# BDgraph: An **R** Package for Bayesian Structure Learning in Graphical Models

**Reza Mohammadi**
University of Amsterdam

**Ernst C. Wit**
Universita della Svizzera Italiana

## Abstract

Graphical models provide powerful tools to uncover complicated patterns in multivariate data and are commonly used in Bayesian statistics and machine learning. In this paper, we introduce an R package **BDgraph** which performs Bayesian structure learning for general undirected graphical models (decomposable and non-decomposable) with continuous, discrete, and mixed variables. The package efficiently implements recent improvements in the Bayesian literature, including that of Mohammadi and Wit (2015), Mohammadi, Abegaz Yazew, van den Heuvel, and Wit (2017a), and Dobra and Mohammadi (2018). To speed up computations, the computationally intensive tasks have been implemented in C++ and interfaced with R. In addition, the package contains several functions for simulation and visualization, as well as two multivariate datasets taken from the literature and are used to describe the package capabilities. The paper includes a brief overview of the statistical methods which have been implemented in the package. The main body of the paper explains how to use the package. Furthermore, we illustrate the package's functionality in both real and artificial examples, as well as in an extensive simulation study.

*Keywords*: Bayesian structure learning, Gaussian graphical models, Gaussian copula, Covariance selection, Birth-death process, Markov chain Monte Carlo, G-Wishart, **BDgraph**, R.

## 1. Introduction

Graphical models (Lauritzen 1996) are commonly used, particularly in Bayesian statistics and machine learning, to describe the conditional independence relationships among variables in multivariate data. In graphical models, each random variable is associated with a node in a graph and links represent conditional dependency between variables, whereas the absence of a link implies that the variables are independent conditional on the rest of the variables (the pairwise Markov property).

In recent years, significant progress has been made in designing efficient algorithms to discover graph structures from multivariate data (Dobra, Lenkoski, and Rodriguez 2011; Dobra and Lenkoski 2011; Jones, Carvalho, Dobra, Hans, Carter, and West 2005; Dobra and Mohammadi 2018; Mohammadi and Wit 2015; Mohammadi *et al.* 2017a; Friedman, Hastie, and Tibshirani 2008; Meinshausen and Buhlmann 2006; Murray and Ghahramani 2004; Pensar, Nyman, Niiranen, Corander *et al.* 2017; Rolfs, Rajaratnam, Guillot, Wong, and Maleki 2012; Wit and Abbruzzo 2015a,b). Bayesian approaches provide a principled alternative to various penalized approaches.

In this paper, we describe the **BDgraph** package (Mohammadi and Wit 2018) in R (R Core Team 2013) for Bayesian structure learning in undirected graphical models. The package can deal with Gaussian, non-Gaussian, discrete and mixed datasets. The package includes various functional modules, including data generation for simulation, several search algorithms, graph estimation routines, a convergence check and a visualization tool; see Figure 1. Our package efficiently implements recent improvements in the Bayesian literature, including those of Mohammadi and Wit (2015); Mohammadi *et al.* (2017a); Lenkoski (2013); Uhler, Lenkoski, and Richards (2014); Dobra and Lenkoski (2011); Hoff (2007). For a Bayesian framework of Gaussian graphical models, we implement the method developed by Mohammadi and Wit (2015) and for Gaussian copula graphical models we use the method described by Mohammadi *et al.* (2017a) and Dobra and Lenkoski (2011). To make our Bayesian methods computationally feasible for moderately high-dimensional data, we efficiently implement the **BDgraph** package in C++ linked to R. To make the package easy to use, the **BDgraph** package uses several S3 classes as return values of its functions. The package is available under the general public license (GPL ≥ 3) from the Comprehensive R Archive Network (CRAN) at http://cran.r-project.org/packages=BDgraph.

In the Bayesian literature, the **BDgraph** is the only R package which is available online for Gaussian graphical models and Gaussian copula graphical models. On the other hand, in frequentist literature, the existing packages are **huge** (Zhao, Liu, Roeder, Lafferty, and Wasserman 2014), **glasso** (Friedman, Hastie, and Tibshirani 2014), **bnlearn** (Scutari 2010), **pcalg** (Kalisch, Mächler, Colombo, Maathuis, and Bühlmann 2012), **ssgraph** (Mohammadi 2018), **netgwas** (Behrouzi and Wit 2017), and **QUIC** (Hsieh, Sustik, Dhillon, and Ravikumar 2011, 2014).

In Section 2 we illustrate the user interface of the **BDgraph** package. In Section 3 we explain some methodological background of the package. In this regard, in Section 3.1 we briefly explain the Bayesian framework for Gaussian graphical models for continuous data. In Section 3.2 we briefly describe the Bayesian framework in the Gaussian copula graphical models for data that do not follow the Gaussianity assumption, such as non-Gaussian continuous, discrete or mixed data. In Section 4 we describe the main functions implemented in the **BDgraph** package. In addition, we explain the user interface and the performance of the package by a simple simulation example. Section 5 includes an extensive simulation study to evaluate the performance of the Bayesian methods implemented in the **BDgraph**, as well as to compare them with alternative approaches which are available in the R package **huge**. In Section 6, using the functions implemented in the **BDgraph** package, we study two actual datasets.

# 2. User interface

In the R environment, one can access and load the **BDgraph** package by using the following commands:

```
R> install.packages( "BDgraph" )
R> library( "BDgraph" )
```

By loading the package we automatically load the **igraph** (Csardi and Nepusz 2006) and **Matrix** (Bates and Maechler 2014) packages, since the **BDgraph** package depends on these two packages. These packages are available on the Comprehensive R Archive Network (CRAN) at http://CRAN.R-project.org. We use the **igraph** package for graph visualization and the **Matrix** package for memory-optimization, using the sparse matrix output.

To speed up computations, we efficiently implement the **BDgraph** package by linking the C++ code to R. The computationally extensive tasks of the package are implemented in parallel in C++ using **OpenMP** (OpenMP Architecture Review Board 2008). For the C++ code, we use the highly optimized **LAPACK** (Anderson, Bai, Bischof, Blackford, Demmel, Dongarra, Du Croz, Greenbaum, Hammarling, McKenney, and Sorensen 1999) and **BLAS** (Lawson, Hanson, Kincaid, and Krogh 1979) linear algebra libraries on systems that provide them. The use of these libraries significantly improves program speed.

We design the **BDgraph** package to provide a Bayesian framework for undirected graph estimation of different types of datasets such as continuous, discrete or mixed data. The package facilitates a pipeline for analysis by three functional modules; see Figure 1. These modules are as follows:
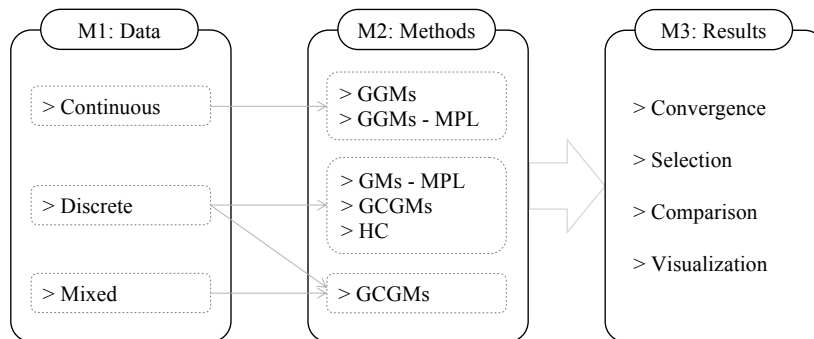


Figure 1: Configuration of the **BDgraph** package which includes three main parts: (M1) data generation, (M2) several search algorithms, (M3) various functions to evaluate convergence of the search algorithms, estimation of the true graph, assessment and comparison of the results and graph visualization.

**Module 1. Data simulation:** Function `bdgraph.sim` simulates multivariate Gaussian, discrete and mixed data with different undirected graph structures, including "random", "cluster", "scale-free", "hub", "circle", "AR(1)", "AR(1)" and "fixed" graphs. Users can determine the sparsity of the graph structure and can generate mixed data, including "count", "ordinal", "binary", "Gaussian" and "non-Gaussian" variables.

**Module 2. Method:** The function `bdgraph` provides two estimation methods with two different algorithms for sampling from the posterior distributions:

1. Graph estimation for the multivariate data that follow the Gaussianity assumption, based on the Gaussian graphical models (GGMs), and using the birth-death MCMC sampling algorithm described in Mohammadi and Wit (2015).
2. Graph estimation for non-Gaussian, discrete, and mixed data, based on Gaussian copula graphical models (GCGMs), and using the birth-death MCMC sampling algorithm described in Mohammadi *et al.* (2017a).

Note, in the `bdgraph` function, besides the BDMCMC algorithms, we implemented the Reversible Jump MCMC (RJMCMC) algorithms (Green 1995), as an alternative.

**Module 3. Result:** Includes four types of functions:

- `Graph selection`: The functions `select`, `plinks`, and `pgraph` provide the selected graph, the posterior link inclusion probabilities and the posterior probability of each graph, respectively.
- `Convergence check`: The functions `plotcoda` and `traceplot` provide several visualization plots to monitor the convergence of the sampling algorithms.
- `Comparison and goodness-of-fit`: The functions `compare` and `plotroc` provide several comparison measures and an ROC plot for model comparison.
- `Visualization`: The plotting functions `plot.bdgraph` and `plot.sim` provide visualizations of the simulated data and estimated graphs.

# 3. Methodological background

In Section 3.1 we briefly explain the Gaussian graphical model for multivariate data. Then we illustrate the birth-death MCMC algorithm for sampling from the joint posterior distribution over Gaussian graphical models; for more details see Mohammadi and Wit (2015). In Section 3.2 we briefly describe the Gaussian copula graphical model (Dobra and Lenkoski 2011), which can deal with non-Gaussian, discrete or mixed data. Then we explain the birth-death MCMC algorithm which is designed for the Gaussian copula graphical models; for more details see Mohammadi *et al.* (2017a).

## 3.1. Bayesian Gaussian graphical models

In graphical models, each random variable is associated with a node and conditional dependence relationships among random variables are presented as a graph $G = (V, E)$ in which $V = \{1, 2, ..., p\}$ specifies a set of nodes and a set of existing links $E \subset V \times V$ (Lauritzen 1996). Our focus here is on undirected graphs, in which $(i, j) \in E \Leftrightarrow (j, i) \in E$. The absence of a link between two nodes specifies the pairwise conditional independence of those two variables given the remaining variables, while a link between two variables determines their conditional dependence.

In Gaussian graphical models (GGMs), we assume that the observed data follow multivariate Gaussian distribution $\mathcal{N}_p(\mu, K^{-1})$. Here we assume $\mu = 0$. Let $\mathbf{Z} = (Z^{(1)}, ..., Z^{(n)})^\top$ be the observed data of $n$ independent samples, then the likelihood function is

$$Pr(\mathbf{Z}|K, G) \propto |K|^{n/2} \exp\left\{-\frac{1}{2}\text{tr}(KU)\right\}, \tag{1}$$

where $U = \mathbf{Z}^\top \mathbf{Z}$.

In GGMs, conditional independence is implied by the form of the precision matrix. Based on the pairwise Markov property, variables $i$ and $j$ are conditionally independent given the remaining variables, if and only if $K_{ij} = 0$. This property implies that the links in graph $G = (V, E)$ correspond with the nonzero elements of the precision matrix $K$; this means that $E = \{(i, j)|K_{ij} \neq 0\}$. Given graph $G$, the precision matrix $K$ is constrained to the cone $\mathbb{P}_G$ of symmetric positive definite matrices with elements $K_{ij}$ equal to zero for all $(i, j) \notin E$.

We consider the G-Wishart distribution $W_G(b, D)$ to be a prior distribution for the precision matrix $K$ with density

$$Pr(K|G) = \frac{1}{I_G(b, D)} |K|^{(b-2)/2} \exp \left\{ -\frac{1}{2} \mathrm{tr}(DK) \right\} 1(K \in \mathbb{P}_G), \tag{2}$$

where $b > 2$ is the degrees of freedom, $D$ is a symmetric positive definite matrix, $I_G(b, D)$ is the normalizing constant with respect to the graph $G$ and $1(x)$ evaluates to 1 if $x$ holds, and otherwise to 0. The G-Wishart distribution is a well-known prior for the precision matrix, since it represents the conjugate prior for multivariate Gaussian data as in (1).

For full graphs, the G-Wishart distribution reduces to the standard Wishart distribution, hence the normalizing constant has an explicit form (Muirhead 1982). Also, for decomposable graphs, the normalizing constant has an explicit form (Roverato 2002); however, for non-decomposable graphs, it does not. In that case it can be estimated by using the Monte Carlo method (Atay-Kayis and Massam 2005) or the Laplace approximation (Lenkoski and Dobra 2011). In the **BDgraph** package, we design the `gnorm` function to estimate the log of the normalizing constant by using the Monte Carlo method proposed Atay-Kayis and Massam (2005).

Since the G-Wishart prior is a conjugate prior to the likelihood (1), the posterior distribution of $K$ is

$$Pr(K|\mathbf{Z}, G) = \frac{1}{I_G(b^*, D^*)} |K|^{(b^*-2)/2} \exp \left\{ -\frac{1}{2} \mathrm{tr}(D^*K) \right\},$$

where $b^* = b + n$ and $D^* = D + S$, that is, $W_G(b^*, D^*)$.

*Direct sampler from G-Wishart*

Several sampling methods from the G-Wishart distribution have been proposed; to review existing methods see Wang and Li (2012). More recently, Lenkoski (2013) has developed an exact sampling algorithm for the G-Wishart distribution, borrowing an idea from Hastie, Tibshirani, and Friedman (2009).

In the **BDgraph** package, we use Algorithm 1 to sample from the posterior distribution of the precision matrix. We implement the algorithm in the package as a function `rgwish`; see the R code below for illustration.

```
R> adj.g <- matrix( c( 0, 0, 1, 0, 0, 0, 1, 0, 0 ), 3, 3 )
R> adj.g

     [,1] [,2] [,3]
[1,]    0    0    1
[2,]    0    0    0
[3,]    1    0    0
```

---

**Algorithm 1** . Exact sampling from the precision matrix

---

**Input:** A graph $G = (V, E)$ with precision matrix $K$ and $\Sigma = K^{-1}$

**Output:** An exact sample from the precision matrix.

1: Set $\Omega = \Sigma$
2: **repeat**
3:    **for** $i = 1, ..., p$ **do**
4:       Let $N_i \subset V$ be the neighbor set of node $i$ in $G$. Form $\Omega_{N_i}$ and $\Sigma_{N_i,i}$ and solve
      $\hat{\beta}_i^* = \Omega_{N_i}^{-1} \Sigma_{N_i,i}$
5:       Form $\hat{\beta}_i \in R^{p-1}$ by padding the elements of $\hat{\beta}_i^*$ to the appropriate locations and zeros
      in those locations not connected to $i$ in $G$
6:       Update $\Omega_{i,-i}$ and $\Omega_{-i,i}$ with $\Omega_{-i,-i}\hat{\beta}_i$
7:    **end for**
8: **until** convergence
9: **return** $K = \Omega^{-1}$

---

```
R> sample <- rgwish( n = 1, adj.g = adj.g, b = 3, D = diag(4) )
R> round( sample, 2 )


      [,1] [,2] [,3]
[1,] 1.30  0.0 0.54
[2,] 0.00  0.6 0.00
[3,] 0.54  0.0 2.96
```

This matrix is a sample from a G-Wishart distribution with $b = 3$ and $D = I_4$ as an identity matrix and a graph structure with adjacency matrix `adj.g`.

### BDMCMC algorithm for GGMs

Consider the joint posterior distribution of the graph $G$ and the precision matrix $K$ given by

$$Pr(K, G \mid \mathbf{Z}) \propto Pr(\mathbf{Z} \mid K) \; Pr(K \mid G) \; Pr(G). \tag{3}$$

For the graph-prior, we consider a uniform distribution over all graph space, $pr(G) \propto 1$, as a non-informative prior. For the prior distribution of the precision matrix conditional on the graph $G$, we use a G-Wishart $W_G(b, D)$.

Here we consider a computationally efficient birth-death MCMC sampling algorithm proposed by Mohammadi and Wit (2015) for Gaussian graphical models. The algorithm is based on a continuous time birth-death Markov process, in which the algorithm explores the graph space by adding/removing a link in a birth/death event.

In the birth-death process, for a particular pair of graph $G = (V, E)$ and precision matrix $K$, each link dies independently of the rest as a Poisson process with death rate $\delta_e(K)$. Since the links are independent, the overall death rate is $\delta(K) = \sum_{e \in E} \delta_e(K)$. Birth rates $\beta_e(K)$ for $e \notin E$ are defined similarly. Thus the overall birth rate is $\beta(K) = \sum_{e \notin E} \beta_e(K)$.

Since the birth and death events are independent Poisson processes, the time between two successive events is exponentially distributed with mean $1/(\beta(K) + \delta(K))$. The time between

successive events can be considered as inverse support for any particular instance of the state $(G, K)$. The probabilities of birth and death events are

$$Pr(\text{birth of link } e) = \frac{\beta_e(K)}{\beta(K) + \delta(K)}, \qquad \text{for each } e \notin E, \tag{4}$$

$$Pr(\text{death of link } e) = \frac{\delta_e(K)}{\beta(K) + \delta(K)}, \qquad \text{for each } e \in E. \tag{5}$$

The birth and death rates of links occur in continuous time with the rates determined by the stationary distribution of the process. The BDMCMC algorithm is designed in such a way that the stationary distribution is equal to the target joint posterior distribution of the graph and the precision matrix (3).

Mohammadi and Wit (2015, Theorem 3.1) derived a condition that guarantees the above birth and death process converges to our target joint posterior distribution (3). By following their Theorem we define the birth and death rates, as below

$$\beta_e(K) = \min\left\{\frac{Pr(G^{+e}, K^{+e}|\mathbf{Z})}{Pr(G, K|\mathbf{Z})}, 1\right\}, \quad \text{for each } e \notin E, \tag{6}$$

$$\delta_e(K) = \min\left\{\frac{Pr(G^{-e}, K^{-e}|\mathbf{Z})}{Pr(G, K|\mathbf{Z})}, 1\right\}, \quad \text{for each } e \in E, \tag{7}$$

in which $G^{+e} = (V, E \cup \{e\})$ and $K^{+e} \in \mathbb{P}_{G^{+e}}$ and similarly $G^{-e} = (V, E \setminus \{e\})$ and $K^{-e} \in \mathbb{P}_{G^{-e}}$. For computation part related to the ratio of posterior see Mohammadi, Massam, and Gerald (2017b).

Algorithm 2 provides the pseudo-code for our BDMCMC sampling scheme which is based on the above birth and death rates.

---
**Algorithm 2** . BDMCMC algorithm for GGMs
---
**Input:** A graph $G = (V, E)$ and a precision matrix $K$.
**Output:** Samples from the joint posterior distribution of $(G, K)$, (3), and waiting times.
 1: **for** N iteration **do**
 2:     **1. Sample from the graph.** Based on birth and death process
 3:        1.1. Calculate the birth rates by (6) and $\beta(K) = \sum_{e \notin E} \beta_e(K)$
 4:        1.2. Calculate the death rates by (7) and $\delta(K) = \sum_{e \in E} \delta_e(K)$
 5:        1.3. Calculate the waiting time by $W(K) = 1/(\beta(K) + \delta(K))$
 6:        1.4. Simulate the type of jump (birth or death) by (4) and (5)
 7:     **2. Sample from the precision matrix.** By using Algorithm 1.
 8: **end for**
---

Note, step 1 of the algorithm is suitable for parallel computation. In the **BDgraph**, we implement this step of algorithm in parallel using **OpenMP** in C++ to speed up the computations.

The BDMCMC sampling algorithm is designed in such a way that a sample $(G, K)$ is obtained at certain jump moments, $\{t_1, t_2, ...\}$ (see Figure 2). For efficient posterior inference of the parameters, we use the Rao-Blackwellized estimator, which is an efficient estimator for continuous time MCMC algorithms (Cappé, Robert, and Rydén 2003, Section 2.5). By using
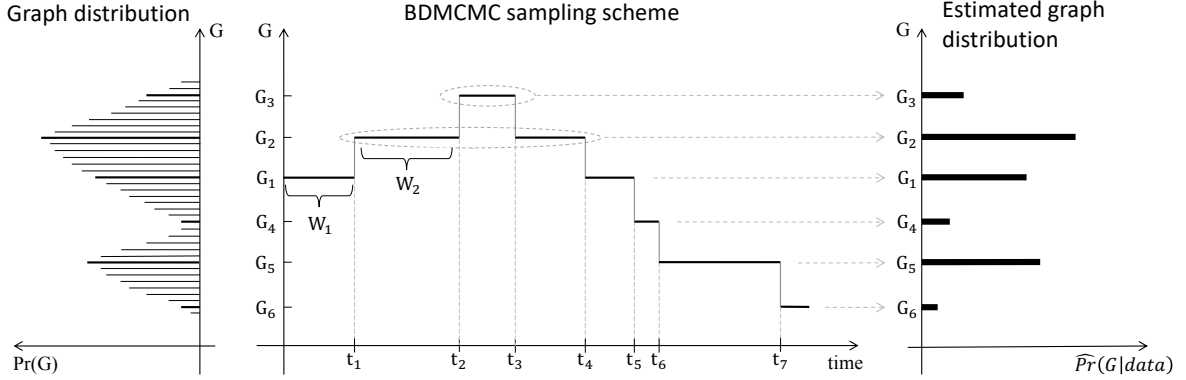
Figure 2:    This image visualizes the Algorithm 2. The left side shows the true posterior distribution of the graph. The middle panel presents a continuous time BDMCMC sampling algorithm where $\{W_1, W_2, ...\}$ denote waiting times and $\{t_1, t_2, ...\}$ denote jumping times. The right side denotes the estimated posterior probability of the graphs in proportion to the total of their waiting times, according to the Rao-Blackwellized estimator.

the Rao-Blackwellized estimator, for example, one can estimate the posterior distribution of the graphs proportional to the total waiting times of each graph.

### 3.2. Gaussian copula graphical models

In practice we encounter both discrete and continuous variables; Gaussian copula graphical modelling has been proposed by Dobra and Lenkoski (2011) to describe dependencies between such heterogeneous variables. Let $\mathbf{Y}$ (as observed data) be a collection of continuous, binary, ordinal or count variables with the marginal distribution $F_j$ of $Y_j$ and $F_j^{-1}$ as its pseudo inverse. For constructing a joint distribution of $\mathbf{Y}$, we introduce a multivariate Gaussian latent variable as follows:

$$Z_1, ..., Z_n \overset{iid}{\sim} \mathcal{N}_p(0, \Gamma(K)),$$
$$Y_{ij} = F_j^{-1}(\Phi(Z_{ij})), \tag{8}$$

where $\Gamma(K)$ is the correlation matrix for a given precision matrix $K$. The joint distribution of $\mathbf{Y}$ is given by

$$Pr\left(Y_1 \leq Y_1, \ldots, Y_p \leq Y_p\right) = C(F_1(Y_1), \ldots, F_p(Y_p) \mid \Gamma(K)), \tag{9}$$

where $C(\cdot)$ is the Gaussian copula given by

$$C(u_1, \ldots, u_p \mid \Gamma) = \Phi_p\left(\Phi^{-1}(u_1), \ldots, \Phi^{-1}(u_p) \mid \Gamma\right),$$

with $u_v = F_v(Y_v)$ and $\Phi_p(\cdot)$ is the cumulative distribution of multivariate Gaussian and $\Phi(\cdot)$ is the cumulative distribution of univariate Gaussian distributions. It follows that $Y_v = F_v^{-1}(\Phi(Z_v))$ for $v = 1, ..., p$. If all variables are continuous then the margins are unique; thus zeros in $K$ imply conditional independence, as in Gaussian graphical models (Hoff 2007; Abegaz and Wit 2015). For discrete variables, the margins are not unique but still well-defined (Nelsen 2007).

In semiparametric copula estimation, the marginals are treated as nuisance parameters and estimated by the rescaled empirical distribution. The joint distribution in (9) is then parametrized only by the correlation matrix of the Gaussian copula. We are interested to infer the underlying graph structure of the observed variables $\mathbf{Y}$ implied by the continuous latent variables $\mathbf{Z}$. Since $\mathbf{Z}$ are unobservable we follow the idea of Hoff (2007) of associating them with the observed data as below.

Given the observed data $\mathbf{Y}$ from a sample of $n$ observations, we constrain the samples from latent variables $\mathbf{Z}$ to belong to the set

$$\mathcal{D}(\mathbf{Y}) = \{\mathbf{Z} \in \mathbb{R}^{n \times p} : L_j^r(\mathbf{Z}) < z_j^{(r)} < U_j^r(\mathbf{Z}), r = 1, \ldots, n; j = 1, \ldots, p\},$$

where

$$L_j^r(\mathbf{Z}) = \max \left\{ Z_j^{(s)} : Y_j^{(s)} < Y_j^{(r)} \right\} \text{ and } U_j^r(\mathbf{Z}) = \min \left\{ Z_j^{(s)} : Y_j^{(r)} < Y_j^{(s)} \right\}. \tag{10}$$

Following Hoff (2007) we infer the latent space by substituting the observed data $\mathbf{Y}$ with the event $\mathcal{D}(\mathbf{Y})$ and define the likelihood as

$$Pr(\mathbf{Y} \mid K, G, F_1, ..., F_p) = Pr(\mathbf{Z} \in \mathcal{D}(\mathbf{Y}) \mid K, G) \; Pr(\mathbf{Y} \mid \mathbf{Z} \in \mathcal{D}(\mathbf{Y}), K, G, F_1, ..., F_p).$$

The only part of the observed data likelihood relevant for inference on $K$ is $Pr(\mathbf{Z} \in \mathcal{D}(\mathbf{Y}) \mid K, G)$. Thus, the likelihood function is given by

$$Pr(\mathbf{Z} \in \mathcal{D}(\mathbf{Y}) \mid K, G) = Pr(\mathbf{Z} \in \mathcal{D}(\mathbf{Y}) \mid K, G) = \int_{\mathcal{D}(\mathbf{Y})} Pr(\mathbf{Z} \mid K, G) dZ \tag{11}$$

where $Pr(\mathbf{Z} \mid K, G)$ is defined in (1).

### BDMCMC algorithm for GCGMs

The joint posterior distribution of the graph $G$ and precision matrix $K$ for the GCGMs is

$$Pr(K, G | Z \in \mathcal{D}(\mathbf{Y})) \propto Pr(K, G) Pr(Z \in \mathcal{D}(\mathbf{Y}) | K, G). \tag{12}$$

Sampling from this posterior distribution can be done by using the birth-death MCMC algorithm. Mohammadi *et al.* (2017a) have developed and extended the birth-death MCMC algorithm to more general cases of GCGMs. We summarize their algorithm as follows:

In step 1, the latent variables $\mathbf{Z}$ are sampled conditional on the observed data $\mathbf{Y}$. The other steps are the same as in Algorithm 2.

**Remark:** in cases where all variables are continuous, we do not need to sample from latent variables in each iteration of Algorithm 2, since all margins in the Gaussian copula are unique. Thus, for these cases, we transfer our non-Gaussian data to Gaussian, and then we run Algorithm 2; see example 6.2.

### Alternative RJMCMC algorithm

RJMCMC is a special case of the trans-dimensional MCMC methodology Green (2003). The RJMCMC approach is based on an ergodic discrete-time Markov chain. In graphical models, a RJMCMC algorithm can be designed in such a way that its stationary distribution is the

---

**Algorithm 3** . BDMCMC algorithm for GCGMs

---

**Input:** A graph $G = (V, E)$ and a precision matrix $K$.

**Output:** Samples from the joint posterior distribution of $(G, K)$, (12), and waiting times.

1: **for** N iteration **do**

2:  **1. Sample the latent data.** For each $r \in V$ and $j \in \{1, ..., n\}$, we update the latent values from its full conditional distribution as follows

$$Z_r^{(j)} | Z_{V \setminus \{r\}} = z_{V \setminus \{r\}}^{(j)}, K \sim N(-\sum_{r'} K_{rr'} z_{r'}^{(j)} / K_{rr}, 1/K_{rr}),$$

   truncated to the interval $\left[ L_r^j(\mathbf{Z}), U_r^j(\mathbf{Z}) \right]$ in (10).

3:  **2. Sample from the graph.** Same as step 1 in Algorithm 2.

4:  **3. Sample from the precision matrix.** By using Algorithm 1.

5: **end for**

---

joint posterior distribution of the graph and the parameters of the graph, e.g., 3 for GGMs and 12 for GCGMs.

A RJMCMC can be implemented in various different ways. Giudici and Green (1999) implemented this algorithm only for the decomposable GGMs, because of the expensive computation of the normalizing constant $I_G(b, D)$. The RJMCMC approach developed by Dobra *et al.* (2011) and Dobra and Lenkoski (2011) is based on the Cholesky decomposition of the precision matrix. It uses an approximation for dealing with the extensive computation of the normalizing constant. To avoid the intractable normalizing constant calculation, Lenkoski (2013) and Wang and Li (2012) implemented a special case of RJMCMC algorithm, which is based on the exchange algorithm (Murray, Ghahramani, and MacKay 2012). Our implementation of RJMCMC algorithm in the **BDgraph** package defines the acceptance probability proportional to the birth/death rates in our BDMCMC algorithm. Moreover, we implement the exact sampling of G-Wishart distribution, as described in Section 3.1.1.

## 4.   The BDgraph environment

The **BDgraph** package provides a set of comprehensive tools related to Bayesian graphical models; we describe below the essential functions available in the package.

### 4.1.   Posterior sampling

We design the function `bdgraph`, as the main function of the package, to take samples from the posterior distributions based on both of our Bayesian frameworks (GGMs and GCGMs). By default, the `bdgraph` function is based on underlying sampling algorithms (Algorithms 2 and 3). Moreover, as an alternative to those BDMCMC sampling algorithms, we implement RJMCMC sampling algorithms for both the Gaussian and non-Gaussian frameworks. By using the following function

```
R> bdgraph( data, n = NULL, method = "ggm", algorithm = "bdmcmc", iter = 5000,
+          burnin = iter / 2, g.start = "empty", g.space = NULL, prior.df = 3,
+          multi.update = NULL, save.all = FALSE, print = 1000 )
```

we obtain a sample from our target joint posterior distribution. `bdgraph` returns an object of S3 class type "`bdgraph`". The functions `plot`, `print` and `summary` are working with the object "`bdgraph`". The input `data` can be an $(n \times p)$ `matrix` or a `data.frame` or a covariance $(p \times p)$ matrix ($n$ is the sample size and $p$ is the dimension); it can also be an object of class "sim", which is the output of function `bdgraph.sim`.

The argument `method` determines the type of methods, GGMs or GCGMs. Option "`ggm`" is based on Gaussian graphical models (Algorithm 2). It is designed for multivariate Gaussian data. Option "`gcgm`" is based on the GCGMs (Algorithm 3). It is designed for non-Gaussian data such as, non-Gaussian continuous, discrete or mixed data.

The argument `algorithm` refers the type of sampling algorithms which could be based on BDMCMC or RJMCMC. Option "`bdmcmc`" (as default) is for the BDMCMC sampling algorithms (Algorithms 2 and 3). Option "`rjmcmc`" is for the RJMCMC sampling algorithms, which are alternative algorithms. See Section 5 to evaluate the performance of these algorithms.

The argument `g.start` specifies the initial graph for our sampling algorithm. It could be `empty` (default) or `full`. Option `empty` means the initial graph is an empty graph and `full` means a full graph. It also could be an object with S3 class `"bdgraph"`, which allows users to run the sampling algorithm from the last objects of the previous run.

The argument `multi.update` determines the number of links that are simultaneously updated in the BDMCMC algorithm.

Note, the package **BDgraph** has two other sampling functions, `bdgraph.ts` and `bdgraph.mpl` which are designed in the similar framework as the function `bdgraph`. The function `bdgraph.ts` is for Bayesian model determination in time series graphical models (Tank, Foti, and Fox 2015). The function `bdgraph.mpl` is for Bayesian model determination in undirected graphical models based on marginal pseudo-likelihood, for both continuous and discrete variables; For more details see Dobra and Mohammadi (2018).

### 4.2. Posterior graph selection

We design the **BDgraph** package in such a way that posterior graph selection can be done based on both Bayesian model averaging (BMA), as default, and maximum a posterior probability (MAP). The functions `select` and `plinks` are designed for the objects of class `bdgraph` to provide BMA and MAP estimations for posterior graph selection.

The function

```
R> plinks( bdgraph.obj, round = 3 )
```

provides estimated posterior link inclusion probabilities for all possible links, which is based on BMA estimation. In cases where the sampling algorithm is based on BDMCMC, these probabilities for all possible links $e = (i, j)$ in the graph can be estimated using a Rao-Blackwellized estimate (Cappé *et al.* 2003, Section 2.5) based on

$$Pr(e \in E | data) = \frac{\sum_{t=1}^{N} 1(e \in E^{(t)}) W(K^{(t)})}{\sum_{t=1}^{N} W(K^{(t)})}, \tag{13}$$

where $N$ is the number of iteration and $W(K^{(t)})$ are the weights of the graph $G^{(t)}$ with the precision matrix $K^{(t)}$.

The function

```
R> select( bdgraph.obj, cut = NULL, vis = FALSE )
```

provides the inferred graph based on both BMA (as default) and MAP estimators. The inferred graph based on BMA estimation is a graph with links for which the estimated posterior probabilities are greater than a certain cut-point (as default `cut=0.5`). The inferred graph based on MAP estimation is a graph with the highest posterior probability.

Note, for posterior graph selection based on MAP estimation we should save all adjacency matrices by using the option `save.all = TRUE` in the function `bdgraph`. Saving all the adjacency matrices could, however, cause memory problems; to see how we cope with this problem the reader is referred to Appendix 7.

### 4.3.  Convergence check

In general, convergence in MCMC approaches can be difficult to evaluate. From a theoretical point of view, the sampling distribution will converge to the target joint posterior distribution as the number of iteration increases to infinity. Because we normally have little theoretical insight about how quickly MCMC algorithms converge to the target stationary distribution we therefore rely on post hoc testing of the sampled output. In general, the sample is divided into two parts: a "burn-in" part of the sample and the remainder, in which the chain is considered to have converged sufficiently close to the target posterior distribution. Two questions then arise: How many samples are sufficient? How long should the burn-in period be?

The `plotcoda` and `traceplot` are two visualization functions for the objects of class `bdgraph` that make it possible to check the convergence of the search algorithms in **BDgraph**. The function

```
R> plotcoda( bdgraph.obj, thin = NULL, control = TRUE, main = NULL, ... )
```

provides the trace of estimated posterior probability of all possible links to check convergence of the search algorithms. Option `control` is designed for the case where if `control=TRUE` (as default) and the dimension ($p$) is greater than 15, then 100 links are randomly selected for visualization.

The function

```
R> traceplot( bdgraph.obj, acf = FALSE, pacf = FALSE, main = NULL, ... )
```

provides the trace of graph size to check convergence of the search algorithms. Option `acf` is for visualization of the autocorrelation functions for graph size; option `pacf` visualizes the partial autocorrelations.

### 4.4.  Comparison and goodness-of-fit

The functions `compare` and `plotroc` are designed to evaluate and compare the performance of the selected graph. These functions are particularly useful for simulation studies. With the function

```
R> compare( sim.obj, bdgraph.obj, bdgraph.obj2 = NULL, bdgraph.obj3 = NULL,
+          colnames = NULL, vis = FALSE )
```

we can evaluate the performance of the Bayesian methods available in our **BDgraph** package and compare them with alternative approaches. This function provides several measures such as the balanced $F$-score measure (Baldi, Brunak, Chauvin, Andersen, and Nielsen 2000), which is defined as follows:

$$F_1\text{-score} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}}, \tag{14}$$

where TP, FP and FN are the number of true positives, false positives and false negatives, respectively. The $F_1$-score lies between 0 and 1, where 1 stands for perfect identification and 0 for no true positives.

The function

```
R> plotroc( sim.obj, bdgraph.obj, bdgraph.obj2 = NULL, bdgraph.obj3 = NULL,
+          bdgraph.obj4 = NULL, cut = 20, smooth = FALSE, label = TRUE,
+          main = "ROC Curve" )
```

provides a ROC plot for visualization comparison based on the estimated posterior link inclusion probabilities.

### 4.5. Data simulation

The function `bdgraph.sim` is designed to simulate different types of datasets with various graph structures. The function

```
R> bdgraph.sim( p = 10, graph = "random", n = 0, type = "Gaussian",
+              prob = 0.2, size = NULL, mean = 0, class = NULL, cut = 4,
+              b = 3, D = diag( p ), K = NULL, sigma = NULL, vis = FALSE )
```

can simulate multivariate Gaussian, non-Gaussian, discrete and mixed data with different undirected graph structures, including "random", "cluster", "scale-free", "hub", "circle", "AR(1)", "AR(1)" and "fixed" graphs. Users can specify the type of multivariate data by option `type` and the graph structure by option `graph`. They can determine the sparsity level of the obtained graph by using option `prob`. With this function users can generate mixed data from "count", "ordinal", "binary", "Gaussian" and "non-Gaussian" distributions. `bdgraph.sim` returns an object of the `S3` class type "sim". Functions `plot` and `print` work with this object type.

### 4.6. An example on simulated data

We illustrate the user interface of the **BDgraph** package by use of a simple simulation; for an extensive simulation study see Section 5. We perform all the computations on an Intel(R) Core(TM) i5 CPU 2.67GHz processor. By using the function `bdgraph.sim` we simulate 70 observations ($n = 70$) from a multivariate Gaussian distribution with 8 variables ($p = 8$) and "scale-free" graph structure, as below.

```
R> data.sim <- bdgraph.sim( n = 70, p = 8, type = "Gaussian",
+                           graph = "scale-free" )
R> round( head( data.sim $ data, 4 ), 2 )
```

```
       [,1]  [,2]  [,3]  [,4]  [,5]  [,6]  [,7]  [,8]
[1,] -0.74  1.10  0.47 -0.89 -0.40  0.10 -0.30 -0.49
[2,] -0.23 -0.37 -0.38 -0.53 -0.05  0.36 -0.53 -1.02
[3,]  0.17 -0.72 -0.20 -0.01 -0.45 -0.12 -0.47 -0.28
[4,]  0.83 -2.02  0.24  1.32  0.32 -0.08 -0.07 -0.88
```

Since the generated data are Gaussian, we run the BDMCMC algorithm which is based on Gaussian graphical models. For this we choose `method = "ggm"`, as follows:

```
R> sample.bdmcmc <- bdgraph( data = data.sim, method = "ggm", iter = 5000,
+                           save.all = TRUE )
```

We choose option "`save.all = TRUE`" to save the samples in order to check convergence of the algorithm. Running this function takes around 3 seconds, as the computational intensive tasks are performed in C++ and interfaced with R.

Since the function `bdgraph` returns an object of class S3, users can see the summary result as follows

```
R> summary( sample.bdmcmc )

$selected_g
[1,] . 1 . 1 1 1 . .
[2,] . . . . . . . 1
[3,] . . . . . . . .
[4,] . . . . . . . .
[5,] . . . . . . 1 .
[6,] . . . . . . . .
[7,] . . . . . . . .
[8,] . . . . . . . .

$p_links
[1,] . 1 0 1.00 0.40 1.00 0.44 0.01
[2,] . . . 0.02 0.08 0.03 0.06 0.67
[3,] . . . .      .    0.16 .    .
[4,] . . . .    0.02 0.08 0.01 0.45
[5,] . . . .      .    0.03 1.00 .
[6,] . . . .      .    .    0.21 0.07
[7,] . . . .      .    .    .    0.00
[8,] . . . .      .    .    .    .

$K_hat
       [,1]  [,2]  [,3]  [,4]  [,5]  [,6]  [,7]  [,8]
[1,] 10.30  3.07  0.00 -2.68 -0.33  3.34 -0.56  0.00
[2,]  3.07  3.17  0.00  0.01  0.04 -0.01  0.03 -0.53
[3,]  0.00  0.00  7.60  0.00  0.00 -0.08  0.00  0.00
[4,] -2.68  0.01  0.00  2.31 -0.01  0.05  0.00  0.28
[5,] -0.33  0.04  0.00 -0.01  6.19  0.01 -7.16  0.00
```

```
[6,]  3.34 -0.01 -0.08  0.05  0.01  7.48  0.23  0.02
[7,] -0.56  0.03  0.00  0.00 -7.16  0.23 14.07  0.00
[8,]  0.00 -0.53  0.00  0.28  0.00  0.02  0.00  3.92
```

The summary results are the adjacency matrix of the selected graph (`selected_g`) based on BMA estimation, the estimated posterior probabilities of all possible links (`p_links`) and the estimated precision matrix (`K_hat`).

In addition, the function `summary` reports a visualization summary of the results as we can see in Figure 3. At the top-left is the graph with the highest posterior probability. The plot at the top-right gives the estimated posterior probabilities of all the graphs which are visited by the BDMCMC algorithm; it indicates that our algorithm visits more than 100 different graphs and the posterior probability of the selected graph is around 0.39. The plot at the bottom-left gives the estimated posterior probabilities of the size of the graphs; it indicates that our algorithm visited mainly graphs with sizes between 5 and 11 links. At the bottom-right is the trace of our algorithm based on the size of the graphs.
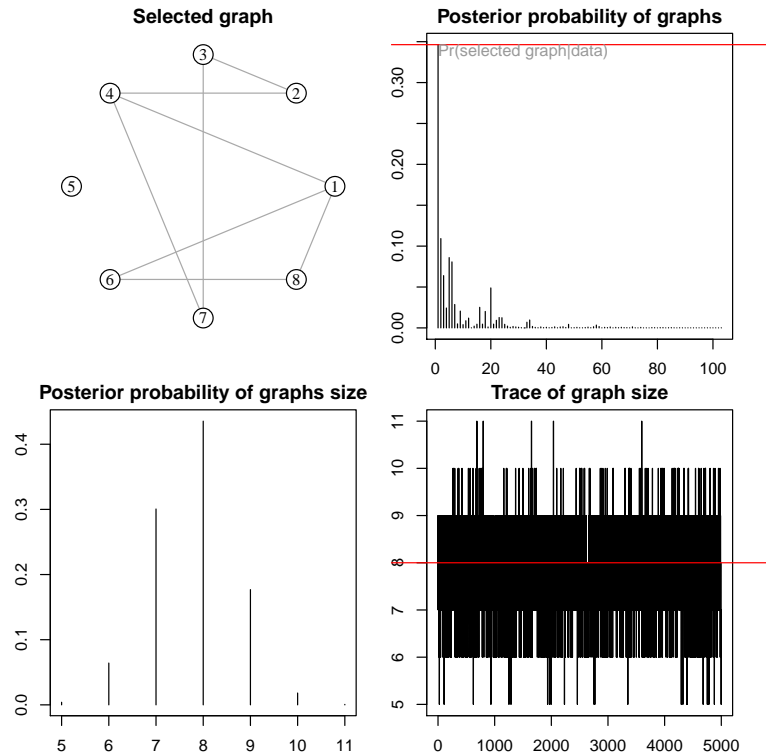


Figure 3: Visualization summary of simulation data based on output of `bdgraph` function. The figure at the top-left is the inferred graph with the highest posterior probability. The figure at the top-right gives the estimated posterior probabilities of all visited graphs. The figure at the bottom-left gives the estimated posterior probabilities of all visited graphs based on the size of the graphs. The figure at in the bottom-right is the trace of our algorithm based on the size of the graphs.

The function `compare` provides several measures to evaluate the performance of our algorithms and compare them with alternative approaches with respect to the true graph structure. To evaluate the performance of our BDMCMC algorithm (Algorithm 2) and compare it with that of an alternative algorithm, we also run the RJMCMC algorithm under the same conditions as below.

```
R> sample.rjmcmc <- bdgraph( data = data.sim, algorithm = "rjmcmc",
+                           iter = 10000 )
```

where the sampling algorithm from the joint posterior distribution is based on the RJMCMC algorithm.

Users can compare the performance of these two algorithms by using the code

```
R> plotroc( data.sim, sample.bdmcmc, sample.rjmcmc )
```

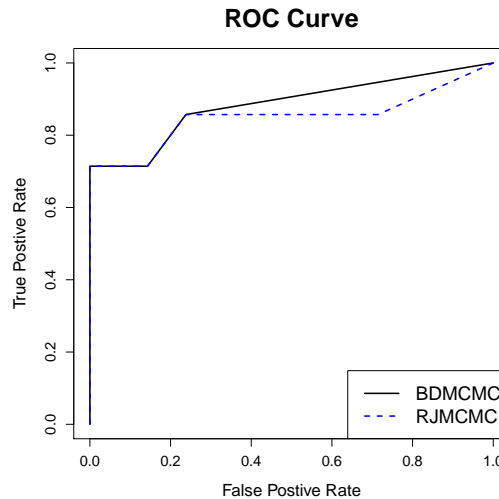which visualizes an ROC plot for both algorithms, BDMCMC and RJMCMC; see Figure 4.



Figure 4:  ROC plot to compare the performance of the BDMCMC and RJMCMC algorithms for a simulated toy example.

We can also compare the performance of those algorithms by using the `compare` function as follows:

```
R> compare( data.sim, sample.bdmcmc, sample.rjmcmc,
+          colnames = c("True graph", "BDMCMC", "RJMCMC" ) )
```

|                | True graph | BDMCMC | RJMCMC |
|----------------|-----------|--------|--------|
| true positive  | 7         | 5      | 5      |
| true negative  | 21        | 18     | 19     |
| false positive | 0         | 3      | 2      |
| false negative | 0         | 2      | 2      |
| F-score        | 1         | 0.67   | 0.71   |

```
specificity            1     0.86     0.90
sensitivity            1     0.71     0.71
MCC                    1     0.55     0.62
```

The results show that for this specific simulated example both algorithms have more or less the same performance; for a comprehensive comparison see next section.

In this simulation example, we run both BDMCMC and RJMCMC algorithms for $10,000$ iterations, $5,000$ of them as burn-in. To check whether the number of iterations is enough and to monitoring the convergence of our both algorithm, we run

```
R> plotcoda( sample.bdmcmc )
R> plotcoda( sample.rjmcmc )
```

The results in Figure 5 indicate that our BDMCMC algorithm converges faster with compare with RJMCMC algorithm.
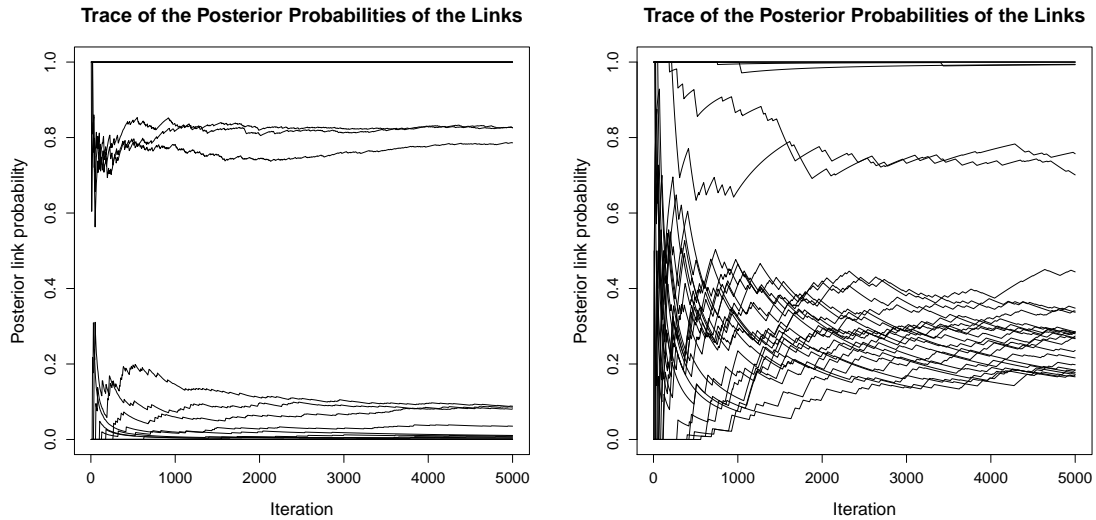


Figure 5: Plot for monitoring the convergence based on the trace of estimated posterior probability of all possible links for the BDMCMC algorithm (left) and the RJMCMC algorithm (right).

## 5. Simulation study

In this section we present a comprehensive simulation study to investigate the performance and capability of the Bayesian methods implemented in our **BDgraph** package and compare them with two frequentist approaches, which are available in the **huge** package (Zhao *et al.* 2014; Zhao, Liu, Roeder, Lafferty, and Wasserman 2012). All the simulation studies were performed in parallel on a 450 batch nodes with 12 cores and 24 GB of memory, running Linux.

We consider 6 various kinds of synthetic graphical models with $p$ nodes:

1. *Random:* A graph in which links are randomly generated from independent Bernoulli distributions with probability $2/(p-1)$ and the corresponding precision matrix is generated from $K \sim W_G(3, I_p)$.

2. *Cluster:* A graph in which the number of clusters is $max\{2, [p/20]\}$. Each cluster has the same structure as the random graph. The corresponding precision matrix is generated from $K \sim W_G(3, I_p)$.

3. *Scale-free:* A graph which has a power-law degree distribution generated by the Barabasi-Albert algorithm (Albert and Barabási 2002). The corresponding precision matrix is generated from $K \sim W_G(3, I_p)$.

4. *Hub:* A graph in which every node is connected to one node, and the corresponding precision matrix is generated from $K \sim W_G(3, I_p)$.

5. *AR(2):* A graph with $k_{ii} = 1$, $k_{i,i-1} = k_{i-1,i} = 0.5$, and $k_{i,i-2} = k_{i-2,i} = 0.25$, and $k_{ij} = 0$ otherwise.

6. *circle:* A graph with $k_{ii} = 1$, $k_{i,i-1} = k_{i-1,i} = 0.5$, and $k_{1p} = k_{p1} = 0.4$, and $k_{ij} = 0$ otherwise.

For each graphical model, we consider various scenarios based on a different number of variables $p = \{20, 100, 150, 200\}$ and sample sizes $n = \{p, 2p\}$. We generate data from $p$ dimensional multivariate normal distribution, $N_p(0, K)$, using function `bdgraph.sim` in the **BDgraph** package.

For each scenario, we evaluate the performance of our **BDgraph** package with two types of sampling algorithms, BDMCMC and RJMCMC. Users can run these algorithms by running the `bdgraph` function with option `algorithm = "bdmcmc"` or `"rjmcmc"`. We run both algorithms in the same situation with $100,000$ iterations and $50,000$ as burn-in.

### 5.1. Performance of our sampling algorithms

We are interested in evaluating the performance of our Bayesian framework based on both of our sampling algorithms, BDMCMC and RJMCMC, as well as the performance of those algorithms based on two different posterior graph estimations, Bayesian model averaging (BMA) and maximum a posterior probability (MAP). As we explain in Section 4.2, for BMA the inferred graph is a graph with links for which the estimated posterior probabilities are greater than cut-point 0.5, the default of the package. For MAP the inferred graph is one with the highest posterior probability.

Table 1 reports the average $F_1$-score (14) with standard errors in parentheses, where we have repeated the experiments 50 times. The BMA performs better, as its $F_1$-score is greater than MAP in all cases for both algorithms. To compare the two algorithms: although the BDMCMC algorithm is computationally expensive, it converges faster, especially for high cases of $p$. Similar patterns can also be found on the averaged ROC curves in Figures 7 and 8.

Figure 6 displays time taken for 1000 sweeps as a function of the dimension of the graphical model for both BDMCMC and RJMCMC algorithms. It shows the BDMCMC algorithm is computationally faster than the RJMCMC algorithm. One reason is that step 1 of the

| p | n | graph | BDMCMC | | RJMCMC | |
|---|---|---|---|---|---|---|
| | | | MAP | BAE | MAP | BAE |
| 20 | 20 | random | 0.37 (0.05) | **0.47** (0.07) | 0.36 (0.05) | 0.46 ( 0.06 ) |
| | | scale-free | 0.35 (0.11) | **0.36** (0.11) | 0.24 (0.05) | **0.36** (0.09) |
| | | cluster | 0.45 (0.08) | **0.48** (0.09) | 0.36 (0.05) | 0.47 (0.09) |
| | | hub | 0.25 (0.08) | 0.26 (0.08) | 0.21 (0.04) | **0.32** (0.07) |
| | | AR2 | 0.4 (0.08) | 0.41 (0.08) | 0.4 (0.06) | **0.46** (0.09) |
| | | circle | 0.9 (0.07) | **0.91** (0.07) | 0.56 (0.08) | 0.8 (0.11) |
| | 40 | random | 0.54 (0.08) | **0.56** (0.08) | 0.42 (0.05) | 0.55 (0.08) |
| | | scale-free | **0.51** (0.15) | **0.51** (0.15) | 0.31 (0.08) | 0.5 (0.12) |
| | | cluster | 0.57 (0.08) | **0.58** (0.08) | 0.43 (0.06) | **0.58** (0.09) |
| | | hub | 0.36 (0.11) | 0.38 (0.1) | 0.27 (0.06) | **0.41** (0.1) |
| | | AR2 | 0.58 (0.07) | 0.58 (0.07) | 0.53 (0.08) | **0.65** (0.06) |
| | | circle | 0.97 (0.03) | **0.98** (0.02) | 0.8 (0.07) | 0.96 (0.04) |
| 100 | 100 | random | 0.36 (0.02) | 0.36 (0.02) | 0.36 (0.01) | **0.37** (0.01) |
| | | scale-free | 0.23 (0.12) | **0.24** (0.13) | 0.12 (0.02) | 0.21 (0.04) |
| | | cluster | 0.39 (0.03) | 0.4 (0.03) | 0.43 (0.02) | **0.46** (0.02) |
| | | hub | 0.19 (0.1) | **0.2** (0.11) | 0.11 (0.02) | 0.18 (0.04) |
| | | AR2 | 0.79 (0.02) | **0.8** (0.02) | 0.47 (0.02) | 0.72 (0.03) |
| | | circle | **1** (0.00) | **1** (0.00) | 0.43 (0.05) | 0.44 (0.04) |
| | 200 | random | 0.47 (0.04) | **0.48** (0.05) | 0.4 (0.02) | 0.42 (0.02) |
| | | scale-free | 0.42 (0.11) | **0.44** (0.11) | 0.17 (0.04) | 0.32 (0.06) |
| | | cluster | 0.5 (0.05) | 0.5 (0.05) | 0.51 (0.02) | **0.59** (0.02) |
| | | hub | 0.39 (0.1) | **0.4** (0.1) | 0.15 (0.03) | 0.28 (0.05) |
| | | AR2 | **0.91** (0.02) | **0.91** (0.02) | 0.6 (0.02) | 0.86 (0.02) |
| | | circle | **1** (0.00) | **1** (0.00) | 0.52 (0.03) | 0.52 (0.05) |

Table 1: The $F_1$-score measure of our Bayesian method based on two sampling algorithms (BDMCMC and RJMCMC), as well as the performance of those algorithms based on two different posterior graph estimations, Bayesian model averaging (BMA) and the maximum a posterior probability (MAP). The result is for 6 different graphs with various number of variables $p = \{20, 100\}$ and different number of observations ($n = \{p, 2p\}$), with 50 replications, and standard deviations are in parentheses. The $F_1$-score reaches its best score at 1 and its worst at 0. The best methods are boldfaced.

Algorithm 2, which is the main computational part, is efficiently implemented in parallel in C++ using **OpenMP**. Another reason is that in the BDMCMC algorithm we use the advantage of multiple updating, which adds or removes multiple links per iteration (Dobra and Mohammadi 2018, Section 5.4).

Table 2 reports the memory usage of our BDMCMC algorithm for both BMA and MAP estimations. As the results show, MAP estimation requires more memory as compared to BMA, since for the MAP estimation we need to save all the graphs visited by our sampling algorithm. However, we have implemented an efficient way to document the adjacency matrices of all the visited graphs by our sampling algorithm; see Appendix 7.

To summarize, the results indicate for our Bayesian framework that (1) the BMA estimation globally performs better than MAP and (2) the BDMCMC algorithm is computationally less expensive and converges faster than the RJMCMC algorithm.
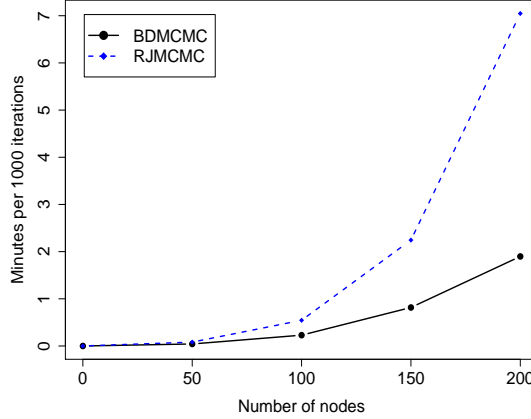
Figure 6:    Time in minutes for 1000 iterations for versus number of nodes ($p$) for both BDMCMC and RJMCMC algorithms. This plot is based on the result from a laptop computer with an Intel CoreTM i5 − 5200U CPU and 8GB RAM.

| p | 50 | 100 | 150 | 200 |
|---|---|---|---|---|
| BMA | 0.8 | 0.9 | 1.1 | 1.4 |
| MAP | 63.7 | 241.1 | 537.9 | 954.1 |

Table 2:   Memory usage in megabytes for versus number of nodes (p) with $100,000$ iterations for our sampling algorithms with both BMA and MAP estimations.


## 5.2.   Comparison with frequentist methods

Here we present the summary of our comparison of the Bayesian approaches of our **BDgraph** package with two frequentist approaches, the graphical lasso (glasso) (Friedman *et al.* 2008) and the Meinshausen-Buhlmann graph estimation (mb) (Meinshausen and Buhlmann 2006), both of which are available in the **huge** package.

Table 3 presents the averaged $F_1$-score, with standard errors in parentheses, corresponding to our Bayesian approach based on the BDMCMC algorithm and the two frequentist methods, glasso and mb, with different regularization parameter selection approaches available in the **huge** package. For our Bayesian approach, the inferred graph is selected based on a BMA estimator with cut-point 0.5 as a default of function `select` in our **BDgraph** package. For the glasso and mb methods, the inferred graph is selected by determining the regularization parameter $\lambda$ which controls the sparsity of the inferred graph. The inferred graphs in the glasso and mb methods are quite sensitive to the values of $\lambda$, since different $\lambda$'s lead to different graph structures (Vujačić, Abbruzzo, and Wit 2015). For the glasso methods, we consider three different regularization parameter selection approaches for choosing $\lambda$: rotation information criterion (ric) (Zhao *et al.* 2012), the stability approach to regularization selection (StARS) (Liu, Roeder, and Wasserman 2010), and the extended Bayesian information criterion (ebic) (Foygel and Drton 2010). These approaches are available in the **huge** package. For the mb method, we consider two regularization parameter selection approaches, namely ric and StARS.

Based on Table 3, we see that our Bayesian method performs better than the glasso and mb approaches except for hub graphs. The results in the table show the performance of

| p | n | | BDMCMC | glasso | | | mb | |
|---|---|---|---|---|---|---|---|---|
| | | | | ric | StARS | ebic | ric | StARS |
| 20 | 20 | random | **0.47** (0.07) | 0.38 (0.12) | 0.21 (0.14) | 0.00 (0.00) | 0.39 (0.12) | 0.29 (0.12) |
| | | scale-free | **0.36** (0.11) | 0.29 (0.15) | 0.28 (0.19) | 0.00 (0.00) | 0.31 (0.17) | 0.34 (0.18) |
| | | cluster | **0.48** (0.09) | 0.41 (0.14) | 0.29 (0.18) | 0.00 (0.00) | 0.41 (0.14) | 0.33 (0.15) |
| | | hub | 0.26 (0.08) | **0.27** (0.11) | 0.15 (0.17) | 0.00 (0.00) | 0.26 (0.13) | 0.23 (0.18) |
| | | AR2 | **0.41** (0.08) | 0.00 (0.01) | 0.1 (0.11) | 0.00 (0.00) | 0.00 (0.01) | 0.12 (0.1) |
| | | circle | **0.91** (0.07) | 0.00 (0.00) | 0.03 (0.14) | 0.00 (0.00) | 0.00 (0.00) | 0.83 (0.18) |
| | 40 | random | **0.56** (0.08) | 0.46 (0.1) | 0.36 (0.13) | 0.01 (0.06) | 0.5 (0.12) | 0.47 (0.1) |
| | | scale-free | 0.51 (0.15) | 0.33 (0.15) | 0.45 (0.16) | 0.00 (0.00) | 0.37 (0.19) | **0.55** (0.17) |
| | | cluster | **0.58** (0.08) | 0.47 (0.15) | 0.49 (0.11) | 0.04 (0.14) | 0.49 (0.13) | 0.52 (0.12) |
| | | hub | 0.38 (0.1) | 0.26 (0.11) | 0.36 (0.15) | 0.00 (0.00) | 0.29 (0.14) | **0.44** (0.15) |
| | | AR2 | **0.58** (0.07) | 0.01 (0.03) | 0.27 (0.12) | 0.00 (0.00) | 0.01 (0.03) | 0.3 (0.12) |
| | | circle | **0.98** (0.02) | 0.00 (0.00) | 0.04 (0.15) | 0.39 (0.03) | 0.00 (0.00) | 0.86 (0.05) |
| 150 | 150 | random | **0.45** (0.03) | 0.26 (0.11) | 0.23 (0.12) | 0.11 (0.15) | 0.23 (0.11) | 0.1 (0.16) |
| | | scale-free | 0.33 (0.1) | 0.35 (0.22) | 0.34 (0.12) | 0.00 (0.00) | **0.46** (0.3) | 0.42 (0.07) |
| | | cluster | **0.51** (0.1) | 0.34 (0.24) | 0.46 (0.04) | 0.38 (0.22) | 0.4 (0.3) | 0.48 (0.19) |
| | | hub | 0.32 (0.09) | 0.36 (0.13) | 0.32 (0.05) | 0.02 (0.07) | **0.52** (0.22) | 0.39 (0.07) |
| | | AR2 | **0.85** (0.01) | 0.00 (0.01) | 0.53 (0.03) | 0.00 (0.00) | 0.00 (0.00) | 0.64 (0.04) |
| | | circle | **1** (0.00) | 0.00 (0.00) | 0.00 (0.00) | 0.19 (0.01) | 0.00 (0.00) | 0.00 (0.00) |
| | 300 | random | **0.58** (0.02) | 0.34 (0.06) | 0.26 (0.01) | 0.35 (0.04) | 0.31 (0.09) | 0.23 (0.12) |
| | | scale-free | 0.46 (0.05) | 0.41 (0.12) | 0.38 (0.13) | 0.37 (0.14) | **0.72** (0.09) | 0.64 (0.07) |
| | | cluster | 0.56 (0.06) | 0.56 (0.08) | 0.58 (0.06) | 0.56 (0.05) | 0.68 (0.09) | **0.72** (0.03) |
| | | hub | 0.4 (0.09) | 0.36 (0.16) | 0.42 (0.07) | 0.41 (0.1) | 0.59 (0.28) | **0.67** (0.05) |
| | | AR2 | **0.91** (0.02) | 0.38 (0.05) | 0.66 (0.01) | 0.00 (0.00) | 0.42 (0.07) | 0.83 (0.01) |
| | | circle | **1** (0.00) | 0.00 (0.00) | 0.07 (0.05) | 0.17 (0.00) | 0.00 (0.00) | 0.00 (0.00) |
| 200 | 200 | random | **0.74** (0.04) | 0.27 (0.05) | 0.18 (0.12) | 0.17 (0.14) | 0.22 (0.09) | 0.13 (0.14) |
| | | scale-free | 0.35 (0.08) | 0.32 (0.15) | 0.33 (0.08) | 0.14 (0.22) | **0.56** (0.25) | 0.51 (0.09) |
| | | cluster | **0.57** (0.14) | 0.26 (0.24) | 0.43 (0.09) | 0.37 (0.16) | 0.33 (0.3) | 0.4 (0.28) |
| | | hub | 0.4 (0.16) | 0.3 (0.16) | 0.32 (0.06) | 0.05 (0.11) | 0.44 (0.25) | **0.51** (0.09) |
| | | AR2 | **0.84** (0.01) | 0.03 (0.02) | 0.59 (0.01) | 0.00 (0.00) | 0.04 (0.02) | 0.71 (0.04) |
| | | circle | **1** (0.00) | 0.00 (0.00) | 0.00 (0.00) | 0.16 (0.00) | 0.00 (0.00) | 0.00 (0.00) |
| | 400 | random | **0.74** (0.04) | 0.29 (0.05) | 0.21 (0.08) | 0.3 (0.04) | 0.25 (0.08) | 0.15 (0.13) |
| | | scale-free | 0.52 (0.09) | 0.36 (0.11) | 0.38 (0.14) | 0.35 (0.15) | 0.57 (0.28) | **0.72** (0.08) |
| | | cluster | 0.66 (0.11) | 0.44 (0.16) | 0.59 (0.07) | 0.56 (0.1) | 0.52 (0.21) | **0.72** (0.03) |
| | | hub | 0.55 (0.12) | 0.34 (0.15) | 0.38 (0.09) | 0.37 (0.1) | 0.59 (0.28) | **0.68** (0.08) |
| | | AR2 | **0.9** (0.01) | 0.57 (0.04) | 0.68 (0.01) | 0.69 (0.01) | 0.6 (0.02) | 0.88 (0.01) |
| | | circle | **1** (0.00) | 0.00 (0.00) | 0.03 (0.04) | 0.15 (0.00) | 0.00 (0.00) | 0.00 (0.00) |

Table 3: The $F_1$-score measure of our Bayesian method (BDMCMC) and glasso and mb methods from the **huge** package for 6 different graphs with various number of variables $p = \{20, 150, 200\}$ and different number of observations ($n = \{p, 2p\}$), with 50 replications, and standard deviations are in parentheses. The $F_1$-score reaches its best score at 1 and its worst at 0. The best methods are boldfaced.

both the glasso and mb methods is quite sensitive to their regularization parameter selection approaches. Both the glasso and mb methods have good performance in some cases and substantially deteriorate in other cases. For example, the mb approach with StARS criterion for scale-free graphs has good performance, but not for circle and random graphs.

To compare the performance of our Bayesian method for both BDMCMC and RJMCMC algorithms with both glasso and mb, the average ROC curves for various $p$ and $n$ are presented

in Figures 7 and 8. For each figure four curves are presented, corresponding to our BDMCMC and RJMCMC algorithms and glasso and mb methods from the **huge** package. We can see that the performance of BDMCMC and RJMCMC is comparable and that both algorithms in most cases outperform the glasso and mb methods. As can be seen from the ROC curves the BDMCMC performs better than the RJMCMC algorithm. From a theoretical point of view BDMCMC and RJMCMC should asymptotically have the same performance, but differences are due to the finite samples.

# 6. Application to real datasets

In this section we analyze two datasets from genetics and sociology, using the functions available in the **BDgraph** package. In Section 6.1 we analyze a labor force survey dataset, involving mixed data. In Section 6.2 we analyze human gene expression data, which do not follow the Gaussianity assumption. Both datasets are available in the **BDgraph** package.

## 6.1. Application to labor force survey data

Hoff (2007) analyzes the multivariate associations among income, education and family background, using data concerning 1002 males in the U.S labor force. The dataset is available in the **BDgraph** package.

```
R> data( "surveyData" )
R> head( surveyData, 5 )

     income degree children pincome pdegree pchildren age
[1,]     NA      1        3       3       1         5  59
[2,]     11      0        3      NA       0         7  59
[3,]      8      1        1      NA       0         9  25
[4,]     25      3        2      NA       0         5  55
[5,]    100      3        2       4       3         2  56
```

Missing data are indicated by `NA`; in general, the rate of missing data is about 9%, with higher rates for the variables `income` and `pincome`. In this dataset we have seven observed variables as follows:

`income`: An ordinal variable indicating respondent's income in 1000s of dollars.
`degree`: An ordinal variable with five categories indicating respondent's highest educational degree.
`children`: A count variable indicating number of children of respondent.
`pincome`: An ordinal variable with five categories indicating financial status of respondent's parents.
`pdegree`: An ordinal variable with five categories indicating highest educational degree of respondent's parents.
`pchildren`: A count variable indicating number of children of respondent's parents.
`age`: A count variable indicating respondent's age in years.

Since the variables are measured on various scales, the marginal distributions are heterogeneous, which makes the study of their joint distribution very challenging. However, we can
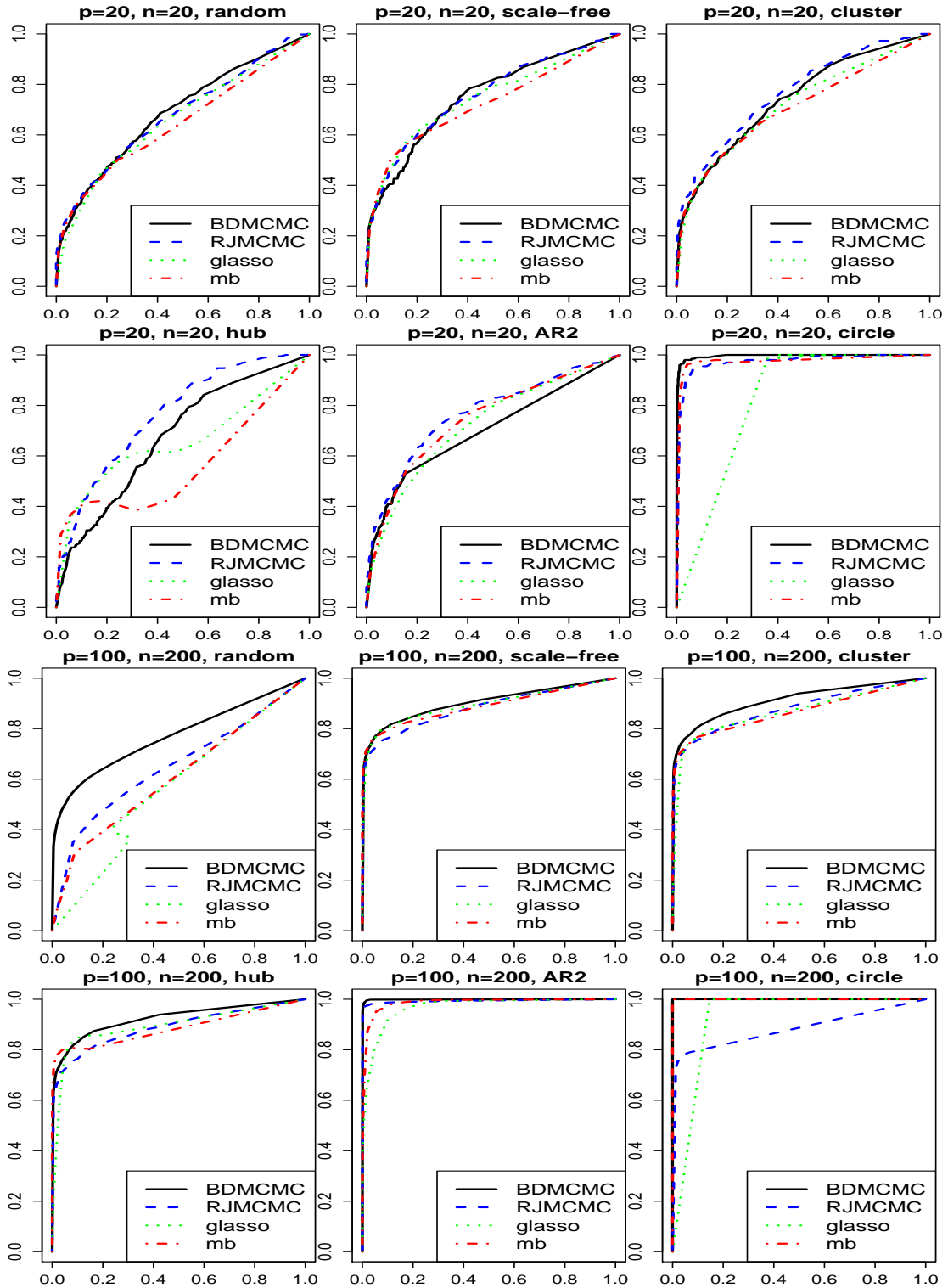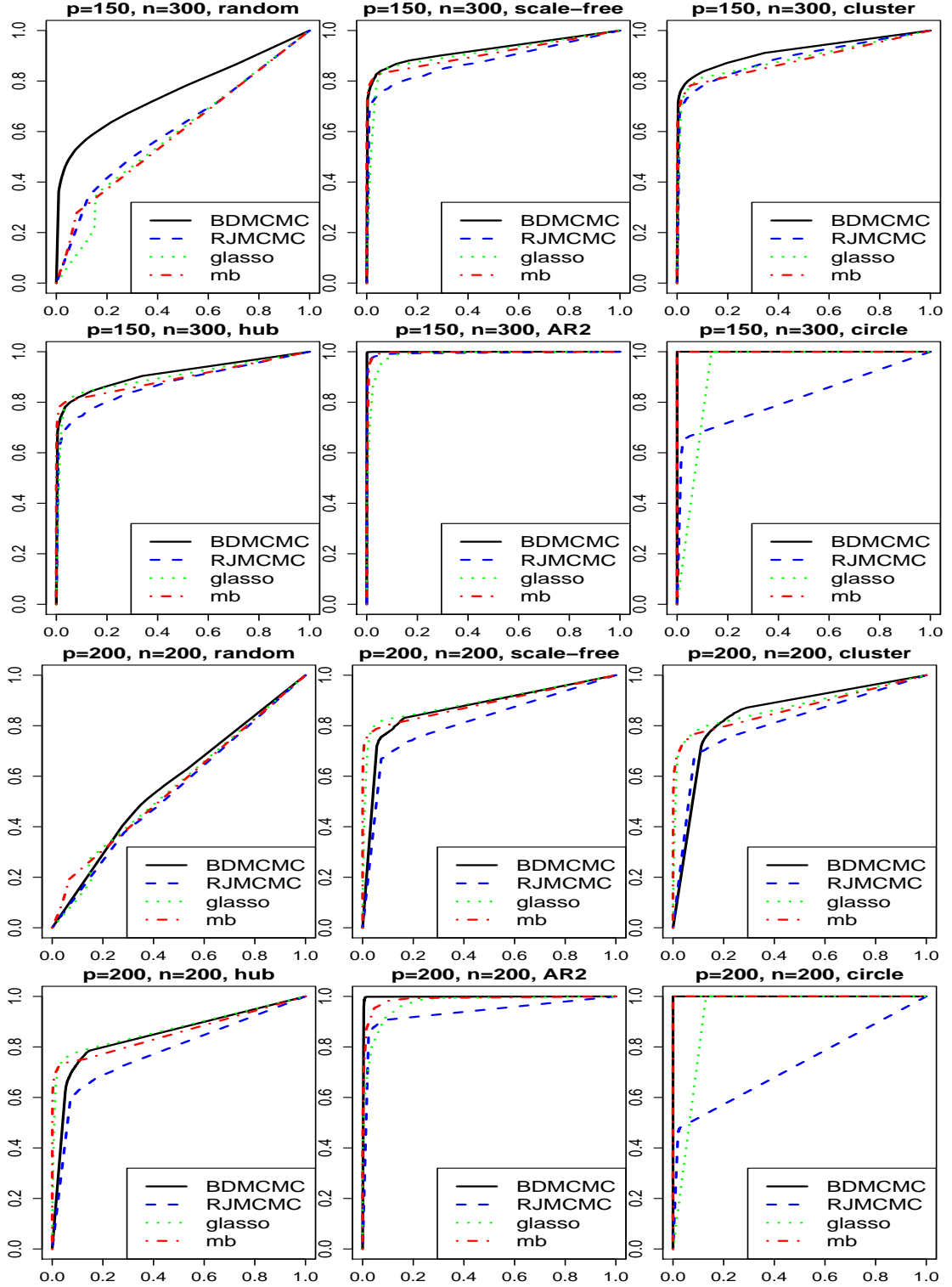
Figure 7: ROC curves present the performances of the two sampling algorithms from the **BDgraph** package and two frequentist methods from the **huge** package for $p = \{20, 100\}$ and varying graph structures and with with 50 replications.

Figure 8:    ROC curves present the performances of the two sampling algorithms from the **BDgraph** package and two frequentist methods from the **huge** package for $p = \{150, 200\}$ and varying graph structures and with with 50 replications.

apply to this dataset our Bayesian framework based on the Gaussian copula graphical models. Thus, we run the function `bdgraph` with option `method = "gcgm"`. For the prior distributions of the graph and precision matrix, as default of the function `bdgraph`, we place a uniform distribution as a uninformative prior on the graph and a G-Wishart $W_G(3, I_7)$ on the precision matrix. We run our function for $200,000$ iterations with $150,000$ as burn-in.

```
R> sample.bdmcmc <- bdgraph( data = surveyData, method = "gcgm",
+                           iter = 200000, burnin = 150000 )
R> summary( sample.bdmcmc )

$selected_g
          income degree children pincome pdegree pchildren age
income         .      1        1       .       .         .   1
degree         .      .        1       .       1         1   .
children       .      .        .       .       1         1   1
pincome        .      .        .       .       1         1   .
pdegree        .      .        .       .       .         1   1
pchildren      .      .        .       .       .         .   .
age            .      .        .       .       .         .   .


$p_links
          income degree children pincome pdegree pchildren  age
income         .      1     1.00    0.41    0.22      0.31 1.00
degree         .      .     0.64    0.26    1.00      0.67 0.02
children       .      .        .    0.41    0.89      1.00 1.00
pincome        .      .        .       .    1.00      0.53 0.02
pdegree        .      .        .       .       .      0.99 1.00
pchildren      .      .        .       .       .         . 0.03
age            .      .        .       .       .         .    .


$K_hat
          income degree children pincome pdegree pchildren   age
income      1.76  -1.05    -0.67   -0.07   -0.05      0.05 -0.51
degree     -1.05   3.42     0.29    0.02   -2.32      0.30 -0.01
children   -0.67   0.29     9.54    0.13    1.02     -1.06 -2.69
pincome    -0.07   0.02     0.13    4.41   -2.40      0.30  0.00
pdegree    -0.05  -2.32     1.02   -2.40   12.40      1.42  0.83
pchildren   0.05   0.30    -1.06    0.30    1.42      7.15  0.01
age        -0.51  -0.01    -2.69    0.00    0.83      0.01  3.16
```

The results of the function `summary` are the adjacency matrix of the selected graph (`selected_g`), estimated posterior probabilities of all possible links (`p_links`) and estimated precision matrix (`K_hat`).

Figure 9 presents the selected graph, a graph with links for which the estimated posterior probabilities are greater than 0.5. Links in the graph are indicated by signs "+" and "-", which represent positively and negatively correlated relationships between associated variables, respectively.
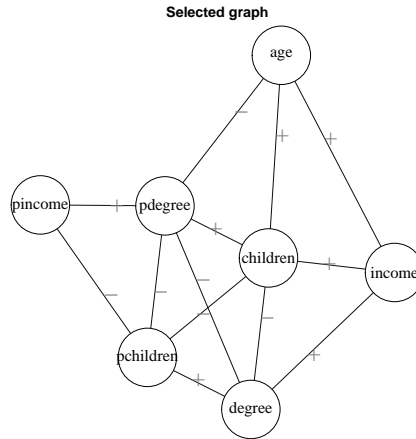
Figure 9:    Inferred graph for the labor force survey data based on output from `bdgraph`. Sign "+" represents a positively correlated relationship between associated variables and "-" represents a negatively correlated relationship.

The results indicate that education, fertility and age have strong associations with income, since there are highly positively correlated relationships between income and those three variables, with posterior probability equal to one for all. It also shows that a respondent's education and fertility are negatively correlated, with posterior probability more than 0.6. The respondent's education is certainly related to his parent's education, since there is a positively correlated relationship, with posterior probability equal to one. Moreover, the results indicate that relationships between income, education and fertility hold across generations.

For this dataset, Hoff (2007) estimated the conditional independence between variables by inspecting whether the 95% credible intervals for the associated regression parameters, which do not contain zero. Our results are the same as that of Hoff except for one link. Our results indicate that there is a strong relationship between parents' education (`pdegree`) and fertility (`child`), a link which is not selected by Hoff.

## 6.2. Application to human gene expression

Here, by using the functions that are available in the **BDgraph** package, we study the structure learning of the sparse graphs applied to the human gene expression data which were originally described by Stranger, Nica, Forrest, Dimas, Bird, Beazley, Ingle, Dunning, Flicek, Koller *et al.* (2007). They collected data to measure gene expression in B-lymphocyte cells from Utah inhabitants with Northern and Western European ancestry. They considered 60 individuals whose genotypes were available online at `ftp://ftp.sanger.ac.uk/pub/genevar`. Here the focus was on the 3,125 Single Nucleotide Polymorphisms (SNPs) that were found in the 5' UTR (untranslated region) of mRNA (messenger RNA) with a minor allele frequency $\geq 0.1$. Since the UTR (untranslated region) of mRNA (messenger RNA) has previously been subject to investigation, it should play an important role in the regulation of gene expression. The raw data were background-corrected and then quantile-normalized across replicates of a single individual and then median normalized across all individuals. Following Bhadra and Mallick (2013), of the 47,293 total available probes, we consider the 100 most variable probes that correspond to different Illumina TargetID transcripts. The data for these 100 probes are

available in our package. To see the data users can run the code

```
R> data( "geneExpression" )
R> dim( geneExpression )
```

```
60 100
```

The data consist of only 60 observations ($n = 60$) across 100 genes ($p = 100$). This dataset is an interesting case study for graph structure learning, as it has been used by Bhadra and Mallick (2013); Mohammadi and Wit (2015); Gu, Cao, Ning, and Liu (2015).

In this dataset, all the variables are continuous but not Gaussian, as can be seen in Figure 10. Thus, we apply Gaussian copula graphical models, using the function `bdgraph` with option `method = "gcgm"`. For the prior distributions of the graph and precision matrix, as default of the function `bdgraph`, we place a uniform distribution as a uninformative prior on the graph and the G-Wishart $W_G(3, I_{100})$ on the precision matrix.
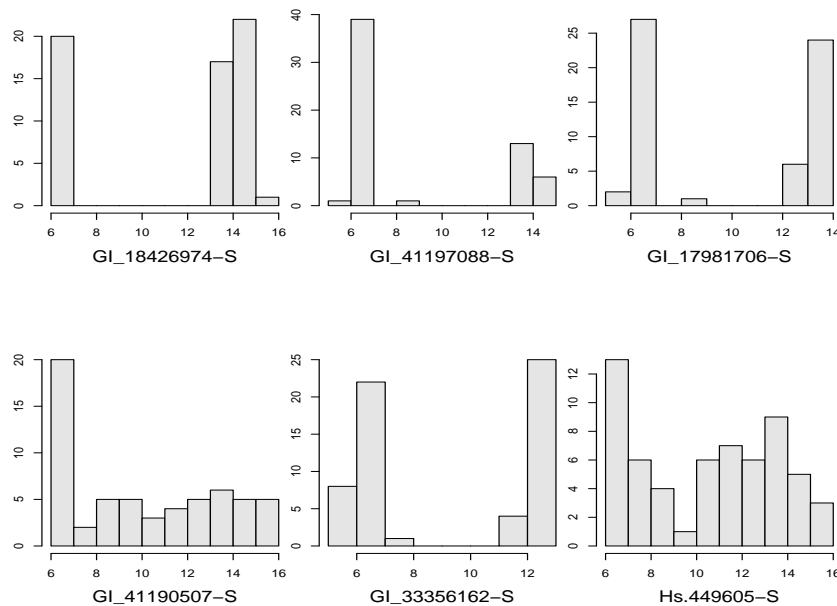


Figure 10: Univariate histograms of first 6 genes in human gene dataset.

We run our function for $100,000$ iterations with $50,000$ as burn-in as follows:

```
R> sample.bdmcmc <- bdgraph( data = geneExpression, method = "gcgm",
+                            iter = 100000 )
```

This took about 55 minutes. We use the following code to visualize the graph with estimated posterior probabilities greater than 0.9.

```
R> select( sample.bdmcmc, cut = 0.9, vis = TRUE )
```

By using option `vis = TRUE`, the function plots the selected graph. Figure 11 visualizes the selected graph which consists of 207 links with estimated posterior probabilities (13) greater than 0.9.
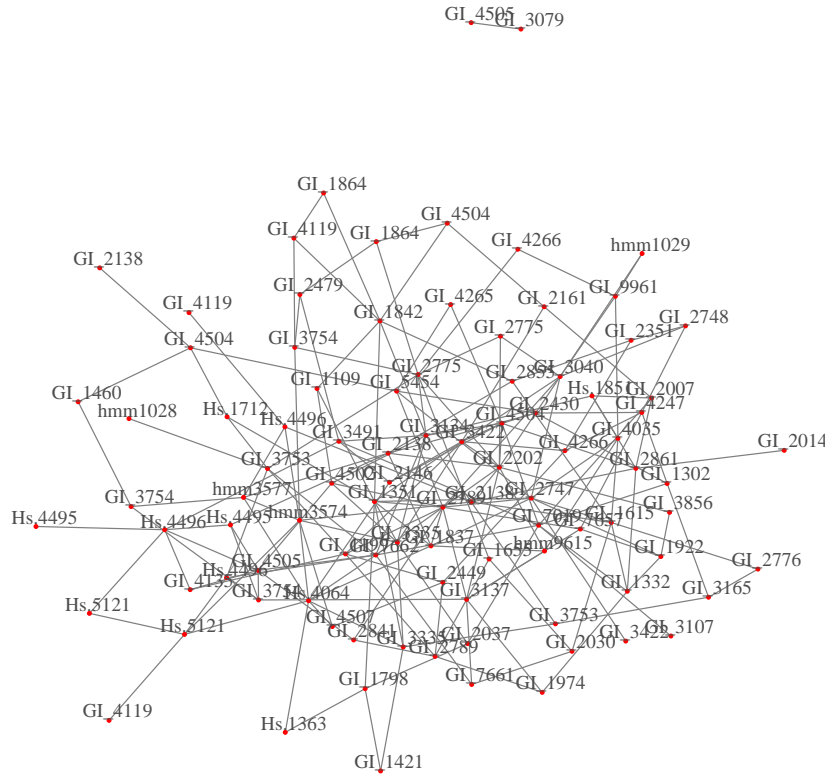


Figure 11:   The inferred graph for human gene expression data using Gaussian copula graphical models. This graph consists of 207 links with estimated posterior probabilities greater than 0.9.

The function `plinks` reports the estimated posterior probabilities of all possible links in the graph. For our data the output of this function is a $100 \times 100$ matrix. Figure 12 reports the visualization of that matrix.

Most of the links in our selected graph conform to results in previous studies. For instance, Bhadra and Mallick (2013) found 54 significant interactions between genes, most of which are covered by our method. In addition, our approach indicates additional gene interactions with high posterior probabilities that are not found in previous studies; this result may complement the analysis of human gene interaction networks.

## 7. Conclusion

We have presented the **BDgraph** package which was designed for Bayesian structure learning in general – decomposable and non-decomposable – undirected graphical models. The package implements recent improvements in computation, sampling and inference of Gaussian graphical models (Mohammadi and Wit 2015; Dobra *et al.* 2011) for Gaussian data and

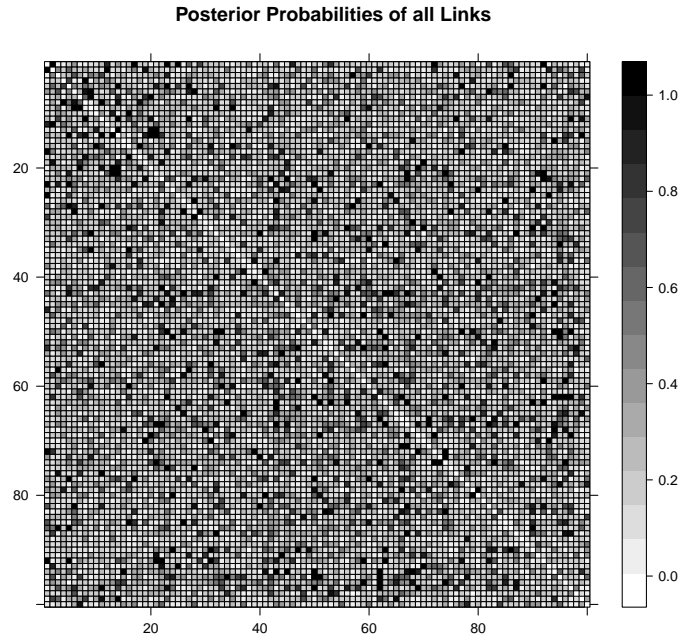**Posterior Probabilities of all Links**



Figure 12: Image visualization of the estimated posterior probabilities of all possible links in the graph on human gene expression data.

Gaussian copula graphical models (Mohammadi *et al.* 2017a; Dobra and Lenkoski 2011) for non-Gaussian, discrete and mixed data.

We are committed to maintaining and developing the **BDgraph** package in the future. Future versions of the package will contain more options for prior distributions of the graph and precision matrix. One possible extension of our package, would be to deal with outliers, by using robust Bayesian graphical modelling using Dirichlet **t**-Distributions (Finegold and Drton 2014; Mohammadi and Wit 2014). An implementation of this method would be desirable in actual applications.

# Acknowledgments

# Appendix: Dealing with memory usage restriction

The memory usage restriction is one of the challenges of Bayesian inference for maximum a posterior probability (MAP) estimation and monitoring the convergence check, especially for high-dimensional problems. For example, to compute MAP estimation in our **BDgraph** pack-

age, we must document the adjacency matrices of all the visited graphs by our MCMC sampling algorithms, which causes memory usage restriction in R. Indeed, the function `bdgraph` in our package in case `save.all = TRUE` is documented to return all of the adjacency matrices for all iterations after burn-in. For instance, in our example 6.2 we have

```
R> iter   <- 100000
R> burnin <- 50000
R> p      <- 100
R> graph  <- matrix( 1, p, p )
R> print( ( iter - burnin ) * object.size( graph ), units = "auto" )
```

```
3.75 Gb
```

A naive way is to save all the matrices, which leads to memory usage restriction, as it costs 3.75 gigabytes of memory. To cope with this problem, instead of saving all adjacency matrices we simply transfer the upper triangular part of the adjacency matrix to one single character; see codes below:

```
R> string_graph <- paste( graph[ upper.tri( graph ) ], collapse = '' )
R> print( ( iter - burnin ) * object.size( string_graph ), units = "auto" )
```

```
240.4 Mb
```

In this efficient way we need only 240.4 megabytes instead of 3.75 gigabytes of memory.

# References

Abegaz F, Wit E (2015). "Copula Gaussian graphical models with penalized ascent Monte Carlo EM algorithm." *Statistica Neerlandica*, **69**(4), 419–441.

Albert R, Barabási AL (2002). "Statistical mechanics of complex networks." *Reviews of modern physics*, **74**(1), 47.

Anderson E, Bai Z, Bischof C, Blackford S, Demmel J, Dongarra J, Du Croz J, Greenbaum A, Hammarling S, McKenney A, Sorensen D (1999). *LAPACK Users' Guide*. Third edition. Society for Industrial and Applied Mathematics, Philadelphia, PA. ISBN 0-89871-447-8 (paperback).

Atay-Kayis A, Massam H (2005). "A Monte Carlo method for computing the marginal likelihood in nondecomposable Gaussian graphical models." *Biometrika*, **92**(2), 317–335.

Baldi P, Brunak S, Chauvin Y, Andersen CA, Nielsen H (2000). "Assessing the Accuracy of Prediction Algorithms for Classification: an Overview." *Bioinformatics*, **16**(5), 412–424.

Bates D, Maechler M (2014). *Matrix: Sparse and Dense Matrix Classes and Methods*. R package version 1.1-2, URL http://CRAN.R-project.org/package=Matrix.

Behrouzi P, Wit EC (2017). "netgwas: An R Package for Network-Based Genome-Wide Association Studies." *arXiv preprint arXiv:1710.01236*.

Bhadra A, Mallick BK (2013). "Joint High-Dimensional Bayesian Variable and Covariance Selection with an Application to eQTL Analysis." *Biometrics*, **69**(2), 447–457.

Cappé O, Robert C, Rydén T (2003). "Reversible Jump, Birth-and-Death and More General Continuous Time Markov Chain Monte Carlo Samplers." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **65**(3), 679–700.

Csardi G, Nepusz T (2006). "The **igraph** Software Package for Complex Network Research." *InterJournal*, **Complex Systems**, 1695. URL http://igraph.org.

Dobra A, Lenkoski A (2011). "Copula Gaussian graphical models and their application to modeling functional disability data." *The Annals of Applied Statistics*, **5**(2A), 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 (2018). "Loglinear model selection and human mobility." *Ann. Appl. Stat.*, **12**(2), 815–845. doi:10.1214/18-AOAS1164. URL https://doi.org/10.1214/18-AOAS1164.

Finegold M, Drton M (2014). "Robust Bayesian Graphical Modeling Using Dirichlet t-Distributions." *Bayesian Analysis*, **9**(3), 521–550.

Foygel R, Drton M (2010). "Extended Bayesian Information Criteria for Gaussian Graphical Models." In J Lafferty, CKI Williams, J Shawe-Taylor, R Zemel, A Culotta (eds.), *Advances in Neural Information Processing Systems 23*, pp. 604–612.

Friedman J, Hastie T, Tibshirani R (2008). "Sparse Inverse Covariance Estimation with the Graphical Lasso." *Biostatistics*, **9**(3), 432–441.

Friedman J, Hastie T, Tibshirani R (2014). ***glasso**: Graphical Lasso- Estimation of Gaussian Graphical Models*. R package version 1.8, URL http://CRAN.R-project.org/package=glasso.

Giudici P, Green P (1999). "Decomposable graphical Gaussian model determination." *Biometrika*, **86**(4), 785–801.

Green PJ (1995). "Reversible jump Markov chain Monte Carlo computation and Bayesian model determination." *Biometrika*, **82**(4), 711–732.

Green PJ (2003). "Trans-dimensional markov chain monte carlo." *Oxford Statistical Science Series*, pp. 179–198.

Gu Q, Cao Y, Ning Y, Liu H (2015). "Local and Global Inference for High Dimensional Gaussian Copula Graphical Models." *arXiv preprint arXiv:1502.02347*.

Hastie T, Tibshirani R, Friedman J (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer-Verlag.

Hoff PD (2007). "Extending the Rank Likelihood for Semiparametric Copula Estimation." *The Annals of Applied Statistics*, pp. 265–283.

Hsieh CJ, Sustik MA, Dhillon IS, Ravikumar P (2011). "Sparse Inverse Covariance Matrix Estimation Using Quadratic Approximation." In J Shawe-Taylor, R Zemel, P Bartlett, F Pereira, K Weinberger (eds.), *Advances in Neural Information Processing Systems 24*, pp. 2330–2338. http://nips.cc/. URL http://cs.utexas.edu/users/sustik/QUIC.

Hsieh CJ, Sustik MA, Dhillon IS, Ravikumar P (2014). "QUIC: quadratic approximation for sparse inverse covariance estimation." *The Journal of Machine Learning Research*, **15**(1), 2911–2947.

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.

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

Lauritzen S (1996). *Graphical Models*, volume 17. Oxford University Press, USA.

Lawson CL, Hanson RJ, Kincaid DR, Krogh FT (1979). "Basic Linear Algebra Subprograms for Fortran Usage." *ACM Transactions on Mathematical Software (TOMS)*, **5**(3), 308–323.

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.

Liu H, Roeder K, Wasserman L (2010). "Stability Approach to Regularization Selection (StARS) for High Dimensional Graphical Models." In *Advances in Neural Information Processing Systems*, pp. 1432–1440.

Meinshausen N, Buhlmann P (2006). "High-Dimensional Graphs and Variable Selection with the Lasso." *The Annals of Statistics*, **34**(3), 1436–1462.

Mohammadi A, Abegaz Yazew F, van den Heuvel E, Wit EC (2017a). "Bayesian modelling of Dupuytren disease using Gaussian copula graphical models." *Journal of Royal Statistical Society-Series C*, **66**(3), 629–645.

Mohammadi A, Wit EC (2014). "Contributed Discussion on Article by Finegold and Drton." *Bayesian Analysis*, **9**(3), 577–579.

Mohammadi A, Wit EC (2015). "Bayesian Structure Learning in Sparse Gaussian Graphical Models." *Bayesian Analysis*, **10**(1), 109–138.

Mohammadi R (2018). *ssgraph: Bayesian Graphical Estimation using Spike-and-Slab Priors*. R package version 1.1.

Mohammadi R, Massam H, Gerald L (2017b). "The Ratio of Normalizing Constants for Bayesian Graphical Gaussian Model Selection." *arXiv preprint arXiv:1706.04416*.

Mohammadi R, Wit EC (2018). **BDgraph**: *Bayesian Structure Learning in Graphical Models using Birth-Death MCMC*. R package version 2.48, URL http://CRAN.R-project.org/package=BDgraph.

Muirhead R (1982). *Aspects of Multivariate Statistical Theory*, volume 42. Wiley Online Library.

Murray I, Ghahramani Z (2004). "Bayesian Learning in Undirected Graphical Models: Approximate MCMC Algorithms." In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pp. 392–399. AUAI Press.

Murray I, Ghahramani Z, MacKay D (2012). "MCMC for doubly-intractable distributions." *arXiv preprint arXiv:1206.6848*.

Nelsen RB (2007). *An Introduction to Copulas*. Springer Science & Business Media.

OpenMP Architecture Review Board (2008). "OpenMP Application Program Interface Version 3.0." URL http://www.openmp.org/mp-documents/spec30.pdf.

Pensar J, Nyman H, Niiranen J, Corander J, *et al.* (2017). "Marginal pseudo-likelihood learning of discrete Markov network structures." *Bayesian analysis*, **12**(4), 1195–1215.

R Core Team (2013). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.

Rolfs B, Rajaratnam B, Guillot D, Wong I, Maleki A (2012). "Iterative thresholding algorithm for sparse inverse covariance estimation." In *Advances in Neural Information Processing Systems*, pp. 1574–1582.

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.

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

Stranger BE, Nica AC, Forrest MS, Dimas A, Bird CP, Beazley C, Ingle CE, Dunning M, Flicek P, Koller D, *et al.* (2007). "Population Genomics of Human Gene Expression." *Nature genetics*, **39**(10), 1217–1224.

Tank A, Foti N, Fox E (2015). "Bayesian structure learning for stationary time series." *arXiv preprint arXiv:1505.03131*.

Uhler C, Lenkoski A, Richards D (2014). "Exact Formulas for the Normalizing Constants of Wishart Distributions for Graphical Models." *arXiv preprint arXiv:1406.4901*.

Vujačić I, Abbruzzo A, Wit EC (2015). "A computationally fast alternative to cross-validation in penalized Gaussian graphical models." *Journal of Statistical Computation and Simulation*, (ahead-of-print), 1–13.

Wang H, Li S (2012). "Efficient Gaussian Graphical Model Determination under G-Wishart Prior Distributions." *Electronic Journal of Statistics*, **6**, 168–198.

Wit EC, Abbruzzo A (2015a). "Factorial graphical models for dynamic networks." *Network Science*, **3**(01), 37–57.

Wit EC, Abbruzzo A (2015b). "Inferring slowly-changing dynamic gene-regulatory networks." *BMC bioinformatics*, **16**(Suppl 6), S5.

Zhao T, Liu H, Roeder K, Lafferty J, Wasserman L (2012). "The **huge** package for high-dimensional undirected graph estimation in R." *The Journal of Machine Learning Research*, **13**(1), 1059–1062.

Zhao T, Liu H, Roeder K, Lafferty J, Wasserman L (2014). ***huge**: High-dimensional Undirected Graph Estimation*. R package version 1.2.6, URL http://CRAN.R-project.org/package=huge.

**Affiliation:**

Reza Mohammadi,
Operation Management Section,
Faculty of Economics end Business,
University of Amsterdam,
Amsterdam, Netherlands,
E-mail: a.mohammadi@uva.nl
URL: http://www.uva.nl/profile/a.mohammadi

Ernst C. Wit,
Institute of Computational Science,
Universita della Svizzera Italiana,
Lugano, Switzerland,
E-mail: e.c.wit@rug.nl
URL: http://www.math.rug.nl/~ernst/