 1000$. The fields corresponding to these cases are therefore left empty.

In terms of the AUC, most algorithms perform similarly. Noteworthy is the overall underperformance of the ECM-BAGUS algorithm and the underperformance of the glasso algorithm in medium-sized problems ($p = 100$) with a high edge density (10%). The Pr^- values are comparable across most algorithms too with estimates close to the desired value of zero. Only the ECM-BAGUS and BGGM algorithms perform significantly worse, the latter even assigning in some cases an average edge inclusion probability of 0.88 to edges that are not in the true graph. Looking at the Pr^+ values, we see more variability between algorithms. For instances with a high number of observations, the K-Horseshoe algorithm performs best. For large-scale graphs, the K-horseshoe algorithm outperforms the others too. The ECM-BAGUS algorithm performs slightly worse than others in most instances. The Pr^+ values of the BGGM algorithm seem close to optimal but are only marginally more than their Pr^- values.

The results also reveal that it is not so much the structure of the graph (random or cluster) as the density that influences the performance of algorithms. More observations lead to better results. Generally, $n = 350 \log p$ observations are sufficient to yield good results.

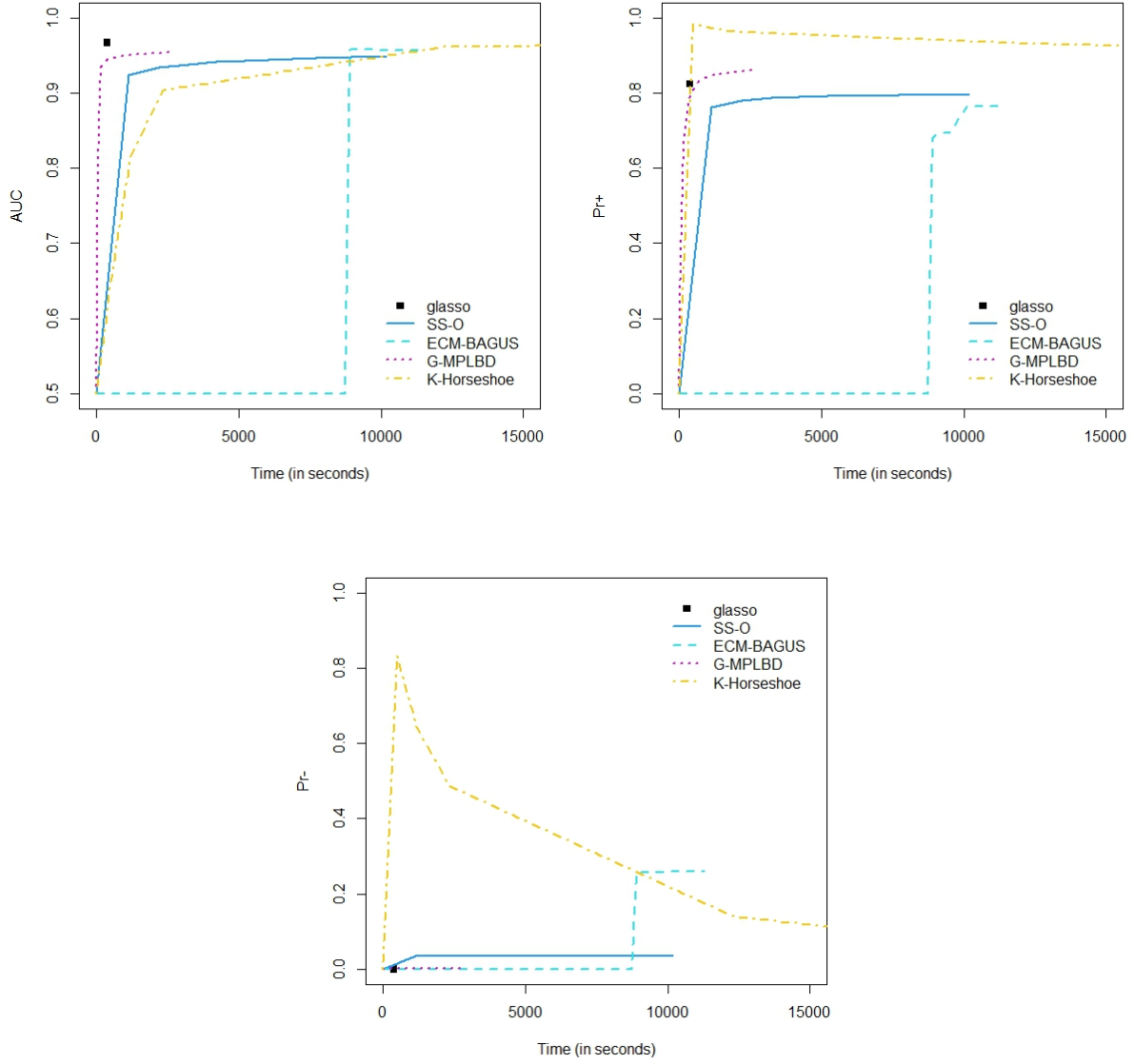


Figure 1: AUC (topleft), Pr^+ (topright) and Pr^- (bottom) over time. The figures show averages over 16 replications on an instance with $p = 1000$, $n = 1050$ on the random graph with density 0.1%.

p	graph	density	n	glasso	RJ-WWA	BD-A	SS	ECM-BAGUS	G-MPLBD	K-Horseshoe	H-BGGM
10	random	10 %	20	2 (0)	1 (1)	0 (1)	0 (0)	1 (0)	0 (0)	0 (0)	0 (0)
			350	2 (0)	0 (1)	0 (0)	0 (0)	1 (0)	0 (0)	0 (1)	0 (0)
	cluster	10 %	20	2 (0)	1 (1)	0 (0)	0 (0)	1 (0)	0 (0)	0 (1)	0 (0)
			350	2 (0)	0 (0)	0 (0)	0 (0)	1 (0)	0 (0)	0 (0)	0 (0)
100	random	1 %	40	3 (0)	3623 (4183)	5313 (4190)	5 (6)	84 (9)	4 (4)	49 (56)	-
			700	3 (0)	147 (77)	1835 (2934)	19 (22)	76 (18)	2 (2)	3 (1)	11 (0)
	cluster	10 %	40	5 (1)	2794 (1513)	9018 (3161)	10 (5)	252 (80)	6 (4)	32 (22)	-
			700	5 (1)	746 (441)	6093 (2319)	12 (5)	482 (164)	8 (3)	18 (13)	10 (0)
			40	3 (0)	620 (402)	4496 (3730)	13 (16)	83 (6)	4 (3)	11 (13)	-
			700	3 (0)	159 (130)	3013 (4124)	19 (17)	70 (12)	1 (2)	5 (3)	11 (0)
1000	random	0.1 %	40	4 (0)	3489 (1499)	10368 (2727)	13 (7)	186 (50)	5 (3)	17 (18)	-
			700	3 (0)	1569 (759)	6465 (3461)	10 (7)	349 (127)	6 (3)	10 (5)	11 (0)
	cluster	0.1 %	60	606 (54)	-	-	9063 (1538)	15057 (1873)	624 (19)	24064 (417)	-
			1050	354 (55)	-	-	4007 (1519)	8907 (1158)	492 (163)	12300 (200)	-
			60	672 (149)	-	-	8510 (1513)	15710 (1214)	623 (35)	7451 (7762)	-
			1050	366 (84)	-	-	3823 (1861)	8224 (1539)	481 (198)	12389 (128)	-

Table 6: Computational cost T (24) in seconds of the algorithms for different instances. The values are averages and standard deviations (between brackets) over 16 replications. The “-” entry indicates that an algorithm did not produce results on that instance.

p	graph	density	n	glasso	RJ-WVA	BD-A	SS	ECM-BAGUS	G-MPLBD	K-Horseshoe	H-BGGM
10	random	10 %	20	0.7 (0.24)	0.72 (0.24)	0.73 (0.21)	0.73 (0.22)	0.7 (0.21)	0.72 (0.22)	0.7 (0.23)	0.67 (0.24)
			350	0.94 (0.08)	0.95 (0.08)	0.95 (0.08)	0.93 (0.09)	0.91 (0.1)	0.95 (0.07)	0.94 (0.06)	0.93 (0.08)
	cluster	10 %	20	0.83 (0.14)	0.82 (0.16)	0.82 (0.16)	0.81 (0.14)	0.79 (0.13)	0.82 (0.13)	0.81 (0.17)	0.75 (0.15)
			350	0.93 (0.08)	0.93 (0.1)	0.93 (0.1)	0.91 (0.14)	0.92 (0.1)	0.94 (0.1)	0.92 (0.11)	0.92 (0.12)
100	random	1 %	40	0.83 (0.06)	0.83 (0.07)	0.81 (0.07)	0.84 (0.06)	0.76 (0.06)	0.81 (0.07)	0.84 (0.07)	-
			700	0.96 (0.03)	0.96 (0.03)	0.96 (0.03)	0.94 (0.02)	0.94 (0.03)	0.95 (0.03)	0.96 (0.02)	0.95 (0.02)
		10 %	40	0.67 (0.03)	0.7 (0.01)	0.68 (0.02)	0.71 (0.02)	0.59 (0.02)	0.67 (0.02)	0.71 (0.02)	-
			700	0.78 (0.06)	0.92 (0.01)	0.91 (0.01)	0.89 (0.01)	0.8 (0.07)	0.9 (0.01)	0.92 (0.01)	0.87 (0.01)
	cluster	1 %	40	0.83 (0.04)	0.85 (0.04)	0.83 (0.05)	0.83 (0.04)	0.78 (0.04)	0.81 (0.04)	0.85 (0.05)	-
			700	0.95 (0.02)	0.96 (0.02)	0.96 (0.02)	0.94 (0.02)	0.94 (0.02)	0.94 (0.02)	0.95 (0.03)	0.94 (0.02)
		10 %	40	0.71 (0.02)	0.77 (0.02)	0.72 (0.02)	0.75 (0.02)	0.61 (0.02)	0.7 (0.02)	0.73 (0.01)	-
			700	0.84 (0.02)	0.94 (0.01)	0.93 (0.01)	0.91 (0.01)	0.84 (0.04)	0.92 (0.01)	0.93 (0.01)	0.87 (0.01)
1000	random	0.1 %	60	0.86 (0.03)	-	-	0.84 (0.03)	0.84 (0.03)	0.72 (0.04)	0.84 (0.03)	-
			1050	0.97 (0.01)	-	-	0.95 (0.01)	0.96 (0.01)	0.95 (0.01)	0.96 (0.01)	-
	cluster	0.1 %	60	0.84 (0.04)	-	-	0.82 (0.05)	0.84 (0.03)	0.7 (0.05)	0.84 (0.04)	-
			1050	0.96 (0.01)	-	-	0.95 (0.02)	0.95 (0.02)	0.95 (0.01)	0.96 (0.01)	-

Table 7: *AUC scores of the algorithms for different instances. The AUC reaches its best score at 1 and its worst at 0. The values are averages and standard deviations (between brackets) over 16 replications. The “-” entry indicates that an algorithm did not produce results on that instance.*

p	graph	density	n	glasso	RJ-WVA	BD-A	SS	ECM-BAGUS	G-MPLBD	K-Horseshoe	H-BGGM
10	random	10 %	20	0.05 (0.05)	0.09 (0.02)	0.1 (0.02)	0.08 (0.01)	0.18 (0.1)	0.08 (0.02)	0.06 (0.04)	0.88 (0.01)
			350	0.05 (0.04)	0.02 (0.01)	0.03 (0.01)	0.03 (0)	0.15 (0.07)	0.01 (0.01)	0.09 (0.06)	0.33 (0.03)
100	cluster	10 %	20	0.04 (0.03)	0.08 (0.01)	0.09 (0.02)	0.08 (0.01)	0.2 (0.12)	0.07 (0.03)	0.02 (0.02)	0.88 (0.01)
			350	0.06 (0.04)	0.02 (0.01)	0.03 (0.01)	0.03 (0)	0.12 (0.05)	0.01 (0.01)	0.07 (0.05)	0.31 (0.02)
	random	1 %	40	0 (0)	0.05 (0)	0.07 (0)	0.09 (0)	0.16 (0.05)	0.03 (0)	0 (0)	-
			700	0.01 (0)	0.01 (0)	0.02 (0)	0.03 (0)	0.11 (0.03)	0.01 (0)	0 (0)	0.23 (0)
1000	cluster	10 %	40	0.13 (0.1)	0.1 (0.02)	0.12 (0.02)	0.12 (0.01)	0.12 (0.03)	0.04 (0.01)	0.01 (0)	-
			700	0.17 (0.04)	0.03 (0.01)	0.04 (0.01)	0.05 (0.02)	0.15 (0.04)	0 (0)	0.04 (0)	0.23 (0)
	random	1 %	40	0 (0)	0.05 (0)	0.07 (0)	0.09 (0)	0.18 (0.04)	0.03 (0)	0 (0)	-
			700	0 (0)	0.01 (0)	0.02 (0)	0.03 (0)	0.14 (0.04)	0.01 (0)	0 (0)	0.23 (0.01)
1000	cluster	10 %	40	0.06 (0.05)	0.08 (0.01)	0.11 (0.01)	0.11 (0.01)	0.12 (0.03)	0.04 (0.01)	0.01 (0)	-
			700	0.11 (0.02)	0.02 (0)	0.03 (0.01)	0.05 (0.01)	0.12 (0.03)	0 (0)	0.03 (0)	0.23 (0.01)
	random	0.1 %	60	0 (0)	-	-	0.18 (0)	0.29 (0.07)	0.1 (0.01)	0.05 (0)	-
			1050	0 (0)	-	-	0.04 (0)	0.26 (0.1)	0 (0)	0.05 (0)	-
1000	cluster	0.1 %	60	0 (0)	-	-	0.18 (0)	0.32 (0.08)	0.1 (0.01)	0.05 (0)	-
			1050	0 (0)	-	-	0.04 (0)	0.23 (0.1)	0 (0)	0.05 (0)	-

Table 8: Pr^- scores (23) of the algorithms for different instances. The Pr^- reaches its best score at 0 and its worst at 1. The values are averages and standard deviations (between brackets) over 16 replications. The “-” entry indicates that an algorithm did not produce results on that instance.

p	graph	density	n	glasso	RJ-WVA	BD-A	SS	ECM-BAGUS	G-MPLBD	K-Horseshoe	H-BGGM
10	random	10 %	20	0.38 (0.34)	0.38 (0.31)	0.4 (0.31)	0.35 (0.29)	0.31 (0.21)	0.4 (0.33)	0.39 (0.37)	0.9 (0.03)
			350	0.87 (0.19)	0.77 (0.23)	0.78 (0.22)	0.56 (0.29)	0.69 (0.24)	0.81 (0.2)	0.85 (0.18)	0.9 (0.11)
100	cluster	10 %	20	0.45 (0.28)	0.45 (0.23)	0.47 (0.23)	0.4 (0.24)	0.35 (0.19)	0.5 (0.22)	0.45 (0.29)	0.91 (0.02)
			350	0.87 (0.15)	0.79 (0.19)	0.8 (0.19)	0.58 (0.23)	0.73 (0.2)	0.82 (0.2)	0.88 (0.17)	0.89 (0.13)
	random	1 %	40	0.3 (0.15)	0.46 (0.12)	0.5 (0.11)	0.5 (0.11)	0.46 (0.07)	0.47 (0.11)	0.41 (0.12)	-
			700	0.83 (0.08)	0.84 (0.08)	0.84 (0.08)	0.63 (0.13)	0.82 (0.06)	0.84 (0.07)	0.87 (0.02)	0.9 (0.04)
1000	cluster	10 %	40	0.3 (0.11)	0.31 (0.03)	0.32 (0.02)	0.29 (0.02)	0.22 (0.04)	0.22 (0.02)	0.24 (0.03)	-
			700	0.59 (0.03)	0.73 (0.02)	0.72 (0.02)	0.57 (0.08)	0.58 (0.11)	0.68 (0.02)	0.79 (0.02)	0.76 (0.01)
			40	0.26 (0.12)	0.56 (0.09)	0.51 (0.08)	0.5 (0.09)	0.46 (0.05)	0.47 (0.08)	0.43 (0.11)	-
	random	10 %	700	0.81 (0.06)	0.83 (0.05)	0.84 (0.04)	0.61 (0.08)	0.83 (0.04)	0.84 (0.04)	0.81 (0.06)	0.9 (0.03)
			40	0.27 (0.09)	0.33 (0.03)	0.35 (0.03)	0.32 (0.02)	0.24 (0.04)	0.26 (0.03)	0.26 (0.03)	-
			700	0.59 (0.03)	0.73 (0.03)	0.73 (0.03)	0.59 (0.06)	0.63 (0.05)	0.68 (0.03)	0.8 (0.03)	0.77 (0.02)
1000	random	0.1 %	60	0.24 (0.11)	-	-	0.61 (0.06)	0.52 (0.07)	0.38 (0.05)	0.62 (0.07)	-
			1050	0.83 (0.04)	-	-	0.8 (0.04)	0.76 (0.09)	0.86 (0.03)	0.91 (0.02)	-
	cluster	0.1 %	60	0.2 (0.15)	-	-	0.59 (0.09)	0.5 (0.08)	0.36 (0.07)	0.62 (0.08)	-
			1050	0.8 (0.06)	-	-	0.78 (0.05)	0.81 (0.08)	0.85 (0.04)	0.91 (0.02)	-

Table 9: Pr^+ scores (22) of the algorithms for different instances. The Pr^+ reaches its best score at 1 and its worst at 0. The values are averages and standard deviations (between brackets) over 16 replications. The “-” entry indicates that an algorithm did not produce results on that instance.

5 Case Study: Human Gene Expression

In this section we demonstrate the capabilities of Bayesian structure learning on a real-world data set. We discuss the challenges that arise when applying Bayesian approaches in practice and present ways to tackle these challenges. Consequently, this section serves as a guideline for practitioners who are new to the field.

We use the SS algorithm and the BD-A algorithm to perform inference on a gene network between $p = 100$ genes using genetic data of $n = 60$ unrelated individuals. Several Bayesian structure learning methods are analyzed on this dataset. See, for example, (Mohammadi & Wit 2015, Li et al. 2019a, Van den Boom et al. 2022a). This dataset can therefore be seen as a benchmark for Bayesian structure learning. We refer to Stranger et al. (2007) and Bhadra & Mallick (2013) for more information on the collection of these data.

Genes are specific sequences of DNA. Gene expression is the process by which a gene produces a protein that influences the functioning of an organism. Some of these proteins directly influence the organism by, for example, initializing a process to break down food. Other proteins, however, only serve to “activate” other genes, which in turn make proteins that activate other genes, and so on. These activation relationships can be shown in a gene network, where every node corresponds to a gene and every edge indicates an activation relationship between two genes. Discovering these gene networks plays a key role in understanding disease susceptibility and therefore, ultimately, in treatment and public health (Stranger et al. 2007). We obtain a sample \mathbf{X} of our variables X_i , $i = 1, \dots, p$, where \mathbf{X} is an $n \times p$ matrix with elements x_{li} denoting the level of protein corresponding to gene $i = 1, \dots, p$ measured in individual $l = 1, \dots, n$.

Structure learning depends on the assumption that the data comes from a multivariate normal distribution with a mean zero. However, in the gene data set, the univariate histograms of each gene are clearly not normally distributed. Neither are their means

zero, see Figure 2. In fact, the normality and zero mean assumptions are rarely met in

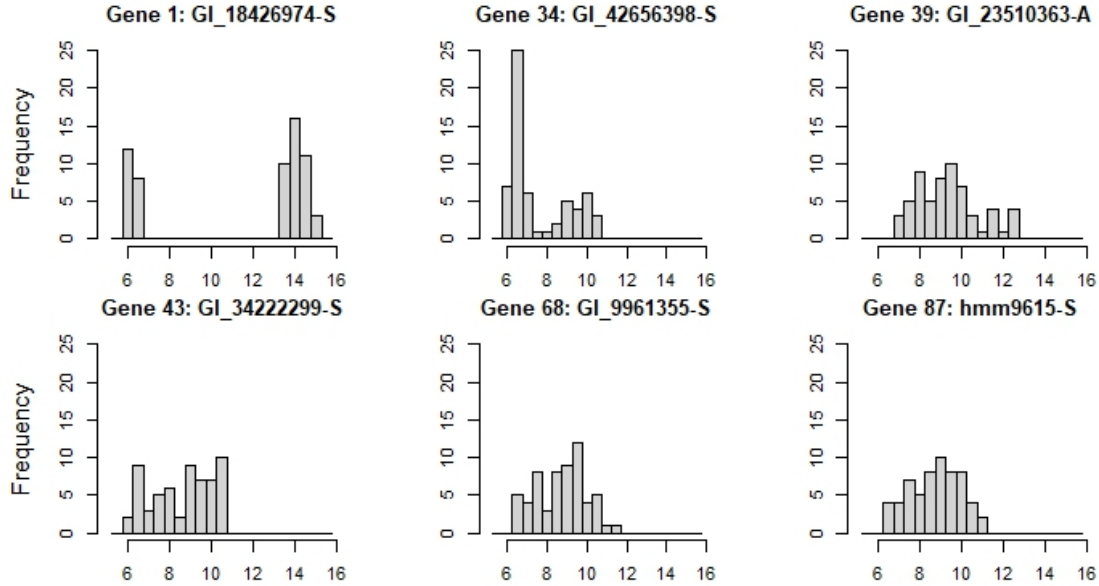


Figure 2: *Univariate histograms of six randomly selected genes in the human gene expression dataset.*

practice. Fortunately, one can transform our non-Gaussian continuous variables X_1, \dots, X_p to Gaussian variables $Z := (Z_1, \dots, Z_p)$ with mean zero, *i.e.* $Z \sim \mathcal{N}(\mathbf{0}, \Sigma_Z)$, such that the sparsity of $\mathbf{K}_Z = \Sigma_Z^{-1}$ still encodes the conditional (in)dependence between the variables X_1, \dots, X_p as in (2), see for example Liu et al. (2009, 2012). The *huge.npn()* function in the *Huge* R package and the *bdgraph.npn()* function in the *BDgraph* R package perform this transformation efficiently. One can now apply Bayesian structure learning algorithms on the transformed dataset \mathbf{Z} . These transformations are only possible for continuous data. For discrete or binary data, Gaussian copula graphical models can be used to perform structure learning, see Section 6.1 for an overview.

We choose to work with MCMC algorithms to perform inference on the posterior of the gene network. They offer a wide variety of output and are computationally feasible on this 100 node graph. Specifically, we use the SS algorithm and the BD-A algorithm, but the same steps apply to other MCMC algorithms.

First, we select the tuning parameters. For the SS algorithm, we follow Wang (2015) and select $v_0 = 0.02$ and $v_1 = 2$. One could also go for the more time-consuming approach and run the algorithm several times for different parameters, and select the parameters that perform best according to some criterion (Gan et al. 2019). The BD-A algorithm has no tuning parameters.

Second, we need to select a prior for the graph. Generally, the prior is selected using prior knowledge about the sparsity of the true graph. The resulting prior on the graph is then given by (21), where we select the prior sparsity equal to $\delta = 0.2$, in line with previous studies on the gene expression dataset; see for example, Van den Boom et al. (2022a) and Li et al. (2019a).

Now, the correct MCMC settings need to be selected. That is, the number of burn-in iterations, the number of MCMC iterations, and the initial graph $G^{(0)}$ of the Markov Chain. To do so, we run the algorithms several times, each time from a different initial graph, and report the number of edges in each Markov chain state. The results are shown in Figure 3.

The starting position should not be important, since the Markov chain should converge to the posterior distribution regardless of the starting graph. Figure 3 tells us that this indeed happens. Moreover, we can deduce that the first 20 (SS algorithm) and the first 15,000 iterations (BD-A algorithm) can be discarded as burn-in iterations. To be on the safe side, we select burn-in periods of 50 (SS) and 25,000 (BD-A) iterations. Figure 3 also shows that, after the burn-in period, the number of edges of the visited states differs between the SS algorithm (approx. 1,000) and the BD-A algorithm (approx. 1,500). This discrepancy might be caused by the approximations used by both algorithms. Moreover, the SS and BD-A methods use different priors for the precision matrix resulting in different posteriors.

To select the number of MCMC iterations (after burn-in), we plot the edge inclusion probabilities of ten randomly selected links at every state. The results are shown in Figure

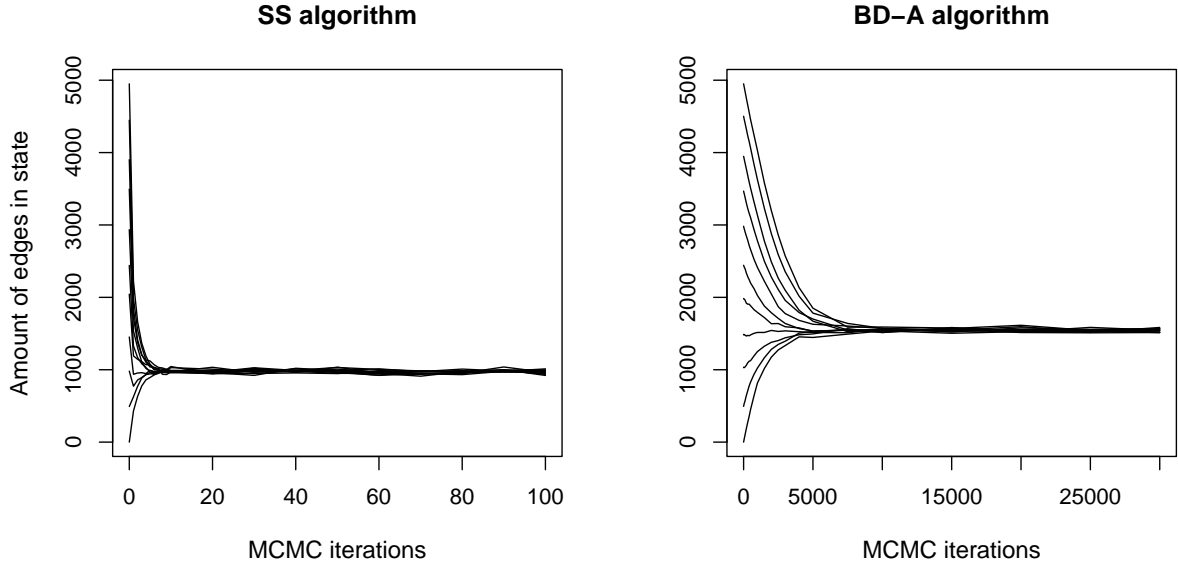


Figure 3: *Amount of edges in the visited graph per MCMC iteration for 10 replications. The density of the initial graph differs per replication, ranging from 0% (empty graph) to 100% (full graph).*

4. We see that convergence happens in less iterations for the SS algorithm (2,000) than for the BD-A algorithm (400,000). This is due to the good mixing of the SS algorithm versus the slow edge-by-edge exploration of the BD-A algorithm. Figure 4 also reveals for the SS algorithm that the final edge inclusion probabilities are all centered around 0.2. It is no coincidence that this is also the value of our prior sparsity. In fact, it turns out that the edge inclusion probabilities of the SS algorithm are very sensitive to the choice of the prior (Figure 5). Potential reasons for this prior sensitivity could be the low number of observations ($n = 60$) compared to the number of variables ($p = 100$), or the fact that the prior sparsity is directly incorporated in the iterative sampling of the graph (Wang 2015, Equation 12). Due to its prior sensitivity, the SS algorithm fails to distinguish between edges that are likely and unlikely to be in the true graph. It is not suitable for the dataset and therefore left out in the remainder of this section. The BD algorithm is also sensitive to the graph prior but manages in all cases to find both edges with high and low edge

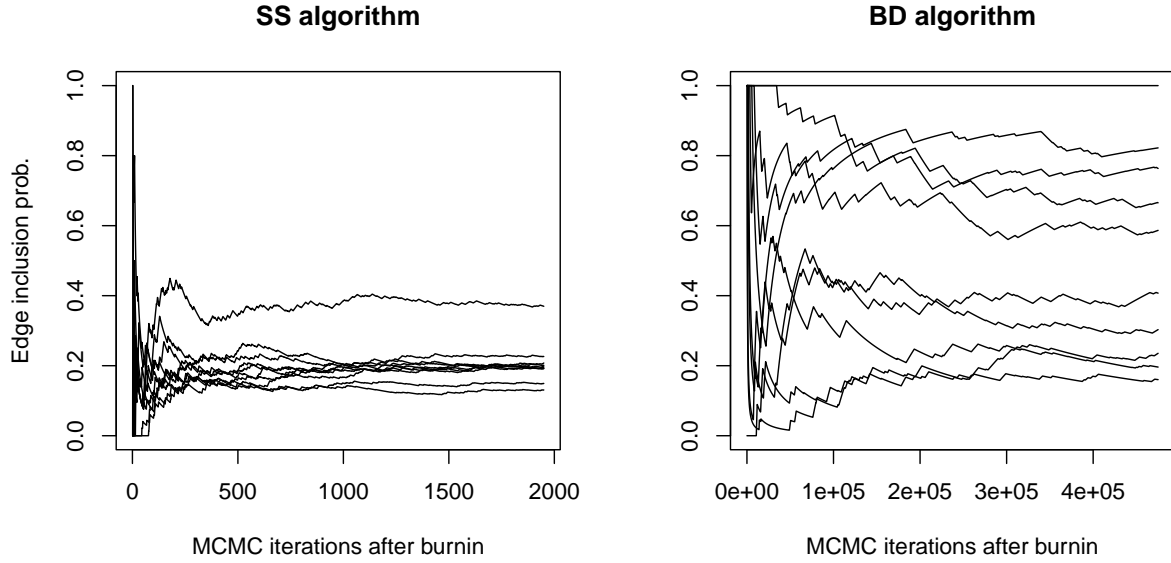


Figure 4: *The edge inclusion probabilities of ten randomly selected edges for every state in the Markov chain.*

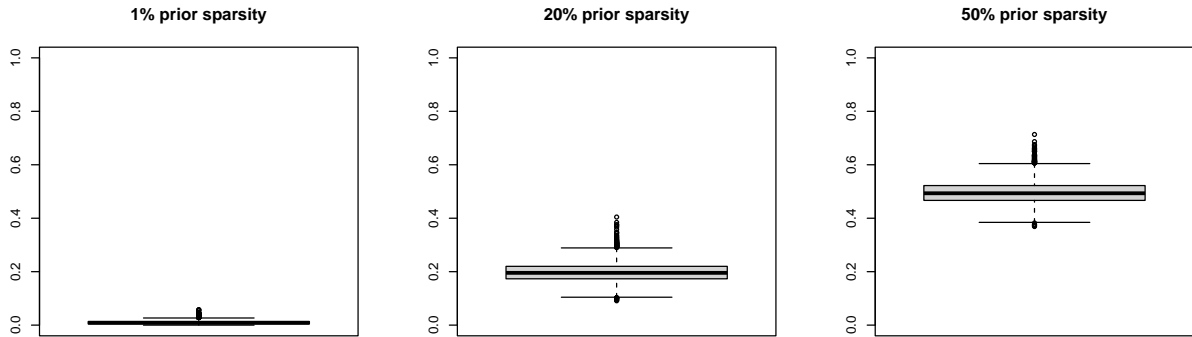


Figure 5: *Boxplots of the final edge inclusion probabilities of the SS algorithm.*

inclusion probabilities (Figure 6).

Figure 7 shows the gene network inferred by the BD-A algorithm. The results show similarities with the existing literature. The five most connected genes of the K-Horseshoe algorithm (Li et al. 2019a) all appear among the most connected genes of our analysis. Moreover, most tree structures of Figure 7 appear exactly in the recovered network given by Bhadra & Mallick (2013, Figure 4).

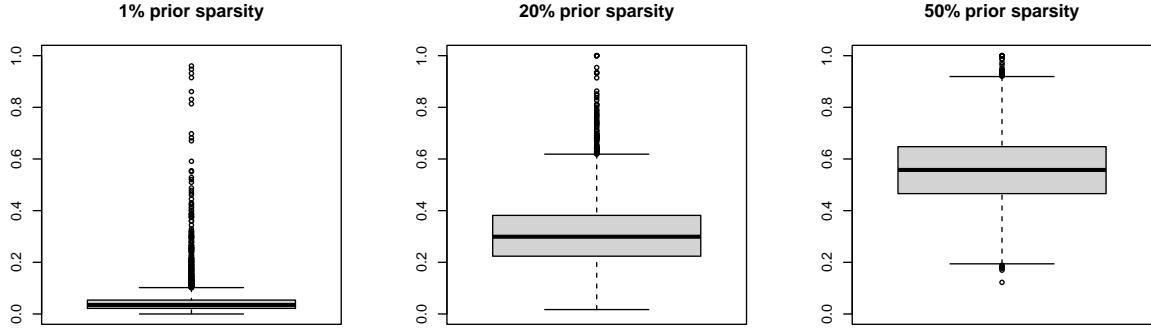


Figure 6: *Boxplots of the final edge inclusion probabilities of the BD algorithm.*

6 Conclusion and Future Research

In structure learning, Bayesian methods constitute a powerful alternative to frequentist ones by providing inference on the entire posterior. The argument that the speed and simplicity of frequentist methods are superior, is waning as Bayesian methods have improved significantly in both aspects. Bayesian methods that provide accurate solutions to thousand variable problems within mere minutes are now a reality and easily accessible in software packages. This section starts with a short overview of structure learning literature beyond the basic Gaussian case and ends with suggestions for further research.

6.1 Extension beyond basic GGMs

While the majority of research on graphical modeling has concentrated on Gaussian graphical models, significant efforts have been directed toward extending beyond basic GGMs over the past decade. We provide a brief overview of these extensions here. For comprehensive reviews, works such as Liang & Jia (2023), Maathuis et al. (2019) offer detailed insights. Additionally, these developments merit a dedicated review paper, highlighting their importance in the field.

Beyond Gaussianity assumption: Graphical modeling for non-Gaussian datasets,

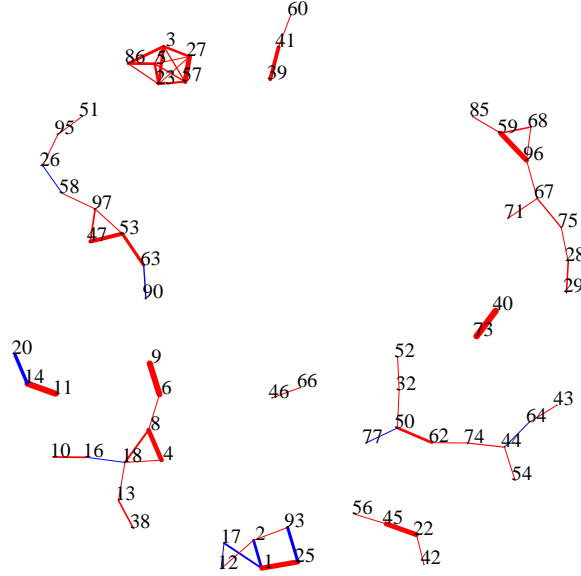


Figure 7: *Gene networks inferred by the BD-A algorithm. Only the 65 edges with the highest edge inclusion probabilities are shown. The width of the edge indicates the edge inclusion probability and varies from narrow (0.58) to thick (1). The color represents the sign of the partial correlation: red (positive) and blue (negative). See supplementary material S4 for the gene names corresponding to the node numbering.*

including types like single nucleotide polymorphisms (SNP), RNA sequencing (RNA-seq), and household income data, is increasingly relevant in data science. Over the past decade, adapting models to handle such non-Gaussian data has been a major research area. For continuous non-Gaussian datasets, like the gene interaction data discussed in Section 5, a two-step method involving a non-parametric data-Gaussianized transformation (Liu et al. 2009) followed by GGM application is effective. This method also extends to multivariate discrete data, as shown in Jia et al. (2017) and (Liang & Jia 2023, Chapter 5). For multivariate discrete or count data, Gaussian copula graphical models offer an alternative, as detailed in Vinciotti et al. (2022), Dobra & Mohammadi (2017), Dobra & Lenkoski (2011). For mixed data types, Bayesian structure learning methods like those in Dobra & Lenkoski (2011), Mohammadi et al. (2017), that use a latent GGM with a copula approach

Hoff (2007) are suitable.

Multiple Gaussian graphical models: Multiple GGMs, as an extension to standard GGMs, are useful for jointly analyzing data from multiple sources (also known as heterogeneous data), *e.g.*, neurological data measured at multiple timescales, or joint neurological, genetic, and phenotypic data. For these types of data, Peterson et al. (2015) use Markov random field prior (Li & Zhang 2010) to model a super-graph linking different graphical models. Tan et al. (2017) uses a logistic regression model to link the connectivity of nodes to covariates specific to each graph. More recently, Li et al. (2019b) developed doubly spike-and-slab mixture priors as a new class of priors for joint estimation of multiple graphical models.

Coloured Gaussian graphical models: these are a class of GGMs with additional symmetry restrictions in the form of equality constraints on the parameters (Højsgaard & Lauritzen 2008). Most of the methods mentioned in Section 3 can be applied to the coloured GGMs. For example, Li et al. (2020) implemented the RJMCMC method by utilizing the colored G-Wishart prior as the Diaconis-Ylvisaker conjugate prior for GGMs. Roverato & Nguyen (2022) applied a stepwise model search procedure to estimate a brain network from fMRI data.

Time-varying GGMs: While extensive literature revolves around learning static graphical time-invariant models, the change of interdependencies with a covariate (*e.g.*, time or space) is often the rule rather than the exception for real-world data, such as friendships between individuals in a social community, communication between genes in a cell, equity trading between companies, and computer network traffic. Bayesian approaches in time-varying GGMs are computationally burdensome with time complexity. Thus, Yu et al. (2022) developed a low-complexity Bayesian approach by imposing temporally dependent spike and slab priors on the graphs such that they are sparse and vary smoothly across time.

Multivariate regressions: GGMs, and sparse graphical models more broadly, are applicable for statistical inference in high-dimensional regression. This application is straightforward when the response variables and explanatory variables are collectively viewed as a set of p -dimensional variables X within our graphical models, as discussed in Section 6 of Lenkoski & Dobra (2011), for example. In such instances, the inference process can address both the selection of relevant variables and the quantification of uncertainty for the regression coefficients, as outlined in the works of Liang et al. (2022, 2013).

6.2 Future Research

In the coming decade, Bayesian methods have the potential to become even faster, expand their theoretical guarantees, grow beyond the Gaussian case, and make more impact with their applications. This section conveys how.

First, new MCMC methods can increase the efficiency and feasible dimension of Bayesian structure learning. Mohammadi et al. (2023b) showed the potential of MCMC methods that move only over the graph space, and not over the space of precision matrices. These methods, however, still only allow graphs to change at most one edge per MCMC iteration. It is an open question whether the balance conditions still hold when allowing changes of multiple edges. If true, this could lead to a significant reduction in computation time. Similarly, Van den Boom et al. (2022a) showed the benefit of the informed proposal and delayed acceptance techniques for the reversible jump approach on the joint space of graphs and precision matrices. It remains to be shown whether methods on the graph space can benefit from the same techniques. Wang (2015) introduced the spike and slab prior in Bayesian structure learning. He circumvented the calculation of the normalizing constant by putting its inverse in the prior of the graph. This leads to a computationally efficient method but creates a challenge when incorporating any knowledge of the graph structure in the prior of the graph. This raises two questions. (i) Could the normalizing constant of the

spike and slab prior be approximated? (ii) Could G-Wishart methods on the joint space benefit from a similar trick? Furthermore, despite the success of the spike and slab method (Wang 2015), there are no MCMC methods with different spike and slab priors. Gan et al. (2019) do use a different spike and slab prior, but then design an expectation-maximization method instead of an MCMC method.

Second, moving away from the standard MCMC approach offers a promising and unexplored perspective to Bayesian structure learning. Dai et al. (2022) argue that Sequential Monte Carlo (SMC) methods remain under-used in statistics, despite several advantages. Van den Boom et al. (2022b) and Tan et al. (2017) show that the SMC approach works for Bayesian structure learning. However, their SMC methods cannot yet compete in terms of computational efficiency with the MCMC methods. Likewise, Nemeth & Fearnhead (2021) outlines the benefits of the stochastic gradient MCMC (SGMCMC) method, and Tan & Friel (2020) designs an SGMCMC method for exponential random graph models. As of now, however, no SGMCMC structure learning method exists. Designing one could advance Bayesian structure learning.

Third, Bayesian structure learning needs more theoretical results. Although the majority of MCMC methods meet the balance conditions, not one of them provides rates of convergence as in Tierney (1994, Section 3.2). Moreover, among the methods in Table 1, only three provide results on the concentration of the posterior around the true value as in Sagar et al. (2024), and only one (Van den Boom et al. 2022b) provides an unbiased estimate. In the future, more theoretical results are essential to the credibility of Bayesian structure learning.

Fourth, recent improvements in structure learning in GGMs could enhance methods beyond the general Gaussian case. The most apparent is the non-Gaussian case, in which the state-of-the-art methods discussed in this paper can be directly applied. This avenue, however, is barely investigated. Similarly, other related fields can benefit from the recent

strides made in the general Gaussian case. They include multiple Gaussian graphical models (Tan et al. 2017), coloured Gaussian graphical models (Li et al. 2020), and graphical models with external network data (Jewson et al. 2022).

Lastly, the increase in the feasible dimension of Bayesian methods will enhance their applications. In Section 5 we reduce the data set containing several thousands of genes to a mere one hundred genes to make the data feasible for computation, potentially losing valuable information. This kind of dimension reduction will be decreasingly necessary as Bayesian methods improve. Likewise, in neuroscience, the models of the brain no longer have to be simplified to just 100 areas as in Dyrba et al. (2020). This could potentially improve the understanding of cognitive diseases like Alzheimer’s. The enhancements in Bayesian structure learning also open up new applications. Especially exciting but yet unexplored examples are graph neural networks and large language models, which make use of the dependency networks among a large number of variables.

SUPPLEMENTARY MATERIAL

Code: all scripts used to perform the simulation study of Section 4 and the case study of 5 can be found on <https://anonymous.4open.science/r/Review-paper-Bayesian-Structure-Learning-in-GGMs-6658>.

Supplementary material S1: the source of the code of all algorithms compared in Section 4.

Supplementary material S2: all algorithm-specific settings for the simulation study of Section 4.

Supplementary material S3: data dictionary for the data used in Section 5.

Supplementary material S4: table to convert the gene numbers of Figure 7 to the names of the genetic transcripts.

References

- Atay-Kayis, A. & Massam, H. (2005), ‘A Monte Carlo method for computing the marginal likelihood in non-decomposable Gaussian graphical models’, *Biometrika* **92**(2), 317–335.
- Banerjee, O., El Ghaoui, L. & d’Aspremont, A. (2008), ‘Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data’, *Journal of Machine Learning Research* **9**, 485–516.
- Banerjee, S. & Ghosal, S. (2015), ‘Bayesian structure learning in graphical models’, *Journal of Multivariate Analysis* **136**, 147–162.
- Bhadra, A. & Mallick, B. (2013), ‘Joint high-dimensional Bayesian variable and covariance selection with an application to eQTL analysis’, *Biometrics* **69**, 447–457.
- Bien, J. & Tibshirani, R. J. (2011), ‘Sparse estimation of a covariance matrix’, *Biometrika* **98**(4), 807–820.
- Carvalho, C. M., Massam, H. & West, M. (2007), ‘Simulation of hyper-inverse Wishart distributions in graphical models’, *Biometrika* **94**(3), 647–659.
- Chandra, N. K., Mueller, P. & Sarkar, A. (2022), ‘Bayesian scalable precision factor analysis for massive sparse Gaussian graphical models’. Unpublished manuscript, arXiv: 2107.11316.
- Cheng, Y. & Lenkoski, A. (2012), ‘Hierarchical Gaussian graphical models: Beyond reversible jump’, *Electronic Journal of Statistics* **6**, 2309 – 2331.
- Dai, C., Heng, J., Jacob, P. E. & Whiteley, N. (2022), ‘An invitation to sequential Monte Carlo samplers’, *Journal of the American Statistical Association* **117**(539), 1587–1600.
- Dawid, A. P. & Lauritzen, S. L. (1993), ‘Hyper Markov laws in the statistical analysis of decomposable graphical models’, *The Annals of Statistics* **21**(3), 1272 – 1317.

- Dempster, A. P. (1972), ‘Covariance selection’, *Biometrics* **28**(1), 157–175.
- Dickey, J. M. (1971), ‘The weighted likelihood ratio, linear hypotheses on normal location parameters’, *The Annals of Mathematical Statistics* **42**(1), 204 – 223.
- Dobra, A. & Lenkoski, A. (2011), ‘Copula Gaussian graphical models and their application to modeling functional disability data’, *The Annals of Applied Statistics* **5**(2), 969–993.
- Dobra, A., Lenkoski, A. & Rodriguez, A. (2011), ‘Bayesian inference for general Gaussian graphical models with application to multivariate lattice data’, *Journal of the American Statistical Association* **106**(496), 1418–1433.
- Dobra, A. & Mohammadi, R. (2017), ‘Loglinear model selection and human mobility’, *The Annals of Applied Statistics* **12**, 815–845.
- Dyrba, M., Mohammadi, R., Grothe, M. J., Kirste, T. & Teipel, S. J. (2020), ‘Gaussian graphical models reveal inter-modal and inter-regional conditional dependencies of brain alterations in alzheimer’s disease’, *Frontiers in Aging Neuroscience* **12**.
- Epskamp, S., Waldorp, L. J., Möttus, R. & Borsboom, D. (2018), ‘The gaussian graphical model in cross-sectional and time-series data’, *Multivariate Behavioral Research* **53**(4), 453–480.
- Fan, J., Feng, Y. & Wu, Y. (2009), ‘Network exploration via the adaptive LASSO and SCAD penalties’, *The Annals of Applied Statistics* **3**(2), 521 – 541.
- Foygel, R. & Drton, M. (2010), ‘Extended Bayesian information criteria for Gaussian graphical models’, *Advances in Neural Information Processing Systems* **23**.
- Friedman, J., Hastie, T. & Tibshirani, R. (2008), ‘Sparse inverse covariance estimation with the graphical lasso’, *Biostatistics* **9**(3), 432–441.

- Gan, L., Narisetty, N. N. & Liang, F. (2019), ‘Bayesian regularization for graphical models with unequal shrinkage’, *Journal of the American Statistical Association* **114**(527), 1218–1231.
- Giudici, P. (1995), ‘Bayes factors for zero partial covariances’, *Journal of Statistical Planning and Inference* **46**(2), 161–174.
- Giudici, P. & Castelo, R. (2003), ‘Improving Markov chain Monte Carlo model search for data mining’, *Machine Learning* **50**, 127–158.
- Giudici, P. & Green, P. J. (1999), ‘Decomposable graphical Gaussian model determination’, *Biometrika* **86**(4), 785–801.
- Glynn, P. W. & Rhee, C.-H. (2014), ‘Exact estimation for Markov chain equilibrium expectations’, *Journal of Applied Probability* **51A**, 377–389.
- Green, P. (1995), ‘Reversible jump Markov chain Monte Carlo computation and Bayesian model determination’, *Biometrika* **82**(4), 711–732.
- Hinne, M., Lenkoski, A., Heskes, T. M. & van Gerven, M. (2014), ‘Efficient sampling of Gaussian graphical models using conditional Bayes factors’, *Stat* **3**, 326 – 336.
- Hoff, P. D. (2007), ‘Extending the rank likelihood for semiparametric copula estimation’, *The Annals of Applied Statistics* **1**(1), 265–283.
- Højsgaard, S. & Lauritzen, S. L. (2008), ‘Graphical Gaussian models with edge and vertex symmetries’, *Journal of the Royal Statistical Society Series B: Statistical Methodology* **70**(5), 1005–1027.
- Jewson, J., Li, L., Battaglia, L., Hansen, S., Rossell, D. & Zwiernik, P. (2022), ‘Graphical model inference with external network data’. Unpublished manuscript, arXiv: 2210.11107.

- Jia, B., Xu, S., Xiao, G., Lamba, V. & Liang, F. (2017), ‘Learning gene regulatory networks from next generation sequencing data’, *Biometrics* **73**(4), 1221–1230.
- Jones, B., Carvalho, C., Dobra, A., Hans, C., Carter, C. & West, M. (2005), ‘Experiments in stochastic computation for high-dimensional graphical models’, *Statistical Science* **20**(4), 388 – 400.
- Khondker, Z., Zhu, H., Chu, H., Lin, W. & Ibrahim, J. (2013), ‘The Bayesian covariance lasso’, *Statistics and Its Interface* **6**, 243–259.
- Kundu, S., Mallick, B. & Baladandayuthapani, V. (2018), ‘Efficient Bayesian regularization for graphical model selection’, *Bayesian Analysis* **14**, 449–476.
- Lam, C. & Fan, J. (2009), ‘Sparsistency and rates of convergence in large covariance matrix estimation’, *Annals of Statistics* **37**, 4254–4278.
- Lauritzen, S. L. (1996), *Graphical Models*, Oxford University Press, Oxford, UK.
- Leday, G. & Richardson, S. (2018), ‘Fast Bayesian inference in large Gaussian graphical models’, *Biometrics* **75**, 1288–1298.
- Lenkoski, A. (2013), ‘A direct sampler for G-Wishart variates’, *Stat* **2**(1), 119–128.
- Lenkoski, A. & Dobra, A. (2011), ‘Computational aspects related to inference in Gaussian graphical models with the G-Wishart prior’, *Journal of Computational and Graphical Statistics* **20**(1), 140–157.
- Leppä-aho, J., Pensar, J., Roos, T. & Corander, J. (2017), ‘Learning Gaussian graphical models with fractional marginal pseudo-likelihood’, *International Journal of Approximate Reasoning* **83**, 21–42.

- Li, F. & Zhang, N. R. (2010), ‘Bayesian variable selection in structured high-dimensional covariate spaces with applications in genomics’, *Journal of the American Statistical Association* **105**(491), 1202–1214.
- Li, F. & Zhang, X. (2017), ‘Bayesian lasso with neighborhood regression method for Gaussian graphical model’, *Acta Mathematicae Applicatae Sinica, English Series* **33**, 485–496.
- Li, Q., Gao, X. & Massam, H. (2020), ‘Bayesian model selection approach for coloured graphical Gaussian models’, *Journal of Statistical Computation and Simulation* **90**(14), 2631–2654.
- Li, Y., Craig, B. A. & Bhadra, A. (2019a), ‘The graphical horseshoe estimator for inverse covariance matrices’, *Journal of Computational and Graphical Statistics* **28**(3), 747–757.
- Li, Z., McCormick, T. & Clark, S. (2019b), Bayesian joint spike-and-slab graphical lasso, in ‘International Conference on Machine Learning’, PMLR, pp. 3877–3885.
- Li, Z. R. & McCormick, T. H. (2019), ‘An expectation conditional maximization approach for Gaussian graphical models’, *Journal of Computational and Graphical Statistics* **28**(4), 767–777.
- Liang, F. & Jia, B. (2023), *Sparse Graphical Modeling for High Dimensional Data: A Paradigm of Conditional Independence Tests*.
- Liang, F., Song, Q. & Yu, K. (2013), ‘Bayesian subset modeling for high-dimensional generalized linear models’, *Journal of the American Statistical Association* **108**(502), 589–606.
- Liang, F., Xue, J. & Jia, B. (2022), ‘Markov neighborhood regression for high-dimensional inference’, *Journal of the American Statistical Association* **117**(539), 1200–1214.
- Liu, H., Han, F., Yuan, M., Lafferty, J. & Wasserman, L. (2012), ‘The nonparanormal

- skeptic’, *Proceedings of the 29th International Conference on Machine Learning, ICML 2012* **2**, 1415–1422.
- Liu, H., Lafferty, J. & Wasserman, L. (2009), ‘The nonparanormal: Semiparametric estimation of high dimensional undirected graphs’, *Journal of Machine Learning Research* **10**(80), 2295–2328.
- Liu, H. & Wang, L. (2017), ‘TIGER: A tuning-insensitive approach for optimally estimating Gaussian graphical models’, *Electronic Journal of Statistics* **11**(1), 241 – 294.
- Maathuis, M., Drton, M., Lauritzen, S. & Wainwright, M. (2019), *Handbook of Graphical Models*, CRC Press, Boca Raton, Florida.
- Meinshausen, N. & Bühlmann, P. (2006), ‘High-dimensional graphs and variable selection with the lasso’, *The Annals of Statistics* **34**(3), 1436 –1462.
- Mohammadi, A., Abegaz, F., van den Heuvel, E. & Wit, E. (2017), ‘Bayesian modelling of dupuytren disease by using Gaussian copula graphical models’, *Journal of the Royal Statistical Society. Series C: Applied Statistics* **66**(3), 629–645.
- Mohammadi, A. & Wit, E. C. (2015), ‘Bayesian structure learning in sparse Gaussian graphical models’, *Bayesian Analysis* **10**(1), 109 – 138.
- Mohammadi, R. (2022), *ssgraph: Bayesian graph structure learning using spike-and-slab priors*. R package version 1.15.
- Mohammadi, R., Massam, H. & Letac, G. (2023a), ‘Accelerating bayesian structure learning in sparse gaussian graphical models’, *Journal of the American Statistical Association* **118**(542), 1345–1358.
- Mohammadi, R., Schoonhoven, M., Vogels, L. & Birbil, I. S. (2023b), ‘High-dimensional

- Bayesian structure learning in Gaussian graphical models using marginal pseudo-likelihood’. Unpublished manuscript , arXiv:2307.00127.
- Mohammadi, R. & Wit, E. C. (2019), ‘Bdgraph: An R package for Bayesian structure learning in graphical models’, *Journal of Statistical Software* **89**(3), 1–30.
- Mohammadi, R., Wit, E. & Dobra, A. (2022), *BDgraph: Bayesian structure learning in graphical models using birth-death MCMC*. R package version 2.72.
- Nemeth, C. & Fearnhead, P. (2021), ‘Stochastic gradient Markov chain Monte Carlo’, *Journal of the American Statistical Association* **116**(533), 433–450.
- Park, T. & Casella, G. (2008), ‘The Bayesian lasso’, *Journal of the American Statistical Association* **103**(482), 681–686.
- Peterson, C., Stingo, F. C. & Vannucci, M. (2015), ‘Bayesian inference of multiple gaussian graphical models’, *Journal of the American Statistical Association* **110**(509), 159–174.
- Ravikumar, P., Wainwright, M., Raskutti, G. & Yu, B. (2011), ‘High-dimensional covariance estimation by minimizing l1-penalized log-determinant divergence’, *Electronic Journal of Statistics* **5**, 935–980.
- Rothman, A., Bickel, P., Levina, E. & Zhu, J. (2008), ‘Sparse permutation invariant covariance’, *Electronic Journal of Statistics* **2**, 494–515.
- Roverato, A. (2002), ‘Hyper inverse Wishart distribution for non-decomposable graphs and its application to Bayesian inference for Gaussian graphical models’, *Scandinavian Journal of Statistics* **29**(3), 391–411.
- Roverato, A. & Nguyen, D. N. (2022), Model inclusion lattice of coloured gaussian graphical models for paired data, in ‘International Conference on Probabilistic Graphical Models’, PMLR, pp. 133–144.

- Sagar, K., Banerjee, S., Datta, J. & Bhadra, A. (2024), ‘Precision matrix estimation under the horseshoe-like prior–penalty dual’, *Electronic Journal of Statistics* **18**, 1–46.
- Sagar, K., Datta, J., Banerjee, S. & Bhadra, A. (2023), ‘Maximum a posteriori estimation in graphical models using local linear approximation’. Unpublished manuscript, arXiv:2104.10750.
- Smith, J., Arashi, M. & Bekker, A. (2023a), *baygel: Bayesian shrinkage estimators for precision matrices in Gaussian graphical models*. R package version 0.3.0.
- Smith, J., Arashi, M. & Bekker, A. (2023b), ‘A data driven Bayesian graphical ridge estimator’. Unpublished manuscript, arXiv:2210.16290.
- Stingo, F. & Marchetti, G. (2014), ‘Efficient local updates for undirected graphical models’, *Statistics and Computing* **25**, 159–171.
- Stranger, B. E., Nica, A. C., Forrest, M. S., Dimas, A., Bird, C. P., Beazley, C., Ingle, C. E., Dunning, M., Flicek, P., Koller, D., Montgomery, S., Tavaré, S., Deloukas, P. & Dermitzakis, E. T. (2007), ‘Population genomics of human gene expression’, *Nature Genetics* **39**, 1217–1224.
- Sun, T. & Zhang, C.-H. (2012), ‘Sparse matrix inversion with scaled lasso’, *Journal of Machine Learning Research* **14**, 3385–3418.
- Tan, L. S. L. & Friel, N. (2020), ‘Bayesian variational inference for exponential random graph models’, *Journal of Computational and Graphical Statistics* **29**(4), 910–928.
- Tan, L. S. L., Jasra, A., Iorio, M. D. & Ebbels, T. M. D. (2017), ‘Bayesian inference for multiple Gaussian graphical models with application to metabolic association networks’, *The Annals of Applied Statistics* **11**(4), 2222 – 2251.

- Tibshirani, R. (1996), ‘Regression shrinkage and selection via the lasso’, *Journal of the Royal Statistical Society. Series B (Methodological)* **58**(1), 267–288.
- Tierney, L. (1994), ‘Markov chains for exploring posterior distributions’, *The Annals of Statistics* **22**(4), 1701 – 1728.
- Van den Boom, W., Beskos, A. & De Iorio, M. (2022a), ‘The G-Wishart weighted proposal algorithm: Efficient posterior computation for Gaussian graphical models’, *Journal of Computational and Graphical Statistics* **31**(4), 1215–1224.
- Van den Boom, W., Jasra, A., De Iorio, M., Beskos, A. & Eriksson, J. (2022b), ‘Unbiased approximation of posteriors via coupled particle Markov chain Monte Carlo’, *Statistics and Computing* **32**(3), 1–19.
- Vandenberghe, L., Boyd, S. & Wu, S.-P. (1998), ‘Determinant maximization with linear matrix inequality constraints’, *SIAM Journal on Matrix Analysis and Applications* **19**(2), 499–533.
- Vinciotti, V., Behrouzi, P. & Mohammadi, R. (2022), ‘Bayesian inference of microbiota systems from count metagenomic data’. Unpublished manuscript , arXiv:2307.00127.
- Waldorp, L. & Marsman, M. (2022), ‘Relations between networks, regression, partial correlation, and the latent variable model’, *Multivariate Behavioral Research* **57**(6), 994–1006.
- Wang, H. (2012), ‘Bayesian graphical lasso models and efficient posterior computation’, *Bayesian Analysis* **7**(4), 867 – 886.
- Wang, H. (2015), ‘Scaling it up: Stochastic search structure learning in graphical models’, *Bayesian Analysis* **10**(2), 351–377.
- Wang, H. & Li, S. Z. (2012), ‘Efficient Gaussian graphical model determination under G-Wishart prior distributions’, *Electronic Journal of Statistics* **6**, 168–198.

- Williams, D., Piironen, J., Vehtari, A. & Rast, P. (2018), ‘Bayesian estimation of Gaussian graphical models with projection predictive selection’. Unpublished manuscript, arXiv:1801.05725.
- Williams, D. R. & Mulder, J. (2019), *BGGM: an R package for Bayesian Gaussian graphical models*. R package version 2.0.0.
- Williams, D. R. & Mulder, J. (2020), ‘Bayesian hypothesis testing for Gaussian graphical models: Conditional independence and order constraints’, *Journal of Mathematical Psychology* **99**.
- Yu, H., Wu, S. & Dauwels, J. (2022), ‘Efficient variational Bayes learning of graphical models with smooth structural changes’, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **45**(1), 475–488.
- Yuan, M. (2010), ‘High dimensional inverse covariance matrix estimation via linear programming.’, *Journal of Machine Learning Research* **11**, 2261–2286.
- Yuan, M. & Lin, Y. (2007), ‘Model selection and estimation in the Gaussian graphical model’, *Biometrika* **94**(1), 19–35.
- Zhao, T., Liu, H., Roeder, K., Lafferty, J. & Wasserman, L. (2012), ‘The huge package for high-dimensional undirected graph estimation in R’, *Journal of Machine Learning Research* **13**, 1059–1062.