An Introduction to the BDgraph for Bayesian Graphical Models

Reza Mohammadi
University of Amsterdam

Ernst C. Wit
Universita della Svizzera Italiana

Abstract

This introduction to the R package **BDgraph** is a (slightly) modified version of Mohammadi and Wit (2019a), published in the *Journal of Statistical Software*.

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) and Dobra and Mohammadi (2018). To speed up computations, the computationally intensive tasks have been implemented in C++ and interfaced with R, and the package has parallel computing capabilities. In addition, the package contains several functions for simulation and visualization, as well as several 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.

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, Abegaz Yazew, van den Heuvel, and Wit 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; Dyrba, Grothe, Mohammadi, Binder, Kirste, Teipel, Initiative *et al.* 2018; Behrouzi and Wit 2019). Bayesian approaches

provide a principled alternative to various penalized approaches.

In this paper, we describe the **BDgraph** package (Mohammadi and Wit 2019b) in R (R Core Team 2019) 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); Dobra and Mohammadi (2018); Lenkoski (2013); Mohammadi, Massam, and Gerald (2017b); 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 $\geq$ 3) from the Comprehensive R Archive Network (CRAN) at `http://cran.r-project.org/packages=BDgraph`.

In the Bayesian literature, the **BDgraph** is one of the few R packages which is available online for Gaussian graphical models and Gaussian copula graphical models. Another R package is **ssgraph** (Mohammadi 2019) which is based on spike-and-slab proir. On the other hand, more packages seem to be available in the frequentist literature. The existing packages include **huge** (Zhao, Liu, Roeder, Lafferty, and Wasserman 2019), **glasso** (Friedman, Hastie, and Tibshirani 2018), **bnlearn** (Scutari 2010), **pcalg** (Kalisch, Mächler, Colombo, Maathuis, and Bühlmann 2012), **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.

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 **BDgraph** package we automatically load the **igraph** (Csardi and Nepusz 2006) package, since the **BDgraph** package depends on this package for graph visualization. The **igraph** package is available on the Comprehensive R Archive Network (CRAN) at `http://CRAN.R-project.org`.

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** (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:

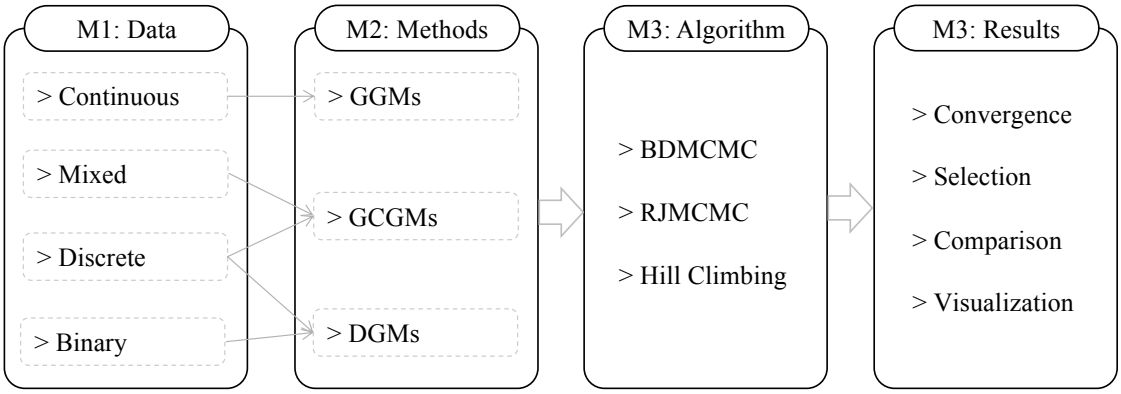

Figure 1: Configuration of the **BDgraph** package which includes three main parts: (M1) data simulation, (M2) several statistical methods, (M3) several search algorithms, (M4) various functions to evaluate convergence of the search algorithms, estimation of the true graph, assessment and comparison of the results and graph visualization.

Module 1. Data simulation: Function `bdgraph.sim` simulates multivariate Gaussian, discrete, binary, and mixed data with different undirected graph structures, including `"random"`, `"cluster"`, `"scale-free"`, `"lattice"`, `"hub"`, `"star"`, `"circle"`, `"AR(1)"`, `"AR(2)"`, 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. Methods: The function `bdgraph` and `bdgraph.mpl` provide several estimation methods regarding to the type of data:

- Bayesian graph estimation for the multivariate data that follow the Gaussianity assumption, based on the Gaussian graphical models (GGMs); see Mohammadi and Wit (2015); Dobra *et al.* (2011).
- Bayesian graph estimation for multivariate non-Gaussian, discrete, and mixed data, based on Gaussian copula graphical models (GCGMs); see Mohammadi *et al.* (2017a); Dobra and Lenkoski (2011).
- Bayesian graph estimation for multivariate discrete and binary data, based on discrete graphical models (DGMs); see Dobra and Mohammadi (2018).

Module 3. Algorithms: The function `bdgraph` and `bdgraph.mpl` provide several sampling algorithms:

- Birth-death MCMC (BDMCMC) sampling algorithms (Algorithms 2 and 3) desciribed in Mohammadi and Wit (2015).
- Reversible jump MCMC (RJMCMC) sampling algorithms desciribed in Dobra and Lenkoski (2011).
- Hill-climbing (HC) search algorithm desciribed in Pensar *et al.* (2017).

Module 4. Results: 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

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

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

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

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), the Laplace approximation (Lenkoski and Dobra 2011), or recent approximation by Mohammadi *et al.* (2017b). 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

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

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

```
     [,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 Ω_{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 = adj, b = 3, D = diag( 3 ) )
R> round( sample, 2 )

      [,1] [,2]  [,3]
[1,]  2.37 0.00 -2.12
[2,]  0.00 6.15  0.00
[3,] -2.12 0.00  7.26
```

This matrix is a sample from a G-Wishart distribution with $b = 3$ and $D = I_3$ as an identity matrix and a graph structure with adjacency matrix `adj`.

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 prior distribution of the graph $G = (V, E)$, we consider a Bernoulli prior on each link inclusion indicator variable as follow

$$Pr(G) \propto \left(\frac{\theta}{1-\theta}\right)^{|E|}, \tag{4}$$

where $|E|$ indicate the number of links in the graph G (graph size) and parameter $\theta \in (0, 1)$ is a prior probability of existing link. For the case $\theta = 0.5$ (as a default option of the **BDgraph**), we will have a uniform distribution over all graph space, 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

$$Pr(\text{birth of link } e) = \frac{\beta_e(K)}{\beta(K) + \delta(K)}, \qquad \text{for each} \quad e \notin E, \tag{5}$$

$$Pr(\text{death of link } e) = \frac{\delta_e(K)}{\beta(K) + \delta(K)}, \qquad \text{for each} \quad e \in E. \tag{6}$$

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

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

$$\delta_e(K) = \min\left\{\frac{Pr(G^{-e}, K^{-e}|\mathbf{Z})}{Pr(G, K|\mathbf{Z})}, 1\right\}, \quad \text{for each} \quad e \in E, \tag{8}$$

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 *et al.* (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.

for N iteration **do**
  1. Sample from the graph. Based on birth and death process
    1.1. Calculate the birth rates by (7) and $\beta(K) = \sum_{e \notin E} \beta_e(K)$
    1.2. Calculate the death rates by (8) and $\delta(K) = \sum_{e \in E} \delta_e(K)$
    1.3. Calculate the waiting time by $W(K) = 1/(\beta(K) + \delta(K))$
    1.4. Simulate the type of jump (birth or death) by (5) and (6)
  2. Sample from the precision matrix. By using Algorithm 1.
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 the Rao-Blackwellized estimator, for example, one can estimate the posterior distribution of the graphs proportional to the total waiting times of each graph.

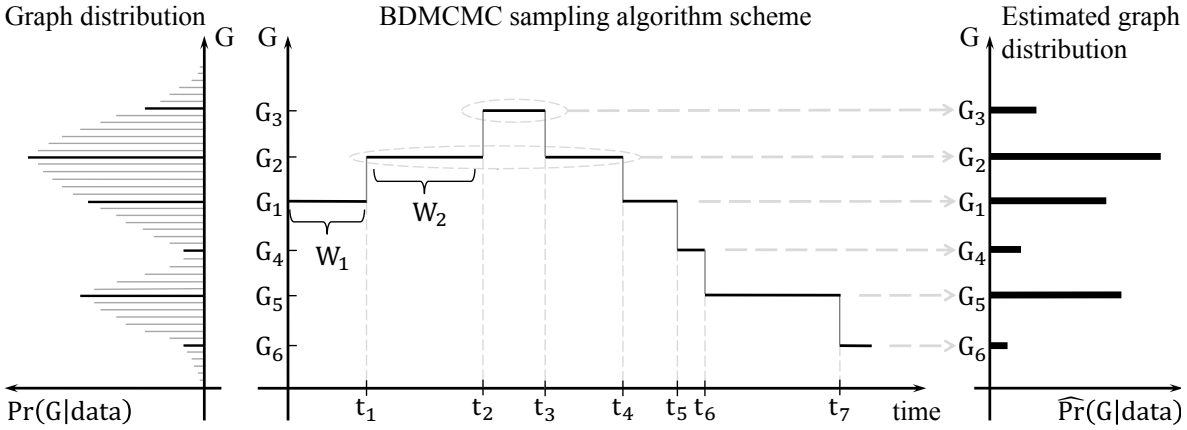

Figure 2: This image visualizes the Algorithm 2. The left side shows the true posterior distribution of the graph. The middle panel presents a continuous time BDMCMC sampling algorithm where $\{W_1, W_2, ...\}$ denote waiting times and $\{t_1, t_2, ...\}$ denote jumping times. The right side denotes the estimated posterior probability of the graphs in proportion to the total of their waiting times, according to the Rao-Blackwellized estimator.

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:

$$\begin{aligned} Z_1, ..., Z_n &\overset{iid}{\sim} \mathcal{N}_p(0, \Gamma(K)), \\ Y_{ij} &= F_j^{-1}(\Phi(Z_{ij})), \end{aligned} \tag{9}$$

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

$$Pr\left(Y_1 \leq Y_1, \ldots, Y_p \leq Y_p\right) = C(F_1(Y_1), \ldots, F_p(Y_p) \mid \Gamma(K)), \tag{10}$$

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 (10) 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

$$L_j^r(\mathbf{Z}) = \max\left\{Z_j^{(s)} : Y_j^{(s)} < Y_j^{(r)}\right\} \text{ and } U_j^r(\mathbf{Z}) = \min\left\{Z_j^{(s)} : Y_j^{(r)} < Y_j^{(s)}\right\}. \tag{11}$$

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

$$Pr(\mathbf{Z} \in \mathcal{D}(\mathbf{Y}) \mid K, G) = Pr(\mathbf{Z} \in \mathcal{D}(\mathbf{Y}) \mid K, G) = \int_{\mathcal{D}(\mathbf{Y})} Pr(\mathbf{Z} \mid K, G) dZ \tag{12}$$

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

$$Pr(K, G|Z \in \mathcal{D}(\mathbf{Y})) \propto Pr(K, G) Pr(Z \in \mathcal{D}(\mathbf{Y})|K, G). \tag{13}$$

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 Mohammadi and Wit (2019a, Section 6.2).

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), (13), 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 (11).

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**

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 joint posterior distribution of the graph and the parameters of the graph, e.g., 3 for GGMs and 13 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. Besides, we using the result of Mohammadi *et al.* (2017b) for the ratio of the normalizing constant of G-Wishart distribution.

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

```
bdgraph( data, n = NULL, method = "ggm", algorithm = "bdmcmc", iter = 5000,
  burnin = iter / 2, not.cont = NULL, g.prior = 0.5, df.prior = 3,
  g.start = "empty", jump = NULL, save = FALSE, print = 1000, cores = NULL,
  threshold = 1e-8 )
```

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, GCGMs. Option "`ggm`" is based on Gaussian graphical models (Algorithm 2) that is designed for multivariate Gaussian data. Option "`gcgm`" is based on the GCGMs (Algorithm 3) that 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 Mohammadi and Wit (2015, Section 4), Mohammadi *et al.* (2017a, Section 2.2.3).

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 `jump` determines the number of links that are simultaneously updated in the BDMCMC algorithm.

For parallel computation in `C++` which is based on **OpenMP** (Board 2008), user can use argument `cores` which specifies the number of cores to use for parallel execution.

Note, the package **BDgraph** has two other sampling functions, `bdgraph.mpl` and `bdgraph.ts` which are designed in the similar framework as the function `bdgraph`. 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). The function `bdgraph.ts` is for Bayesian model determination in time series graphical models (Tank, Foti, and Fox 2015).

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

```
plinks( bdgraph.obj, round = 2, burnin = NULL )
```

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

$$Pr(e \in E|data) = \frac{\sum_{t=1}^{N} 1(e \in E^{(t)}) W(K^{(t)})}{\sum_{t=1}^{N} W(K^{(t)})}, \tag{14}$$

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

```
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 = 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 Mohammadi and Wit (2019a, Appendix).

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

```
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

```
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

```
compare( target, est, est2 = NULL, est3 = NULL, est4 = NULL, main = 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:

$$F_1\text{-score} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}}, \tag{15}$$

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

```
plotroc( target, est, est2 = NULL, est3 = NULL, est4 = 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

```
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, binary and mixed data with different undirected graph structures, including `"random"`, `"cluster"`, `"scale-free"`, `"lattice"`, `"hub"`, `"star"`, `"circle"`, `"AR(1)"`, `"AR(2)"`, 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.

There is another function in the **BDgraph** package with the name `graph.sim` which is designed to simulate different types of graph structures. The function

```
graph.sim( p = 10, graph = "random", prob = 0.2, size = NULL, class = NULL,
  cut = 4, vis = FALSE )
```

can simulate different undirected graph structures, including `"random"`, `"cluster"`, `"scale-free"`', `"lattice"`, `"hub"`, `"star"`, and `"circle"` graphs. Users can specify the type of graph structure by option `graph`. They can determine the sparsity level of the obtained graph by using option `prob`. `bdgraph.sim` returns an object of the `S3` class type "`graph`". Functions `plot` and `print` work with this object type.

5. An example on simulated data

We illustrate the user interface of the **BDgraph** package by use of a simple simulation. We perform all the computations on an MacBook Pro with 2.9 GHz Intel Core i7 processor. By using the function `bdgraph.sim` we simulate 60 observations ($n = 60$) from a multivariate Gaussian distribution with 8 variables ($p = 8$) and "scale-free" graph structure, as below.

```
R> data.sim <- bdgraph.sim( n = 60, p = 8, graph = "scale-free",
+                           type = "Gaussian" )
R> round( head( data.sim $ data, 4 ), 2 )

      [,1]  [,2]  [,3]  [,4]  [,5]  [,6]  [,7]  [,8]
[1,]  0.72 -0.91 -1.23 -0.16  0.20 -0.47  0.08  1.07
[2,]  0.25 -0.11  0.09  0.53  0.10 -0.04 -0.13 -0.67
[3,] -0.42 -0.09 -0.28 -0.42  2.04  0.84 -0.79  1.24
[4,] -0.33 -0.50  0.68 -1.33 -1.15  0.25 -0.35  2.97
```

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( bdgraph( data = data.sim, method = "ggm",
+                            algorithm = "bdmcmc", iter = 5000, save = TRUE )
```

We choose option "`save = TRUE`" to save the samples in order to check convergence of the algorithm. Running this function takes less than one second, 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] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
[1,]    0    1    1    0    0    0    1    0
[2,]    0    0    0    1    0    0    0    0
[3,]    0    0    0    0    0    1    0    0
[4,]    0    0    0    0    0    0    0    1
```

```
[5,]    0    0    0    0    0    0    0    0
[6,]    0    0    0    0    0    0    0    0
[7,]    0    0    0    0    0    0    0    0
[8,]    0    0    0    0    0    0    0    0

$p_links
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
[1,]    0 0.51 1.00 0.27 0.21 0.31 0.74 0.11
[2,]    0 0.00 0.29 1.00 0.25 0.18 0.49 0.14
[3,]    0 0.00 0.00 0.24 0.27 0.79 0.44 0.22
[4,]    0 0.00 0.00 0.00 0.32 0.30 0.34 1.00
[5,]    0 0.00 0.00 0.00 0.00 0.25 0.40 0.22
[6,]    0 0.00 0.00 0.00 0.00 0.00 0.23 0.37
[7,]    0 0.00 0.00 0.00 0.00 0.00 0.00 0.19
[8,]    0 0.00 0.00 0.00 0.00 0.00 0.00 0.00

$K_hat
      [,1]  [,2]  [,3]  [,4]  [,5]  [,6]  [,7]  [,8]
[1,]  3.81  0.33  3.19 -0.09  0.04  0.14 -0.84  0.02
[2,]  0.33  4.24 -0.06 -3.43 -0.07 -0.02  0.41 -0.02
[3,]  3.19 -0.06  5.54 -0.08 -0.06 -0.75  0.41  0.08
[4,] -0.09 -3.43 -0.08  9.28 -0.15  0.10 -0.18  1.62
[5,]  0.04 -0.07 -0.06 -0.15  0.76 -0.06  0.16 -0.04
[6,]  0.14 -0.02 -0.75  0.10 -0.06  3.08  0.04 -0.14
[7,] -0.84  0.41  0.41 -0.18  0.16  0.04  5.56  0.04
[8,]  0.02 -0.02  0.08  1.62 -0.04 -0.14  0.04  1.21
```

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 2000 different graphs. 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 4 and 18 links. At 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, method = "ggm",
+                            algorithm = "rjmcmc", iter = 5000, save = TRUE )
```

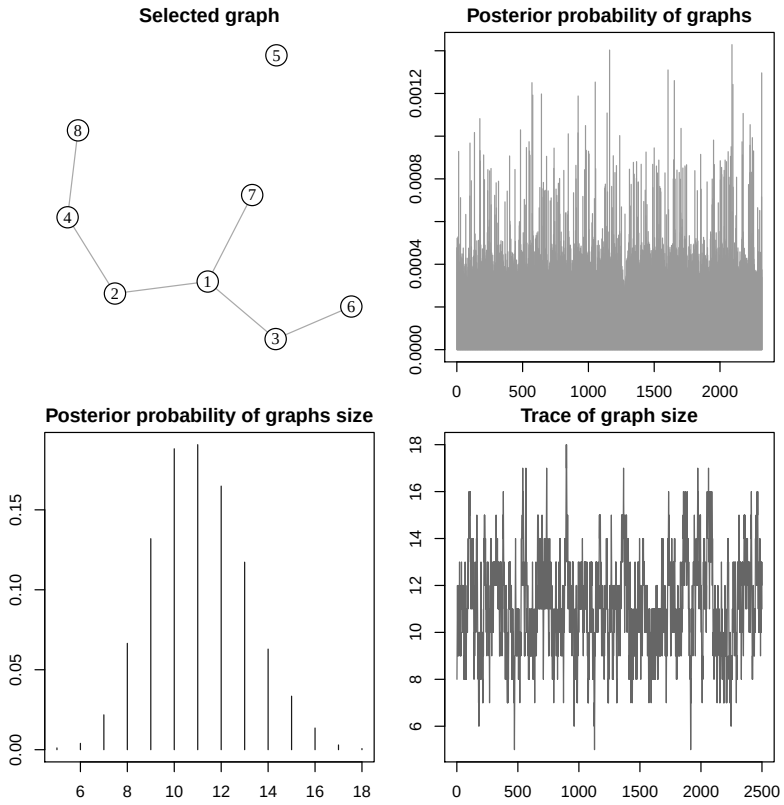

Figure 3: Visualization summary of simulation data based on output of `bdgraph` function. The figure at the top-left is the inferred graph with the highest posterior probability. The figure at the top-right gives the estimated posterior probabilities of all visited graphs. The figure at the bottom-left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

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, smooth = TRUE )
```

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

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

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

                True graph BDMCMC RJMCMC
true positive        7  5.000  5.000
true negative       21 20.000 19.000
```

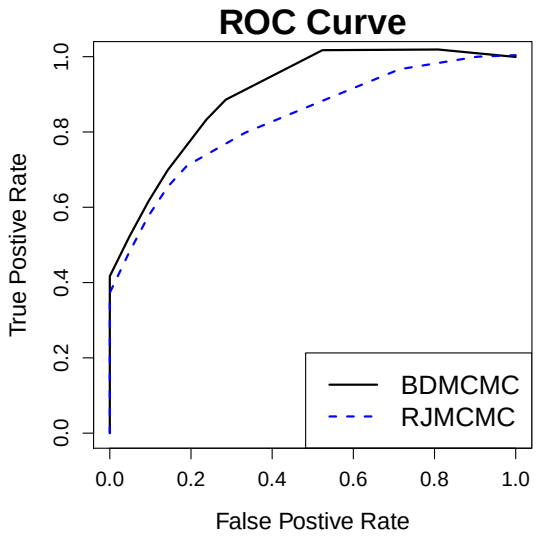

Figure 4: ROC plot to compare the performance of the BDMCMC and RJMCMC algorithms for a simulated toy example.

```
false positive      0  1.000  2.000
false negative      0  2.000  2.000
F1-score            1  0.769  0.714
specificity         1  0.952  0.905
sensitivity         1  0.714  0.714
MCC                 1  0.704  0.619
```

The results show that for this specific simulated example both algorithms have more or less the same performance; See Mohammadi and Wit (2015, Section 4), Mohammadi *et al.* (2017a, Section 2.2.3).

In this simulation example, we run both BDMCMC and RJMCMC algorithms for $5,000$ iterations, $2,500$ 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.

References

Abegaz F, Wit E (2015). "Copula Gaussian graphical models with penalized ascent Monte Carlo EM algorithm." *Statistica Neerlandica*, **69**(4), 419–441. doi:10.1111/stan.12066.

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.

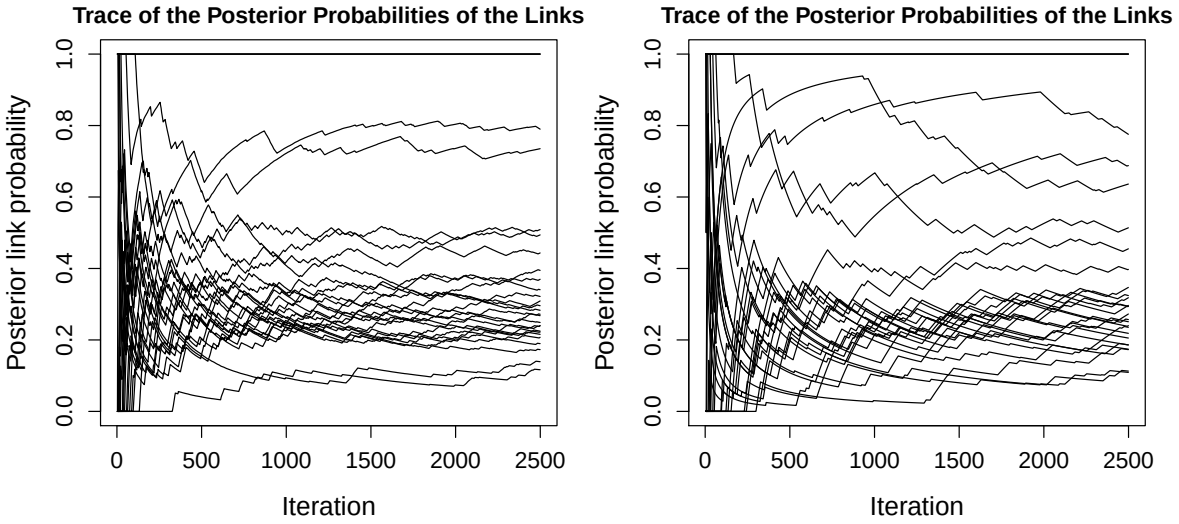

Figure 5: Plot for monitoring the convergence based on the trace of estimated posterior probability of all possible links for the BDMCMC algorithm (left) and the RJMCMC algorithm (right).

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. doi:10.1093/biomet/92.2.317.

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. doi:10.1093/bioinformatics/16.5.412.

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

Behrouzi P, Wit EC (2019). "Detecting epistatic selection with partially observed genotype data by using copula graphical models." *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, **68**(1), 141–160. doi:10.1111/rssc.12287.

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

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. doi:10.1111/1467-9868.00409.

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. doi:10.1214/10-aoas397.

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. doi:10.1198/jasa.2011.tm10465.

Dobra A, Mohammadi R (2018). "Loglinear model selection and human mobility." *The Annals of Applied Statistics*, **12**(2), 815–845.

Dyrba M, Grothe MJ, Mohammadi A, Binder H, Kirste T, Teipel SJ, Initiative ADN, *et al.* (2018). "Comparison of different hypotheses regarding the spread of Alzheimer's disease using Markov random fields and multimodal imaging." *Journal of Alzheimer's Disease*, **65**(3), 731–746. doi:10.3233/jad-161197.

Friedman J, Hastie T, Tibshirani R (2008). "Sparse Inverse Covariance Estimation with the Graphical Lasso." *Biostatistics*, **9**(3), 432–441. doi:10.1093/biostatistics/kxm045.

Friedman J, Hastie T, Tibshirani R (2018). ***glasso****: Graphical Lasso- Estimation of Gaussian Graphical Models*. R package version 1.10, URL http://CRAN.R-project.org/package=glasso.

Giudici P, Green P (1999). "Decomposable graphical Gaussian model determination." *Biometrika*, **86**(4), 785–801. doi:10.1093/biomet/86.4.785.

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

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. doi:10.1214/07-aoas107.

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. doi:10.1214/088342305000000304.

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. doi:10.18637/jss.v047.i11.

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. doi:10.1145/355841.355847.

Lenkoski A (2013). "A Direct Sampler for G-Wishart Variates." *Stat*, **2**(1), 119–128. doi:10.1002/sta4.23.

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. doi:10.1198/jcgs.2010.08181.

Meinshausen N, Buhlmann P (2006). "High-Dimensional Graphs and Variable Selection with the Lasso." *The Annals of Statistics*, **34**(3), 1436–1462. doi:10.1214/009053606000000281.

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. doi:10.1111/rssc.12171.

Mohammadi A, Wit EC (2015). "Bayesian Structure Learning in Sparse Gaussian Graphical Models." *Bayesian Analysis*, **10**(1), 109–138. doi:10.1214/14-ba889.

Mohammadi R (2019). ***ssgraph**: Bayesian Graphical Estimation using Spike-and-Slab Priors*. R package version 1.8.

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 (2019a). "BDgraph: An R Package for Bayesian Structure Learning in Graphical Models." *Journal of Statistical Software*, **89**(3), 1–30. doi:10.18637/jss.v089.i03.

Mohammadi R, Wit EC (2019b). ***BDgraph**: Bayesian Structure Learning in Graphical Models using Birth-Death MCMC*. R package version 2.60, URL http://CRAN.R-project.org/package=BDgraph.

Muirhead R (1982). *Aspects of Multivariate Statistical Theory*, volume 42. Wiley Online Library. doi:10.1002/9780470316559.

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.

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. doi:10.1214/16-ba1032.

R Core Team (2019). *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. doi:10.1111/1467-9469.00297.

Scutari M (2010). "Learning Bayesian Networks with the **bnlearn** R Package." *Journal of Statistical Software*, **35**(3), 22. doi:10.18637/jss.v035.i03.

Tank A, Foti N, Fox E (2015). "Bayesian structure learning for stationary time series." *In Proceedings of the 31st Conference on Uncertainty in Artificial Intelligence*, pp. 872–881.

Wang H, Li S (2012). "Efficient Gaussian Graphical Model Determination under G-Wishart Prior Distributions." *Electronic Journal of Statistics*, **6**, 168–198. doi:10.1214/12-ejs669.

Wit EC, Abbruzzo A (2015a). "Factorial graphical models for dynamic networks." *Network Science*, **3**(01), 37–57. doi:10.1017/nws.2015.2.

Wit EC, Abbruzzo A (2015b). "Inferring slowly-changing dynamic gene-regulatory networks." *BMC bioinformatics*, **16**(Suppl 6), S5. doi:10.1186/1471-2105-16-s6-s5.

Zhao T, Liu H, Roeder K, Lafferty J, Wasserman L (2019). ***huge**: High-dimensional Undirected Graph Estimation*. R package version 1.3.2, 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/