2021

Statistical methods for patterns and interconnections between variables with applications in epigenetics data

jiajing wang

Statistical methods for patterns and interconnections between variables with applications in epigenetics data

by

Jiajing Wang

A Dissertation

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Doctor of Philosophy

Major: Applied Statistics

Department of Mathematical Sciences

The University of Memphis

December 2021

ACKNOWLEDGMENTS

First of all, I would like to express my sincere gratitude to my supervisors, Prof. Hongmei Zhang and Prof. E. Olúgun George, for their guidance, patience, and motivation throughout my graduate career. Their thorough scientific knowledge and insightful expert advice have benefited me a lot during my research journey. Sincere gratitude is extended to their generous participation in guiding, constructive feedback, kind support, and advice during my PhD. I'm also grateful to the rest of my committee members, Prof. Dale Bowman, Prof. Meredith Ray and Dr. Shengtong Han, for the time, efforts and valuable advice on my dissertation. Without their help, I would not have made this dissertation.

I also own thanks to all the faculty and staff members of the applied statistics program in Department of Mathematical Sciences at University of Memphis. I am grateful to Prof. Su Chen, Prof. Lih-Yuan Deng, for important helping on the statistical courses. I also thanks to my supervisor and colleagues at Cedars Sinai, for their flexibility and support during my PhD study.

Finally, I would like to thank my parents and families. Your love and support make this work meaningful. I also thank for all the friends in Memphis, they gave me great love and encourage me to keeping working on this work.

Abstract

Epigenetics is one possible mechanism explaining the disease heritability without changing DNA sequence. DNA methylation (DNAm), as an epigenetic factor with a memory of environmental exposure, may potentially explain the existing missing heritabability. Learning the patterns of DNAm transmission at different CpG sites and studying the interconnection among the transmitted CpG sites is beneficial to disease prediction and prevention. In my dissertation, I will focus on two perspectives to study the transmission of DNA methylation: clustering and network.

The first project describes a nested Bayesian clustering method to identify DNA methylation (DNAm) sites (CpG sites) such that DNAm is transmitted from one generation to the next, and to study heterogeneity among CpG sites with DNAm transmitted. To facilitate this goal, the beta regression is employed to infer the transmission status and, for CpG sites with DNAm transmitted, to cluster transmission patterns at a population level. The transmission status and patterns are inferred under a Bayesian framework. Simulations with different scenarios are used to demonstrate and evaluate the applicability of the method. We demonstrate the approach using a triad (mother, father, and offspring) data set with DNA methylation assessed at 4063 CpG sites to detect transmitted CpGs and their DNAm transmission patterns.

The second project proposes a comprehensive comparison of three existing Gaussian graphical models on epigenetic network constructions based on the precision matrix. The three methods, the projection method, the horseshoe method and the HRS (hit and run sampler) method, are assessed in different scenarios, and six statistics, sensitivity, specificity, MCC, F1-score, KL-divergence and quadractic loss, are used to compare the performance across different approaches. The simulation study suggests that both projection method and the horseshoe method performed well in edge set identification in

low-dimensional setting, but a higher loss in precision matrix estimation, and the HRS method always performed well in both graphical structure identification and precision matrix estimation in high-dimensional setting. The three methods are further applied in 1043 CpGs that are maternal-transmission dominated to identify the potential network structures.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Epigenetics is a mechanisms potentially heritable that regulates expression of genes without changing DNA sequence [1], and the epigenetics changes are crucial for the development and differentiation of distinct cell lineages in an organism [2]. For many diseases, e.g., asthma [3], eczema [4], genetic variants could explain a small fraction of disease heritability [5, 6]. Various explanations for the missing heritability phenomenon have been suggested, including much larger numbers of variants of small effect yet to be found; low power to detect gene-gene interactions; and inadequate accounting for shared environment among relatives [7]. The potentially inheritable DNA methylation (DNAm), suggested to be an epigenetic factor with a memory of environmental exposure, is among the factors that may explain the missing heritability. DNAm occurs predominantly at Cytosines within CG dinucleotide (CpG sites). It has been found that DNAm can be inherited transgenerationally [8] and is associated with health outcomes, including allergic and autoimmune diseases [9], such as asthma and eczema during childhood and adolescence. We may expect that asymmetric transmission patterns of DNA methylation from parents to offspring in the general population is beneficial to disease prediction and prevention [10].

Learning the patterns of DNA methylation transmission at different CpG sites and studying the interconnection among the transmitted CpG sites are thus greatly needed. In this dissertation, I proposed a nested Bayesian clustering method to identify the transmission patterns via clustering CpG sites showing different transmission patterns and proposed a comprehensive comparison of three existing Gaussian graphical models on epigenetic network constructions for those CpG sites showing similar patterns. In the following, I describe the motivation and related background of these two topics.

1.1 Clustering

Recently, several studies have shown that the probability of atopic disease transfer from one affected mother is roughly four times higher than from an affected father [11], which indicates epigenetic marks are potentially heritable and there may be an asymmetric transmission patterns of epigenetic marks from parents to their offspring. To study this epigenetic mechanism process, some clustering methods were developed to identify the patterns of DNA methylation transmission [12]. However, to our knowledge, currently available approaches, such as the method in Han et al [12], detect patterns based on candidate CpG sites selected by size of correlations between offpsring and each parent without formally assessing heritability. Doing so may potentially mis-classify non-transmitted CpG sites as being transmitted, and vice versa. There is a desire first to sort out CpG sites which are transmitted from one generation to the next at the population level and then among the transmitted CpGs to study the transmission patterns. For both tasks, detecting transmission status and transmission patterns, the concept of probabilistic clustering analyses can be applied.

Classic clustering methods are separated by unsupervised approaches and the model-based approaches. Unsupervised approaches, such as K-means algorithms, or various hierarchical clustering methods, are not able to evaluate the strength of inheritance while clustering. Research findings showed that model-based clustering techniques are often superior to non-parametric approaches [13]. The existing model-based clustering methods, such as the Bernoulli-lognormal mixture model [13] and recursively-partitioned beta mixture model [14], focus on associations among individual subjects and are not able to identify the transmission patterns at the population level. To the best of our knowledge, very limited contribution has been made to this field except for the work in Han et al.[12], which focuses on transmission patterns detection only and does not assess the status of transmission.

I propose a nested Bayesian clustering method to infer transmission status, and among

transmitted CpG sites, we examined DNAm heterogeneity via clustering CpG sites showing different transmission patterns. In the following, I review in more detail different clustering methods and present the framework of Bayesian inferences.

1.1.1 Clustering methods

Clustering is the method for dividing a population or set of data points into a number of groups so that data points in the same groups are more similar than those in other groups. The clustering methods are separated by non-parametric and parametric clustering approaches. Non-parametric clustering approaches are in general unsupervised methods based on data partitioning or to group the data hierarchically. Parametric approaches in general is model-based method. The goal of the parametric approaches is to optimize the distribution mixture to determine the number of clusters [15]. The commonly used clustering methods in both two directions are reviewed as follows.

Non-parametric cluster analyses identify clusters based on distance between clustering objects. To define distances, the Pearson sample correlations, Euclidean distance and Manhattan distance are often used for continuous variables, and Gower distances are commonly used for a mixture of continuous and categorical variables. With distance metric defined, objects are grouped together if they are close to each other based on their distance to a cluster center or distances between each objects [15]. Three clustering techniques including partitioning-based methods, hierarchical clustering methods and a hybrid of these two approached are commonly applied.

Partitioning-based methods aim to maximize specific criteria to map vectors to clusters [15]. The most used partitioning-based clustering technique is K-means approach, which was first introduced by Cox [16] and then improved by Hartigan and Wong [17]. It starts from specifying the number of clusters, K, and initial centroid of each cluster. Let $\boldsymbol{c}$ with length K denote a collection of centroids, $\boldsymbol{c} = \{\boldsymbol{c_1}, \ldots, \boldsymbol{c_k}, \ldots, \boldsymbol{c_K}\}$ with $\boldsymbol{c_k} = \{c_{k1}, \ldots, c_{kp}\}$ being the centroid of cluster k for p components. Each data point's distance from the cluster center is determined, and each data point is assigned to a cluster based on

$argmin_{c_k \in \boldsymbol{c}} d(\boldsymbol{c_k}, \boldsymbol{x})$ (x denotes a data point with length p). Then the centroid of each cluster is updated based on the mean of the data points in each cluster [15]. The process continues until there is no data point is reassigned to a different cluster. To determine the number of clusters, the K is chosen by optimizing the total within-cluster variation, which is defined as

$$SS_{within} = \sum_{\boldsymbol{x_i} \in C_k} (\boldsymbol{x_i} - \boldsymbol{c_k})^T (\boldsymbol{x_i} - \boldsymbol{c_k}),$$

where C_k denotes cluster k, and $\boldsymbol{x_i}$ the ith data point. The number of clusters can be determined by the elbow rule, that is, plotting a set of SS_{within} with varing number of clusters, and selecting k when SS_{within} experiences the greatest reduction followed by a smoothed out SS_{within} reduction [15]. The hierarchical clustering methods is an algorithm that builds hierarchy of clusters. There are two methods in hierarchical clustering, agglomerative and divisive clustering. Agglomerative clustering starts from singleton clusters, and based on certain criteria, these singleton clusters are merged by cluster pairs until all clusters are merged into one single cluster [15]. There are several commonly used criteria to merge the clusters, such as single linkage, complete linkage, average linkage, Ward's method, and Centroid method. The single linkage focuses on the minimal distances between the data points, while the complete linkage is defined as the maximum distance between every pair of individuals. The average linkage focuses on average of distances between clusters, which is a weighted average distance between individuals in the clusters; Ward's method is to calculate the sum of squared Euclidean distance from the mean vector of each cluster based on total variations in the data, and the Centroid method is to evaluate distance between clusters using centroids. On the other hand, divisive clustering is to cluster the data points in an opposite way. This clustering method starts from one cluster including all clustering objects, and then the single cluster breaks into two clusters depending on the criteria listed above except for the complete linkage.

Parametric approaches is to cluster the objects based on the model. One of the commonly used parametric approach is to cluster objects based on associations, which is

commonly examined by linear regressions. This type of parametric clustering can be applied to identify gene or other genetic features [15]. Let $\boldsymbol{y}$ is a $(n \times K)$ matrix, denotes K variables with n subjects. The matrix $\boldsymbol{x}$ with dimension of $n \times m$ denotes m covariates for each subject. Suppose the linear regression is

$$y_{ik}|(\boldsymbol{\mu_k} = c, \boldsymbol{x}_{ik}) = \boldsymbol{x}_{ik}^{\boldsymbol{T}}\boldsymbol{\beta_c} + \varepsilon_{ik}$$

$$\varepsilon_{ik} \sim N(0, \sigma_c^2),$$

where $\boldsymbol{\mu_k} = (\mu_{k,1}, \ldots, \mu_{k,C})$ is a vector of indicators denoting which cluster variable k belongs to, and σ_c^2 is the variance of random error ε for cluster c. To put the K variables into C clusters, Qin and Self [18] proposed an expectation-maximization (EM) algorithm to infer the parameters including cluster assignment. The number of clusters is determined by Bayesian information criterion (BIC) and bootstrapped maximum volumn (BMV), $BMV_c = max\{volume(\hat{\Sigma}_c), c = 1, \ldots, C\}$, where $\hat{\Sigma}_c$ is an estimate of covariance matrix of the regression coefficients for each of the C clusters, which used to measure the stability of cluster centers for each given C [15]. The number of clusters is selected when clusters with a large BIC and small BMV.

1.1.2 Bayesian inferences

The Bayesian method is built upon the Bayes' rule.

$$p(\boldsymbol{\theta}|y) = \frac{p(y|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(y)},$$

where y might be a data point and $\boldsymbol{\theta}$ is some model parameters. The term $p(\boldsymbol{\theta}|y)$ represents a conditional probability or posterior distribution, which is the probability that the model parameters $(\boldsymbol{\theta})$ is computed conditional on the data (y). The term $p(y|\boldsymbol{\theta})$ represents the conditional probability of the data given the model's parameters, and it represents the likelihood function. The term $p(\boldsymbol{\theta})$ represents the probability of particular model parameter values in the population and is termed as the prior distribution. The term $p(y)$, representing the marginal distribution, is a normalizing factor and is calculated by

integrating over all values of $\boldsymbol{\theta}$ of the product $p(y|\boldsymbol{\theta})p(\boldsymbol{\theta})$. This term can be dropped from the equation as it does not depend on $\boldsymbol{\theta}$. Thus, based on the Bayes' rule, the posterior distribution is proportional to the likelihood function multiplied by the prior distribution,

$$p(\boldsymbol{\theta}|y) \propto p(y|\boldsymbol{\theta})p(\boldsymbol{\theta}),$$

which reflects one's updated knowledge, balancing prior distribution with observed data.

To construct a proper Bayesian model, the selection of prior distribution plays an important role on the Bayesian inferences. Prior distribution can have different levels of information, i.e. diffuse, weakly informative and informative priors, which reflects the information from relative uncertainty to complete certainty. In general, diffuse or weakly informative priors are preferred. However, a diffuse prior aligned with the likelihood can sometimes produce inaccurate or bias results with relatively flat priors [19]. Likewise, an informative prior with likelihood function at certain situations can provide a relative true result. To find a reasonable informative prior distribution, two strategies are always suggested. One is to incorporate expertise' knowledge when defining prior distributions, and the other is to use a data-based prior such that the hyperparameter of the prior distribution is obtained from the sample data using method such as maximum likelihood or sample statistics [20].

Bayesian inference is obtained from the posterior distribution by using sampling-based estimation methods. When sampling non-hierarchical Bayesian model, the samples can be drawn from the posterior distribution directly, especially if conjugate prior distributions are used. For complicated, unusual or in high dimensional models, simulations are commonly implemented to draw posterior inferences. The Markov Chain Monte Carlo (MCMC) method is a technique for sampling a set of parameter values from a posterior distribution using a Markov Chain and utilizing the Monte Carlo algorithm to generate a estimated posterior distribution and associated statistics with sampled parameters [21]. Recently, different MCMC-based sampling approaches have been developed, which were specified by the initial parameters values and transition kernel, so that the corresponding

stationary distribution converges to the correct posterior distribution. Metropolis-hastings, which is proposed by Metropolis et al [22] and Hastings et al [23], is an algorithm for obtaining random variables that employs a proposal distribution and an accept/reject step for the proposed parameter values. Reversible jump MCMC is an extension of the Metropolis-Hastings algorithm, which allows simulation of trans-dimensional moves within parameter space [24, 21]. Geman and Geman [25] conducted the Gibbs sampler method, which is a Metropolis-hastings algorithm by using the proposal distribution with an associated acceptance probability of 1. Hamiltonian Monte Carlo, which is developed by Duane [26], is a Metropolis-hastings algorithm based on Hamiltonian dynamics. In the dissertation, I will focus on the Metropolis-hastings algorithm and the Gibbs Sampler.

The Metropolis-hastings algorithm is a random walk with an acceptance/rejection mechanism that is used to converge to a target distribution $p(\theta|y)$ [27]. The Metropolis-hastings algorithm update the state θ from a distribution with probability $p(\theta)$ by using a transition kernel (proposal distribution) $g(\theta^*|\theta)$ as follows:

1. Sample a proposal θ^* from a proposal distribution $g(\theta^*|\theta)$.
2. Calculate the ratio of the density
$$r = \frac{p(\theta^*)p(\theta^{t-1}|\theta^*)}{p(\theta^{t-1})p(\theta^*|\theta^{t-1})}$$
3. Set
$$\theta^t = \begin{cases} \theta^* & \text{with probability min(r,1)} \\ \theta^{t-1} & \text{Otherwise} \end{cases} \tag{1.1}$$

If the proposal density is symmetric, for example $p(\theta_a|\theta_b) = p(\theta_b|\theta_a)$ for all θ_a, θ_b and t, then the Metropolis-hastings algorithm reduces to the Metropolis algorithm developed by Metropolis et at [22].

The Gibbs sampler is a special case of the Metropolis-hastings algorithm where the proposals are accepted with probability 1. Suppose the parameter value θ has been divided

into d components, $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_d)$. At each iteration of the Gibbs sampler cycles, each θ_j^t is sampled from the conditional distribution given all the other components of $\boldsymbol{\theta}$, $p(\theta_j|\theta_{-j}^{t-1}, y)$, where $\boldsymbol{\theta}_{-j} = (\theta_1^t, \ldots, \theta_{j-1}^t, \theta_{j+1}^{t-1}, \ldots, \theta_d^{t-1})$ [27]. The Gibbs sampler is used when the conditional distribution of each variable is known and is easy to sample from.

To assess the convergence of MCMC chains, some suggest a single long chain is adequate to explore the posterior density. However, many practitioners prefer to use two or more parallel chains with distinct starting values and various random seeds to provide greater initial variability across multiple chains, in order to make chains reach stationarity [21]. Convergence of multiple chains can be assessed using Gelman and Rubin [28] scale reduction factors. For each scalar estimated ψ, we label the simulation as $\psi_{ij}(i = 1, \ldots, n, j = 1, \ldots, m)$, and the between (B) and within (W) sequence variance is

$$B = \frac{n}{m-1} \sum_{j=1}^{m} (\overline{\psi_{\cdot j}} - \overline{\psi_{\cdot\cdot}})^2,$$

$$W = \frac{1}{m} \sum_{i=1}^{n} S_j^2, \text{where } S_j^2 = \sum_{i=1}^{n} (\overline{\psi_{ij}} - \overline{\psi_{\cdot j}})^2$$

Gelman and Rubin [28, 29] proposed to compare variation in the sample parameter values within and between chains. When assessing the convergence of MCMC sampling method, the effect sample size also need to be considered. Since the correlated parameter samples from the MCMC technique may affect MCMC samples convergence, the effect sample size of the sample parameters can be calculated as a measure of the algorithm's effectiveness [21]. Low sampling efficiency indicates a high autocorrelation, and the long simulation is required to obtain reliable estimation and sufficiently small Monte Carlo errors in the estimated posterior [21]. After the assessment of the convergence in the MCMC sampling, the posterior mean and the $100(1-\alpha)\%$ credible interval for parameter $\boldsymbol{\theta}$ are summarized by the samples in the sequence after discarding the early iteration in the Markov Chain simulation.

1.2 Network

DNA methylation, as a primary epigenetic modified form of genomic DNA, has been shown to be associated with many biological processes that regulate gene expression [30]. Understanding the interconnection of DNA methylation is necessary to explore the epigenetic mechanisms and to identify the prognostic genes and their corresponding biological functions[31, 32]. Gaussian graphical models (GGM), which describes conditional independence of variables through the presence or absence of edges in the underlying graph [33] and is accomplished by covariance matrix or precision matrix estimation, have wide applications in the analysis of biological networks. The covariance matrix represents two variables are marginally independent without observing other variables when an off-diagonal element is zero. On the other hand, the precision matrix indicates a subtle relationship between the random variables. If the elements in the precision matrix is zero, then it indicates the two variables are conditionally independent given the observation of other nodes. Intuitively, under the context of genes, the DNAm of the genes does not work independently, and it is necessary to consider other genes when calculating the correlation between variables. In this case, I will focus on the Gaussian graphical model based on the precision matrix estimation to construct graphical structures. In the following, I give a brief introduction on network constructions based on precision matrix.

1.2.1 Gaussian Graphical Model

Suppose $\boldsymbol{X}$ is an $(n \times p)$ matrix of n samples and p observations. The k-th row vector of $\boldsymbol{X}$, $\boldsymbol{x_k}$, for $k = 1, \cdots, n$ follows a multivariate normal distribution $N(\mathbf{0}, \boldsymbol{\Sigma})$, where $\boldsymbol{\Sigma}$ is a $p \times p$ positive definite covariance matrix and unknown, and $\boldsymbol{\Omega} = \boldsymbol{\Sigma}^{-1}$ is the inverse of covariance matrix, where i-th row and j-th column is denoted by ω_{ij}. In the multivariate normal distribution, the i-th diagonal element is the inverse of the residual variance, and the ij-th off-diagonal element in $\boldsymbol{\Omega}$ is the negative of the partial covariance between features i and j, $\omega_{ij} = 0$ implies that x_{ki} and x_{kj} are conditionally independent given the

remaining features. Therefore the Gaussian graphical model is an undirected graphical structure among $(x_{k1}, \cdots, x_{kp})$. The graph is denoted by $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$, where $\boldsymbol{V} = 1, \cdots, p$ is the node set, which represents the total number of columns in $\boldsymbol{X} = \{\boldsymbol{X}_1, \cdots, \boldsymbol{X}_p\}$ and each dimension is denoted by $\boldsymbol{X_i}$, and $\boldsymbol{E} \subset \boldsymbol{V} \times \boldsymbol{V}$ is the edge set. The edge set for $\boldsymbol{G}$ contains nodes $(\boldsymbol{X_i}, \boldsymbol{X_j})$ that share a conditional relationship if $\boldsymbol{X_i} \not\perp\!\!\!\perp \boldsymbol{X_j} | \boldsymbol{X}_{v-i,-j}$ and conditionally independent nodes $\boldsymbol{X_i} \perp\!\!\!\perp \boldsymbol{X_j} | \boldsymbol{X}_{v-i,-j}$ are not included in $\boldsymbol{E}$ and correspond to $\omega_{ij} = 0$ [34].

1.2.2 Network constructions

Suppose a random vector $\boldsymbol{x_k} \sim N(\boldsymbol{0}, \boldsymbol{\Sigma})$, for $k = 1, \ldots, n$ with probability density

$$
\begin{aligned}
f(\boldsymbol{x_k}) &= \frac{1}{(2\pi)^{p/2} \det(\boldsymbol{\Sigma})^{1/2}} \exp\{-\frac{1}{2} \boldsymbol{x_k}^T \Sigma^{-1} \boldsymbol{x_k}\} \\
&\propto \det(\boldsymbol{\Omega})^{1/2} \exp\{-\frac{1}{2} \boldsymbol{x_k}^T \boldsymbol{\Omega} \boldsymbol{x_k}\},
\end{aligned}
$$

where $\boldsymbol{\Sigma} = E(\boldsymbol{x_k} \boldsymbol{x_k}^T)$ is the covariance matrix, and $\boldsymbol{\Omega} = \boldsymbol{\Sigma}^{-1}$ is the precision matrix. Then the log-likelihood is

$$
\begin{aligned}
l(\Omega) &= \frac{1}{2} \sum_{i=1}^{n} \log f(\boldsymbol{x_k}) = \frac{1}{2} \text{logdet}(\boldsymbol{\Omega}) - \frac{1}{2n} \sum_{i=1}^{n} \boldsymbol{x_k}^T \boldsymbol{\Omega} \boldsymbol{x_k} \\
&= \frac{1}{2} \text{logdet}(\boldsymbol{\Omega}) - \frac{1}{2} \left\langle \boldsymbol{S}, \boldsymbol{\Omega} \right\rangle,
\end{aligned}
$$

where $\boldsymbol{S} = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{x_k} \boldsymbol{x_k}^T$ is the sample covariance; $\left\langle \boldsymbol{S}, \boldsymbol{\Omega} \right\rangle = tr(\boldsymbol{S\Omega})$. In ideal situation, when the ratio of variables p to observations n is sufficient small $(p/n \ll 1)$, the maximum likelihood (ML) can provide an accurate estimate for Ω. However, when there are more variables (p) than observations (n) in the biological network, the estimation of precision matrix by using the maximum likelihood method becomes unstable and requires shrinkage estimation [35].

To address the large n small p issue, in the penalized likelihood framework, graphical lasso [36] and graphical SCAD [37] methods estimate high-dimensional inverse-covariance matrix under an arbitrary sparsity pattern. In particular, the precision matrix Ω is estimated by penalizing the likelihood using the L1-penalty [36, 38, 39].

Another approach is to estimate Ω in a column-by-column fashion. Yuan [40] and Cai et al.[41] propose the graphical Dantzig selector and CLIME respectively, which can be solved by linear programming. Several Bayesian methods were also developed. Pati et al.[42] investigated sparse Bayesian factor models for dimensionality reduction in high dimensional situations using a point mass mixture prior on the factor loadings, assuming that the true covariance matrix is represented by such a factor model; Wang et al.[43] proposed a Gibbs sampling algorithm for Bayesian analysis of Gaussian graphical lasso models. In the dissertation, I first to select three most recently approaches based on precision matrix to infer the graphical structure, and then compare the performance of these three methods. The first method is proposed by Williams et al. [34]. It is a Bayesian method and estimates sparse matrices with projection predictive selection. This method identifies the edge set from a decision theoretic perspective based on predictive utility, rather than posterior model probabilities or credible interval width. Li et al.'s method [44] is the second approach that I investigate, which developed an estimator of the inverse-covariance matrix for high-dimensional multivariate normal data using the horseshoe prior. The graphical horseshoe provides estimates with small information divergence from the sampling model when the true inverse-covariance matrix is sparse [44] and the posterior mean can be almost unbiased under certain conditions. The third approach is a modification of Wang's algorithm [43], which is proposed by Oya et al [45]. The method assures the positive definiteness of a precision matrix in the Gaussian graphical model in each cycle of the Gibbs sampler by using Hit and Run sampler algorithm.

I utilize simulations to comprehensively compare these three approaches by evaluating their performance under different dimensional settings and graphical structures.

The reminder of the dissertation is organized as follows. Section 2 introduces the nested Bayesian clustering method based on Beta regression. The model assumptions, setting of priors, conditional posterior distributions, and detailed procedure are described

in this section. We demonstrate and evaluate the applicability of the methods through simulations, and discusses the real data application in this section. Section 3 introduces the details of three existing Bayesian network approaches based on the Gaussian graphical model. The three methods are compared in different dimensional setting based on various graphical structures through simulation study, and then are applied to the real data to identify the graphical structures. Section 4 is the summary and discussion of the dissertation.

Chapter 2

Model-based clustering of epigenetic inheritance patterns via Beta Regressions

2.1 Model assumption

Suppose there are I triads (each triad consists of one child and the child's two parents) and J CpG sites which are common to all triads. We assume that all CpG sites are independent of each other. Let $Z1_{ij}$ and $Z2_{ij}$ denote DNA methylation at CpG site j for the ith child's mother and father, respectively, and y_{ij} be the DNA methylation level at site j of the child. The range of DNA methylation is between 0 and 1 and the distribution of DNA methylation at a CpG site can be reasonably fit by a Beta distribution [14],

$$Z1_{ij} \sim Beta(\alpha_j^M,\ \beta_j^M),\ Z2_{ij} \sim Beta(\alpha_j^F, \beta_j^F),$$

where $0 < Z1_{ij}, Z2_{ij} < 1$, α_j^M, β_j^M, α_j^F, and β_j^F are unknown scale parameters, $i = 1, \cdots, I, j = 1, \cdots, J$. Similarly,

$$y_{ij} \sim Beta(\alpha_j^0,\ \beta_j^0), \tag{2.1}$$

where $0 < y_{ij} < 1$, α_j^0 and β_j^0 are two unknown scale parameters in the Beta distribution.

2.2 Identify the transmission status

DNA methylation of offspring at certain CpG sites can be potentially inherited from parents. We examine this at the population level. That is, if at CpG site j, DNA methylation of a child is expected to be transmitted from their parents, then we have the following Beta regression applied to mean DNA methylation,

$$O_j = \gamma_{0j} + \gamma_{1j} M_j + \gamma_{2j} F_j, \tag{2.2}$$

where $O_j = \text{logit}(\frac{\alpha_j^0}{\alpha_j^0+\beta_j^0}) = \log(\alpha_j^0) - \log(\beta_j^0)$, $M_j = \log(\alpha_j^M) - \log(\beta_j^M)$ and $F_j = \log(\alpha_j^F) - \log(\beta_j^F)$ denote logit transformed mean DNA methylation at CpG site j for child, father and mother, respectively. Under this context, the distribution of a child's

DNA methylation depends on their parents' DNA methylation and (2.1) should be re-written as

$$y_{ij}|Z1_{ij}, Z2_{ij} \sim Beta(\alpha_j^0, \beta_j^0).$$

However, not all CpG sites transmit DNA methylation from parents to offspring. To incorporate this concept to the regression model (2.2), we introduce an indicator variable into the model to denote transmission status,

$$O_j = (\delta_j \gamma_{0j} + (1-\delta_j)\gamma_j) + \gamma_{1j}\delta_j M_j + \gamma_{2j}\delta_j F_j, \tag{2.3}$$

where $\boldsymbol{\delta} = (\delta_1, \delta_2, \cdots, \delta_J)^T$ is a $J \times 1$ vector denoting transmission status of each CpG site $j = 1, \cdots, J$. If $\delta_j = 0$, CpG site j is not transmitted from parents to their offspring and average DNA methylation of a child is determined by γ_j. Otherwise, DNA methylation at CpG site j is transmitted from the parents, and transmission patterns are determined by $\boldsymbol{\gamma}_j^{tr} = (\gamma_{0j}, \gamma_{1j}, \gamma_{2j})$, where γ_{1j} represents the strength of maternal transmission and γ_{2j} the strength of paternal transmission. Note that the patterns focus on transmission strengths (slopes) rather than non-transmission related portions in DNA methylation (intercepts). A Bayesian approach is applied to infer transmission status δ_j of each CpG site and, for transmitted sites, the patterns of transmission. Frequentist approaches, e.g., via the expectation-maximization algorithm as in Han et al.[12], can also be used to infer the parameters. The advantage of Bayesian approach exists in its ability to easily introduce prior knowledge into the estimation process. We start the Bayesian inference from specifying prior distributions of the parameters, δ_j and γ_j. For δ_j, we assume a Bernoulli distribution, $\delta_j|p_j \sim Bernoulli(1, p_j)$, with $\boldsymbol{p} = (p_1, \cdots, p_j, \cdots, p_J)$ denoting a vector of transmission probabilities for each of the J CpG sites. For γ_j, a non-informative prior is selected, $N(0, a)$, a normal distribution with mean 0 and variance a. The variance is assumed to be known and large, e.g, $a = 100$. For regression coefficients of transmitted CpGs, $\boldsymbol{\gamma}_j^{tr}$, we discuss in the next section its setting in the context of clustering and related specification of prior distributions.

2.3 Clustering the Transmitted CpG sites

Among the transmitted CpGs, we further group the CpG sites showing similar transmission patterns. Under the context of clustering, equation (2.3) is revised to the following

$$O_j = \gamma_{0k} + \gamma_{1k} M_j + \gamma_{2k} F_j,$$

where the strength of transmission in cluster k is determined by $\boldsymbol{\gamma}_{\boldsymbol{k}}^{tr} = (\gamma_{0k}, \gamma_{1k}, \gamma_{2k})$, representing the clustering of $\boldsymbol{\gamma}_j^{tr}$ across J CpG sites. All CpG sites in cluster k share the same transmission pattern from the parents to their offspring. The parameters γ_{1k} and γ_{2k} represent the strength of maternal transmission and paternal transmission for CpGs in cluster k, respectively. For these regression coefficients, the same non-informative prior distribution as that for γ_j is selected, i.e., $N(0,a)$ with a large and known.

The cluster assignment of CpG j is determined by $\boldsymbol{\mu}_j = (\mu_{j1}, \mu_{j2}, \cdots, \mu_{jK})^T = (0,0,\cdots,1,0,\cdots,0)^T$, a $K \times 1$ vector and $\mu_{jk} = 1$ indicates CpG site j belonging to cluster k. Denote by π_k the probability of a site falling into cluster k. We assume the prior distribution of μ_{jk} as $Mult(1,\pi)$ (Multinomial distribution), where $\boldsymbol{\pi} = (\pi_1, \pi_2, \cdots, \pi_K)$ with $0 \le \pi_k \le 1$, $k = 1,2,\cdots,K$, $\sum_{k=1}^{K} \pi_k = 1$. For the distribution of its hyper-prior, π, we assume $\pi \sim Dir(\mathbf{1})$ with $\mathbf{1}$ being a vector of 1 with length K, a Dirichlet distribution with parameter 1, which is equivalent to a multivariate uniform distribution.

2.4 The likelihood function and the posterior distribution

Let $\theta = (\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\delta}, \boldsymbol{\mu}, \gamma_j, \boldsymbol{\gamma}^{tr}, j = 1,\cdots,J)$ denote a collection of all parameters, where $\boldsymbol{\alpha} = (\boldsymbol{\alpha}^0, \boldsymbol{\alpha}^M, \boldsymbol{\alpha}^F)$ with $\boldsymbol{\alpha}^0 = (\alpha_1^0, \alpha_2^0, \cdots, \alpha_J^0)$, $\boldsymbol{\alpha}^M = (\alpha_1^M, \alpha_2^M, \cdots, \alpha_J^M)$, $\boldsymbol{\alpha}^F = (\alpha_1^F, \alpha_2^F, \cdots, \alpha_J^F)$. Analogous to $\boldsymbol{\alpha}$, parameter $\boldsymbol{\beta}$ has the same structure, $\boldsymbol{\beta} = (\boldsymbol{\beta}^0, \boldsymbol{\beta}^M, \boldsymbol{\beta}^F)$ with $\boldsymbol{\beta}^0 = (\beta_1^0, \beta_2^0, \cdots, \beta_J^0)$, $\boldsymbol{\beta}^M = (\beta_1^M, \beta_2^M, \cdots, \beta_J^M)$ and $\boldsymbol{\beta}^F = (\beta_1^F, \beta_2^F, \cdots, \beta_J^F)$. For the transmission status, $\boldsymbol{\delta} = (\delta_1, \delta_2, \cdots, \delta_J)^T$ is a $J \times 1$ vector, and $\boldsymbol{\mu} = (\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \cdots, \boldsymbol{\mu}_J)^T$ with $\boldsymbol{\mu}_j = (\mu_{j1}, \mu_{j2}, \cdots, \mu_{jK})^T = (0,0,\cdots,1,0,\cdots,0)^T$ is a $K \times 1$ vector denoting cluster

assignment. Finally, for non-transmitted CpG site j, parameter γ_j is the average DNA methylation of a child. Among the transmitted CpG sites, parameter $\boldsymbol{\gamma}^{tr}$ is a collection of regression coefficients, $\boldsymbol{\gamma}^{tr} = (\boldsymbol{\gamma}_1^{tr}, \boldsymbol{\gamma}_2^{tr}, \cdots, \boldsymbol{\gamma}_K^{tr})$ with $\boldsymbol{\gamma}_k^{tr} = (\gamma_{0k}, \gamma_{1k}, \gamma_{2k})$, $k = 1, 2, \cdots, K$. Denote by $Y = (y_{ij}, Z1_{ij}, Z2_{ij})$, $i = 1, 2, \cdots, I$, $j = 1, 2, \cdots, J$. The likelihood of θ is ,

$$L(\theta|Y) \propto \prod_{i=1}^{I}\prod_{j=1}^{J}(\prod_{k=1}^{K} \mathrm{I}(\delta_j = 1)p(y_{ij}|Z1_{ij}, Z2_{ij}, \theta)^{\mu_{jk}} \tag{2.4}$$

$$p(Z1_{ij}|\theta)p(Z2_{ij}|\theta) + (1 - \mathrm{I}(\delta_j = 1))p(y_{ij}|\theta)).$$

The joint posterior distribution of $\boldsymbol{\theta}$ is then given as

$$\begin{aligned} p(\theta|Y) \quad \propto \quad & p(\boldsymbol{\theta})\prod_{i=1}^{I}\prod_{j=1}^{J}(\prod_{k=1}^{K} \mathrm{I}(\delta_j = 1)p(y_{ij}|Z1_{ij}, Z2_{ij}, \theta)^{\mu_{jk}} \\ & p(Z1_{ij}|\theta)p(Z2_{ij}|\theta) + (1 - \mathrm{I}(\delta_j = 1))p(y_{ij}|\theta)), \end{aligned}$$

with the prior of $\boldsymbol{\theta}$ as defined in earlier sections.

Markov Chain Monte Carlo (MCMC) simulations from the joint posterior distribution are implemented to draw posterior inferences. Specifically, we utilize the Gibbs sampler with Metropolis-Hasting steps to sample from the full conditional posterior distributions of each parameter, which are then used to infer the parameters of interest. In the following, we list the conditional posterior distributions with $(\cdot)$ denoting data and other parameters to be conditional on.

For parameter δ_j, representing transmission status of CpG sites, we have the following conditional posterior distribution,

$$\delta_j|(\cdot) \sim Bernoulli(p_{post}). \tag{2.5}$$

Since

$$p(\delta_j = 1|(\cdot)) \propto p(\boldsymbol{\theta})\prod_{i=1}^{I}\prod_{k=1}^{K} p(y_{ij}|Z1_{ij}, Z2_{ij}, \theta)^{\mu_{jk}} p(Z1_{ij}|\theta)p(Z2_{ij}|\theta) = a,$$

and

$$p(\delta_j = 0|(\cdot)) \propto p(\boldsymbol{\theta})\prod_{i=1}^{I} p(y_{ij}|\theta) = b,$$

parameter p_{post} in (2.5) is defined as

$$p_{post} = \frac{a}{a+b}.$$

Among the transmitted CpG sites, the conditional posterior of μ_j is a multinomial distribution with π_k in $\boldsymbol{\pi} = (\pi_1, \pi_2, \cdots, \pi_K)$ determined by,

$$\begin{aligned}\pi_k|(\cdot) = p(\mu_{jk}|(\cdot)) \propto \prod_{i=1}^{I} & p(y_{ij}|Z1_{ij}, Z2_{ij}, \boldsymbol{\alpha}_j, \boldsymbol{\beta}_j, \boldsymbol{\gamma}_k^{tr}, \delta_j, \mu_{jk}) \\ & p(Z1_{ij}|\alpha_j^M, \beta_j^M) p(Z2_{ij}|\alpha_j^F, \beta_j^F) p(\boldsymbol{\gamma}_k^{tr}|\delta_j) \\ & p(\mu_{jk}|\boldsymbol{\pi}) p(\pi_k).\end{aligned}$$

The conditional posterior distribution of $\boldsymbol{\gamma}_j$ for the non-transmitted CpG sites is,

$$P(\gamma_j|(\cdot)) \propto \prod_{i=1}^{I} p(y_{ij}|\alpha_j^0, \beta_j^0, \gamma_j) p(\gamma_j|\delta_j),$$

and the conditional posterior distribution of the transmission patterns $\boldsymbol{\gamma}_k^{tr} = (\gamma_{mk}, m = 0, 1, 2)$ for the transmitted CpG sites is defined as

$$\begin{aligned}P(\gamma_{mk}|(\cdot)) \propto \prod_{i=1}^{I}\prod_{j=1}^{J} & p(y_{ij}|Z1_{ij}, Z2_{ij}, \boldsymbol{\alpha}_j, \boldsymbol{\beta}_j, \boldsymbol{\gamma}_k^{tr}, \delta_j, \mu_{jk}) \\ & p(Z1_{ij}|\alpha_j^M, \beta_j^M) p(Z2_{ij}|\alpha_j^F, \beta_j^F) p(\gamma_{mk}).\end{aligned}$$

The conditional posterior distributions of γ_j and γ_k^{tr} are not in the standard form and to sample γ_j and γ_k^{tr}, we apply the Metropolis-Hasting algorithm and take the normal distribution as the proposal distribution. With all the conditional posterior distributions defined, for a given K, we apply the Gibbs sampler to sequentially sample from each distributions to determine each CpG site's transmission status as well as cluster assignment among transmitted CpGs. Since clustering of CpG sites happens after we determine the transmission status of each CpG site, we denote this as a nested clustering method.

To determine the number of clusters K, we implement the Bayesian information criterion (BIC) introduced in Han et al.[12] constructed via the likelihood calculated using posterior estimates of the parameters. A scree plot of BICs for different values of K and a sharp decrease in BIC followed by a flatten pattern indicates an optimal selection of K.

2.5 Simulation Study

This section, via simulation, demonstrates and evaluates the proposed method by using 100 Monte Carlo (MC) replicates, and assesses its sensitivity and specificity to cluster detections.

2.5.1 Simulation scenarios

We generate 100 Monte Carlo (MC) replicates. Each MC replicate represents DNA methylation of 650 CpG sites from 100 triads. DNA methylation data is simulated using Beta distributions. The scale parameters in the Beta distribution are generated from truncated normal distributions for each CpG site, which potentially results in unique patterns of data at each CpG sites. Out of 650 CpG sites, about 500 transmitted CpG sites are equally assigned into 2 clusters with coefficients $\boldsymbol{\gamma}_1^{tr} = (-4.2, 1.2, 2.5)$, $\boldsymbol{\gamma}_2^{tr} = (-2.7, 3, 2.5)$, representing CpG sites paternal and maternal dominated transmission, respectively. To assess the sensitivity and specificity of the method with respect to different transmission patterns, for another set of 100 MC replicates, we assign the 500 transmitted CpG sites into 3 clusters. In this scenario, the first two clusters have the same setting as before. For the third cluster, the coefficients are chosen as $\boldsymbol{\gamma}_3^{tr} = (-8, 1, 1)$, that is, parental transmission is evenly distributed between mother and father. These two scenarios are constructed based on different transmission patterns.

To summarize our results, for each MC replicate, we calculate the rate of accuracy of detected transmission status, record the number of clusters identified, and calculate the sensitivity $(SE = TP/(TP+FN))$ and specificity $(SP = TN/(TN+FP))$ of clustering, where "TP" denotes true positives (correct cluster identification), "FN" denotes false negatives, "TN" denotes true negatives, and "FP" denotes false positives. Medians of these statistics along with IQR (25^{th} percentile - 75^{th} percentile) across the 100 MC replicates are summarized and used to assess the robustness of the method as well as uncertainty of the inferred statistics. As noted earlier, the number of clusters is determined by use of the scree plot of BICs with each BIC corresponding to a specific number of

clusters. As an illustration, Figure 2.1 shows the pattern of BIC from one MC replicate, from which we infer 2 clusters.

2.5.2 Results

We discuss findings from the fully Bayesian sampling scheme discussed in Section 2.1-2.4. In total, we run one chain with 20,000 MCMC iterations and use the last 1000 iterations to draw inferences on the parameters. In the process of inferring transmission patterns, i.e., estimating $\boldsymbol{\gamma}_k^{tr}$ for cluster k, rather than drawing posterior samples for each component in $\boldsymbol{\gamma}_k^{tr}$, we utilize linear regressions based on CpGs assigned to cluster k to improve sampling efficiency.

Summarizing the result across the 100 MC replicates, the uncertainty on the number of clusters is shown in Table 2.1 as the frequency for each cluster number over the 100 MC replicates in the two- and three-cluster situations. In both situations, the numbers of clusters inferred are correct; the median of number of clusters is 2 with IQR of $(2,4)$ when the underlying true number of clusters is 2, and for the three-cluster situation, these two statistics are 3 and $(3,4)$. The rate of accuracy of transmission status, and sensitivity and specificity of clustering are summarized in Table 2.2. Overall, regardless of the underlying truth on the number of clusters, high accuracy, sensitivity, and specificity are observed, indicating the effectiveness of the proposed method, and its potential to handle a combination of different transmission status.

Table 2.1: The occurrence frequency of each cluster number over 100 MC replicates

		Underlying truth:	2 clusters	
Number of candidate clusters (K)	2	3	4	5
Frequency	59	9	19	12
		Underlying truth:	3 clusters	
Number of candidate clusters (K)	2	3	4	5
Frequency	17	56	15	12

In the above analysis, we focus on the assessment of the method on its ability to deal with different transmission patterns. To further assess the proposed method, we compare

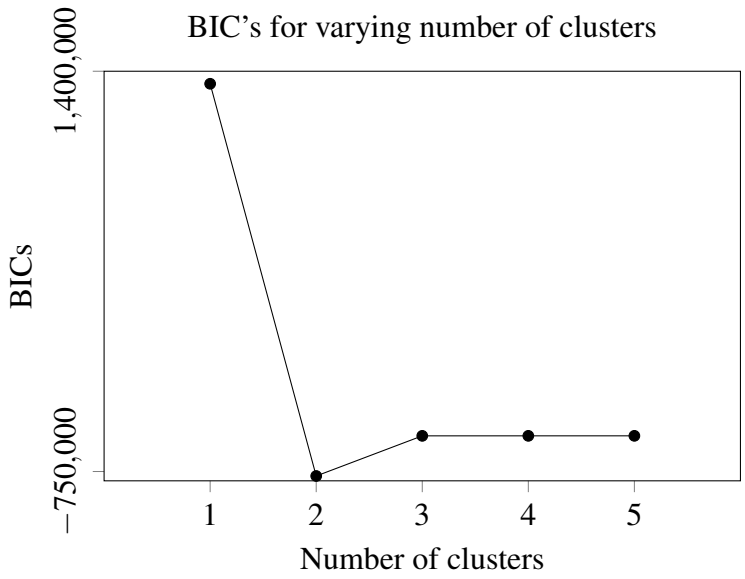


Figure 2.1: BIC values for a varying number of clusters in the simulated data with 650 CpG sites in one MC replicate. The best number of clusters is achieved at K=2.

with the approach by Han et al.[12], from which the proposed approach is extended. Furthermore, to demonstrate properties reflected by finite samples, we examine how the results of clustering and detection of transmission are affected by different sample sizes and different numbers of CpG sites. Both two-cluster and three-cluster situations are considered. To compare with the method in Han et al [12], we use data simulated following the scenario in Section 2.5.1 for two-cluster situation, and we assume DNA methylation at all CpG sites was transmitted and set the number of clusters as three (denote this situation as S0). To evaluate the performance of our method with different sample sizes and numbers of CpG sites compared to the setting in Section 2.5.1, we considered the following five situations. The first three situations are for two clusters situation and the rests are for three clusters situation. For each situation, 100 MC replicates are simulated,

S1. decreasing the sample size from 100 to 85.

S2. increasing the sample size from 100 to 200. S1 and S2, along with the setting used

Table 2.2: Summary of rate of accuracy transmission status, sensitivity and specificity of clustering across 100 MC replicates for both two and three clusters situations. IQR stands for 25^{th} percentile - 75^{th} percentile. Each MC replicate presents DNA methylation of 650 CpG sites from 100 triads generated. Out of 650 CpG sites, about 500 CpG sites are equally assigned into two or three clusters, respectively.

			Two Clusters	Three Clusters
Transmission status	Accuracy	Median	0.9985	1.000
		IQR	(0.9954,1)	(0.9938,1)
Cluster 1	Sensitivity	Median	0.9960	1.000
		IQR	(0.9921,1)	(0.9939,1)
	Specificity	Median	0.9796	0.9881
		IQR	(0.9684,0.9842)	(0.9762,0.9939)
Cluster 2	Sensitivity	Median	0.9796	0.9750
		IQR	(0.9684,0.9842)	(0.9426,0.9850)
	Specificity	Median	0.9960	1.000
		IQR	(0.9921,1)	(0.9970,1)
Cluster 3	Sensitivity	Median	-	1.000
		IQR	-	(1,1)
	Specificity	Median	-	1.000
		IQR	-	(1,1)

in Section 2.5.1 for two clusters situation, are to assess the impact of sample sizes on the inference of clustering and transmission status.

S3. increasing the number of CpG sites to 1500 and equally assigning the 1000 transmitted CpGs to two clusters (500 CpG sites in each cluster). This is to assess the influence of larger number of CpG sites.

S4. increasing the sample size from 100 to 200 for three clusters situation.

S5. increasing the number of CpG sites to 1500 and equally assigning the 1000 transmitted CpG sites to three clusters.

The results of S0 to S3 are summarized in Table 2.3 and the results of S4 to S5 are summarized in Table 2.4. Under S0 assuming DNA methylation at all CpG sites is transmitted, the approach in Han et al. groups the CpG sites into two clusters in most of

the MC replicates. In particular, some of the nontransmitted CpGs are grouped with transmitted CpGs, leading to low sensitivity or specificity. Although this might be due to large variations in the data, the results emphasize the need to take into account DNA methylation transmission status. Turning to the impact of sample sizes and numbers of CpG sites, as the sample size increases (from 85 [S1] to 100 [S0], and then to 200 [S2, S4]), overall both sensitivity and specificity increase. Similar patterns are observed when increasing the number of CpG sites([S3], [S5]), which is understandable, since a larger number of CpG sites is equivalent to having a larger sample size in terms of parameter estimation. For three clusters, although the IQR of transmission status and sensitivity or specificity in particular clusters are larger with increased sample sizes or number of clusters, most of the replicates can correctly identify transmitted CpG sites and group into the clusters. All these findings support the large sample properties on parameter inferences.

Table 2.3: Results (accuracy, sensitivity, and specificity) from Han et al.'s approach (S0) and from two-cluster settings with sample size and number of CpG sites different from those in section 2.5.1 (S1 to S3), summarized across 100 MC replicates.

			S0	S1	S2	S3
Transmission	Accuracy	Median		0.9985	1.000	0.9987
		IQR		(0.9969, 1)	(0.9996,1)	(0.9950,1)
Cluster 1	Sensitivity	Median	0.7333	0.9960	1.000	0.9961
		IQR	(0.1912, 0.7647)	(0.9920,1)	(0.9960,1)	(0.9942,0.9980)
	Specificity	Median	0.4723	0.9688	0.9959	0.9781
		IQR	(0.1660, 0.5799)	(0.9540,0.9763)	(0.9919,0.9961)	(0.9724,0.9838)
Cluster 2	Sensitivity	Median	0.2285	0.9688	0.9959	0.9781
		IQR	(0.1904, 0.2744)	(0.9540,0.9763)	(0.9919,0.9961)	(0.9724,0.9838)
	Specificity	Median	0.6879	0.9960	1.000	0.9961
		IQR	(0.4491, 0.7319)	(0.9920,1)	(0.9960,1)	(0.9942,0.9980)

2.6 Real data analysis for clustering epigenetic sites on inheritance

We applied the proposed nested clustering method to DNA methylation data measured in two generations for a random subset of participants in a birth cohort located on the Isle of Wight, United Kingdom [46]. The methylation level for each queried CpG is presented

Table 2.4: Results (accuracy, sensitivity, and specificity) from from three-cluster settings with sample size and number of CpG sites different from those in section 2.5.1 (S4, S5), summarized across 100 MC replicates.

			S4	S5
Transmission	Accuracy	Median	1.000	1.000
		IQR	(0.5227, 1)	(0.4567,1)
Cluster 1	Sensitivity	Median	1.000	1.000
		IQR	(0.2342,1)	(0, 1)
	Specificity	Median	1.000	1.000
		IQR	(1, 1)	(1, 1)
Cluster 2	Sensitivity	Median	1.000	1.000
		IQR	(1, 1)	(1, 1)
	Specificity	Median	1.000	1.000
		IQR	(1, 1)	(1, 1)
Cluster 3	Sensitivity	Median	1.000	1.000
		IQR	(0.9879, 1)	(1, 1)
	Specificity	Median	1.000	1.000
		IQR	(0.8562, 1)	(0.3565, 1)

as beta values. The beta values represent the ratio of the methylated (M) probe intensities to the sum of methylated (M) and unmethylated (U) probe intensities (beta=M/(U+M+C) with constant C=100 introduced for the situation of too small M+U). In total, DNA methylation at more than $450K$ CpG sites are available.

In the current study, DNA methylation at 4063 CpGs on autosomes in 41 triads are included and selected based on potential inheritance measured by correlations in DNA methylation between a parent and his/her offspring. A CpG site will be excluded from further consideration if the mother-child or father-child correlation in DNA methylation is < 0.5. Although DNA methylation at these 4063 candidate CpG sites show a potential of inheritance, uncertainty of inheritance and effects of both maternal and paternal CpGs on their offspring DNA methylation (instead of individual effects) are not considered by simple correlations. We use these 4063 CpGs as candidate sites to infer status of transmission as well as clustering transmittted CpG sites. The nested clustering method discussed in Section 2.1-2.4 is applied to identify transmitted CpG sites at the population

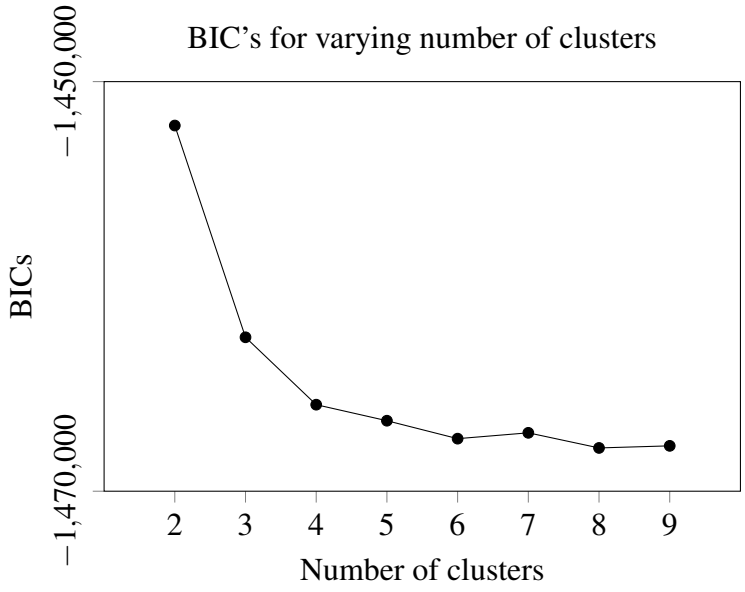


Figure 2.2: BIC values for a varying number of clusters for the IOW real data with 4063 CpG sites and 41 triads after screening by a cut off at 0.5 in correlation. The BIC curve decreases sharply, then reaches stable at K=5.

Table 2.5: Coefficient estimation of 3837 CpGs identified as transmitted sites. CI denotes the 95% credible interval.

Cluster index(k)	$\hat{\gamma_{0k}}$(CI)	$\hat{\gamma_{1k}}$(CI) maternal transmission	$\hat{\gamma_{2k}}$(CI) paternal transmission	No.CpG sites
1	0.24	0.52	0.52	422
	(0.23, 0.25)	(0.47, 0.56)	(0.47, 0.57)	
2	0.08	0.60	0.43	643
	(0.07, 0.08)	(0.55, 0.63)	(0.39, 0.47)	
3	0.55	0.53	0.49	177
	(0.53, 0.57)	(0.42, 0.65)	(0.37, 0.61)	
4	-0.06	0.64	0.39	1054
	(-0.07, -0.06)	(0.60, 0.69)	(0.34, 0.43)	
5	-0.32	0.57	0.50	1541
	(-0.33, -0.31)	(0.55, 0.60)	(0.48, 0.53)	

level and assign the transmitted CpG sites to different clusters. The scree plot of BIC, which is used to estimate the number of clusters, is displayed in Figure 2.2. The plot indicates that the best number of cluster is achieved at K=5. The estimated coefficients

with their 95% credible intervals, which are used to explain the transmission strength, and the number of transmitted CpG sites included in each cluster are summarized in Table 2.5.

Of the 4063 candidate CpGs, 3837 CpG sites are identified as transmitted sites. Recall that the parameter γ_{1k} represents the strength of maternal transmission and the γ_{2k} is for the strength of paternal transmission. Among the 5 identified clusters, the first cluster containing 422 CpG sites shows an even transmission strength between mothers and fathers, indicating parents have the same contribution to offspring's DNA methylation. Cluster 3 also shows a similar transmission strength between the parents, indicated by the overlapping credible intervals. The remaining clusters are mainly maternal-transmission dominated, as indicated by larger estimates of γ_{1k} as well as non-overlapping credible intervals.

With 3837 of the 4063 CpGs identified as potentially transmitted CpGs, DNA methylation at some CpGs is likely not transmitted based on our approach although the correlations is 0.5 or larger. This is likely due to large uncertainties in the data as well as the consideration of effects from both parents in our transmission assessment.

Chapter 3

Simulation studies to compare methods for Gaussian network constructions

In the Introduction, I briefly introd the Gaussian graphical model and the methods of network constructions based on the precision matrix. In the following, I will provide details of the three methods that will be compared.

3.1 Bayesian Predictive Covariance Selection

The projection method, which is developed by Williams et al. [34], is a two step procedure that provides a theoretically decision inference after selection [34]. That is, instead of constructing the conventional posterior given constraints, the full posterior is projected to the restricted subspace [34]. In the projection method, the first step is to construct the reference model $\boldsymbol{M}_*$. The random variables can be partitioned into two groups $\boldsymbol{X} = (\boldsymbol{X_i}, \boldsymbol{X}_{-i})$, where $\boldsymbol{X}_{-i} = (\boldsymbol{X_1}, \cdots, \boldsymbol{X}_{i-1}, \boldsymbol{X}_{i+1}, \cdots, \boldsymbol{X_p})$. Assuming $\boldsymbol{X}$ follows a multivariate normal distribution, the conditional distribution of $\boldsymbol{X_i}$ given $\boldsymbol{X}_{-i}$ is also normally distributed. The reference model $\boldsymbol{M}_*$ is defined as

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \varepsilon, \varepsilon \sim N(0, \sigma^2), \tag{3.1}$$

where $\boldsymbol{X}$ is a $n \times (p-1)$ matrix that includes a subset of nodes $\boldsymbol{X}_{-i}$ from $\boldsymbol{V}\{1, \cdots, p\}$, $\boldsymbol{\beta}$ is a $p-1$ dimensional vector including $\beta_j^{(i)}, j \in \{1, \cdots, p-1\}$, and $\boldsymbol{y}$ is the *i-th* node to be predicted.

To infer $\boldsymbol{\beta}$, the horseshoe prior distribution is used,

$$\begin{aligned} \beta_j | \lambda_j, \tau &\sim N(0, \lambda_j^2 \tau^2), j \in 1, \cdots, p-1, \\ \lambda_j &\sim C^+(0, 1), \\ \tau &\sim C^+(0, \tau_0), \end{aligned} \tag{3.2}$$

where C^+ is a half-Cauchy distribution for the local and global hyperparameters, which are denoted by λ and τ, respectively. The global hyperparameter τ provides shrinkage to

all estimate and the local hyperparameter λ allows for stronger signals to avoid shrinking and can be used to detect non-zero relationships. The scale for the prior distribution of τ is assumed as a generic value ($\tau_0 = 1$).

The second step is to determine which nodes can be set to zero ($\hat{\beta}_j^{(i)} = 0$) for each of the p regression models, so that the reduced model's predictive distribution is close to $\boldsymbol{M}_*$ [47]. For the purpose of mapping a parameter $\boldsymbol{\Theta} \mapsto \boldsymbol{\Theta}_\perp$, the projected posterior draws $\{\boldsymbol{\theta}_\perp^{\boldsymbol{s}}\}_{s=1}^{\boldsymbol{S}}$, which denote as the parameters of the projected submodel $\boldsymbol{M}_\perp$, were derived by minimizing the discrepancy between the predictive distributions conditional their corresponding parameters,

$$\begin{aligned} \theta_\perp &= \underset{\theta' \in \Theta_\perp}{\operatorname{argmin}}\, \delta(\theta, \theta') \\ &\triangleq \underset{\theta' \in \Theta_\perp}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{n} KL(p(\tilde{y}|\theta, X_i, M_*)|p(\tilde{y}|\theta', x_i, M_\perp)), \end{aligned} \tag{3.3}$$

where the discrepancy δ is defined as Kullback-Leibler (KL) divergence averaged over the empirical distribution for $\boldsymbol{X}$. Then, by using forward search strategy, the reduced models with minimal projection loss are selected, which is defined as the mean discrepancy over the posterior of the reference model in (3.4).

$$L = \frac{1}{S} \sum_{s=1}^{S} \delta(\theta_s, \theta_\perp^s), \tag{3.4}$$

The posterior draws are obtained based on the Pareto smoothed importance sampling (PSIS), which a method for stabilizing importance sampling estimates, including stabilized effective sample size estimates, Monte Carlo error estimates and convergence diagnostics [48]. To assess the accuracy of the reduced models, the leave-one-out(LOO) cross-validation is used. In PSIS-LOO, the posterior draws can be obtained from the LOO posteriors after weighting them by the importance weights,

$$\omega_{is} \propto \frac{1}{p(y_i|\theta_s)}$$

which are then smoothed to stabilize the LOO estimates [20]. Thus the projection is carried out with the data point i left out, where the submodel draws $\{\theta_s\}_{s=1}^{S}$ are weighted

with ω_{is} when predicting the left-out point [34]. Then, for a given model complexity k, the predictive accuracy $\hat{u_k}$ along with the standard error s_k are calculated, and the log predictive density (LPD) is used to assess the accuracy. Based on the predictive accuracy $\hat{u_k}$, the edge set is determined by the criteria in (3.5), that is, the selected smallest model has an acceptable difference $\triangle u > 0$ relative to the reference model $\hat{u_*}$ with some level of confidence α, which is estimated using the Bayesian bootstrap.

$$Pr[u_* - u_k \leq \triangle u] \geq 1 - \alpha, \tag{3.5}$$

The choice of $\triangle u$ and α is determined by how much predicted accuracy is prepared to be sacrificed in exchange for model simplicity [34].

After the selection is completed, for each node-wise regression, the projected estimates are placed into a $p \times p$ matrix $\mathbf{\Lambda}_{i,j\neq i}$. In order to achieve symmetric non-zero entries in $\mathbf{\Lambda}$ in high dimensional settings, the 'or-rule' is used, that is, neighborhoods are re-projected to take into account the inconsistent estimates. To calculate the partial correlation matrix, the approach is based on the Kramer et al. [49], that is,

$$\tilde{\boldsymbol{P}}_{i,-i} = [\hat{p}_{ij}], j \in \{1, \cdots, p-1\},$$

$$\hat{\rho}_{ij} = \text{sign}(\Lambda)\min\{1, \sqrt{\Lambda_{ij} \circ \Lambda_{ij}^T}\}, \text{if sign}(\Lambda_{ij}) = \text{sign}(\Lambda_{ji}),$$

$$\text{and 0 otherwise.}$$

where $\circ$ denotes the Hadamard product between the corresponding matrix elements. For the precision matrix estimation ($\mathbf{\Omega}$), in low-dimensional setting, the diagonal and off-diagonal elements are defined as

$$\omega_{ii} = \frac{1}{Var(e_k)_{ii}}, \omega_{ij} = \frac{-\beta_{ij}}{Var(e_k)_{ii}},$$

where $Var(e_i)$ is the residual variance. While in high dimensional setting, the projected residual variance is calculated by including only the selected nodes for each neighborhood,

$$\hat{\Omega}_{ii} = \frac{1}{Var(e_k)_{ii}},$$

$$Var(e_k)_{ii} = y - \Lambda_{i,-i} X_{-i},$$

where $\boldsymbol{y} = \boldsymbol{X}_i$, $\boldsymbol{X}_{-i}$ is the model matrix excluding the *i-th* variable, and $\boldsymbol{\Lambda}_{i,-i}$ is the matrix containing the projected matrix and the off-diagonal elements in $\boldsymbol{\Omega}$ is

$$\hat{\omega}_{ij} = -\frac{1}{2}\left(\frac{\Lambda_{i,i\neq j}}{Var(e_k)_{ii}} + \frac{\Lambda_{j,j\neq i}}{Var(e_k)_{jj}}\right),$$

where symmetry is obtained by averaging the values.

3.2 Graphical Horseshoe Model

The graphical horseshoe model, which is proposed by Li et al.[44], uses horseshoe priors on the off-diagonal elements of the precision matrix and an informative prior on the diagonal elements, while respecting the constraint $\boldsymbol{\Omega} \in S_p$ [44]. The element-wise priors are specified as follows

$$\begin{aligned} \omega_{ii} &\propto 1, \\ \omega_{ij:i<j} &\sim Normal(0, \lambda_{ij}^2\tau^2), \\ \lambda_{ij:i<j} &\sim C^+(0,1), \\ \tau &\sim C^+(0,1), \end{aligned} \tag{3.6}$$

where $C^+(0,1)$ denotes a half-Cauchy random variable with density $p(x) \propto (1+x^2)^{-1}; x > 0$ and the normal scale mixtures with half-Cauchy hyperpriors on the off-diagonal elements is a horseshoe prior. The distinctive scale parameter λ_{ij} and τ are local and global shrinkage parameter, respectively. The prior on $\boldsymbol{\Omega}$ under graphical horseshoe model can be written as

$$p(\boldsymbol{\Omega}|\tau) \propto \prod_{i<j} \text{Normal}(\omega_{ij}|\lambda_{ij}^2.\tau^2) \prod_{i<j} C^+(\lambda_{ij}|0,1)\mathbf{1}_{\boldsymbol{\Omega}\in\boldsymbol{S_p}}, \tag{3.7}$$

where $\boldsymbol{S_p}$ is the space of $p \times p$ positive definite matrices. Therefore, the posterior distribution of the precision matrix $\boldsymbol{\Omega}$ is

$$p(\boldsymbol{\Omega}|\boldsymbol{X}, \boldsymbol{\Lambda}, \tau) \propto |\boldsymbol{\Omega}|^{\frac{n}{2}} \exp\{-\text{tr}(\frac{1}{2}\boldsymbol{S}\boldsymbol{\Omega})\} \prod_{i<j} \exp(-\frac{\omega_{ij}^2}{2\lambda_{ij}^2\tau^2})\mathbf{1}_{\boldsymbol{\Omega}\in\boldsymbol{S_p}}, \tag{3.8}$$

where $\boldsymbol{X}$ is a $n \times p$ matrix and $\boldsymbol{S} = \boldsymbol{X}\boldsymbol{X}'$.

To draw the posterior samples under the graphical horseshoe hierarchical model, an augmented block Gibbs sampler method is used, which is proposed by Makalic and

Schmidt [50]. Two augmented variables $\upsilon_{ij:i<j}$ amd ξ are introduced for conjugate sampling of the shrinkage parameter $\lambda_{ij:i<j}$ and τ. In each iteration, following Wang et al.[43], each column and row of $\boldsymbol{\Omega}$, $\boldsymbol{S}$ and $\boldsymbol{\Lambda}$ are partitioned from a $p \times p$ matrix as

$$\boldsymbol{\Omega} = \begin{pmatrix} \boldsymbol{\Omega}_{(-p)(-p)} & \boldsymbol{\omega}_{(-p)p} \\ \boldsymbol{\omega}^{'}_{(-p)p} & \omega_{pp} \end{pmatrix}, \boldsymbol{S} = \begin{pmatrix} \boldsymbol{S}_{(-p)(-p)} & \boldsymbol{s}_{(-p)p} \\ \boldsymbol{s}^{'}_{(-p)p} & s_{pp} \end{pmatrix}, \boldsymbol{\Lambda} = \begin{pmatrix} \boldsymbol{\Lambda}_{(-p)(-p)} & \boldsymbol{\lambda}_{(-p)p} \\ \boldsymbol{\lambda}^{'}_{(-p)p} & 1 \end{pmatrix},$$

where $(-p)$ denotes the set of all elements except for p, and $\boldsymbol{\Lambda}_{(-p)(-p)}$ and $\boldsymbol{\lambda}_{(-p)p}$ have entries λ^2_{ij}. Then, the full conditional of the last column of $\boldsymbol{\Omega}$ is

$$p(\boldsymbol{\omega}_{(-p)p}, \omega_{pp} | \boldsymbol{\Omega}_{(-p)(-p)}, \boldsymbol{X}, \boldsymbol{\Lambda}, \tau) \propto (\omega_{pp} - \boldsymbol{\omega}^{'}_{(-p)p} \boldsymbol{\Omega}^{-1}_{(-p)(-p)} \boldsymbol{\omega}_{(-p)p})^{\frac{n}{2}} \times \exp\{-\boldsymbol{s}^{'}_{(-p)p} \boldsymbol{\omega}_{(-p)p} - \frac{s_{pp}\omega_{pp}}{2} - \frac{\boldsymbol{\omega}^{'}_{(-p)p} (\boldsymbol{\Lambda}^* \tau^2)^{-1} \boldsymbol{\omega}_{(-p)p}}{2}\},$$

where $\boldsymbol{\Lambda}^*$ is a diagonal matrix with $\boldsymbol{\lambda}_{(-p)p}$ in the diagonal. Let $\boldsymbol{\beta} = \boldsymbol{\omega}_{(-p)p}$ and $\gamma = \omega_{pp} - \boldsymbol{\omega}^{'}_{(-p)p} \boldsymbol{\Omega}^{-1}_{(-p)(-p)} \boldsymbol{\omega}_{(-p)p}$, the full conditional distribution of $\boldsymbol{\beta}$ and γ is

$$p(\boldsymbol{\beta}, \gamma | \boldsymbol{\Omega}_{(-p)(-p)}, \boldsymbol{X}, \boldsymbol{\Lambda}, \tau) \propto \gamma^{\frac{n}{2}} \exp[-\frac{1}{2}\{s_{pp}\gamma + \boldsymbol{\beta}^{'} s_{pp} \boldsymbol{\Omega}^{-1}_{(-p)(-p)} \boldsymbol{\beta} + \boldsymbol{\beta}^{'} (\boldsymbol{\Lambda}^* \tau^2)^{-1} \boldsymbol{\beta} + 2\boldsymbol{s}^{'}_{(-p)p} \boldsymbol{\beta}\}]$$
$$\sim \text{Gamma}(\frac{n}{2} + 1, \frac{s_{pp}}{2}) \text{Normal}(-C\boldsymbol{s}_{(-p)p'} C),$$

where $C = \{s_{pp} \boldsymbol{\Omega}^{-1}_{(-p)(-p)} + (\boldsymbol{\Lambda}^* \tau^2)^{-1}\}^{-1}$. Based on the posterior distribution of the last row and column in $\boldsymbol{\Omega}$, each row and column in the matrix $\boldsymbol{\Omega}$ are updated at a time. Then the local and global shrinkage parameters λ_{ij} and τ are sampled and followed by Makalic and Schmidt [50] as

$$\lambda^2_{ij} | \upsilon_{ij} \sim \text{InvGamma}(1, \frac{1}{\upsilon_{ij}} + \frac{\omega^2_{ij}}{2\tau^2}), \upsilon_{ij} | \cdot \sim \text{InvGamma}(1, 1 + \frac{1}{\lambda^2_{ij}}). \tag{3.9}$$

The process of sampler is summarized in Algorithm 1 [44].

Algorithm 1 The Graphical Horseshoe Sampler

function GHS($S, n, burnin, nmc$) ▷ where $S = X^{'}X$,n=sample size

Set p to be number of rows (or columns) in S

Set initial values $\Omega = I_{p\times p}$, $\Sigma = I_{p\times p}$, $\Lambda = \mathbb{1}$, $N = \mathbb{1}$, $\tau = 1$, $\xi = 1$,where $\mathbb{1}$ is a matrix with all elements equal to 1, Λ has entries of λ_{ij}^2, N has entries of υ_{ij}

for $iter = 1$ to $(burinin + nmc)$ **do**

for $i = 1$ to p **do**

$\gamma \sim \text{Gamma}(\text{shape} = \frac{n}{2} + 1, \text{rate} = \frac{s_{ii}}{2})$ ▷ sample γ

$\boldsymbol{\Omega}^{-1}_{(-i)(-i)} = \boldsymbol{\Sigma}_{(-i)(-i)} - \frac{\boldsymbol{\sigma}_{(-i)i}\boldsymbol{\sigma}^{'}_{(-i)i}}{\sigma_{ii}}$

$C = (s_{ii}\boldsymbol{\Omega}^{-1}_{(-i)(-i)} + \text{diag}(\boldsymbol{\lambda}_{(-i)i}\tau^2)^{-1})^{-1}$

$\boldsymbol{\beta} \sim \text{Normal}(-C\boldsymbol{s}_{(-i)i}, C)$ ▷ sample $\boldsymbol{\beta}$

$\boldsymbol{\omega}_{(-i)i} = \boldsymbol{\beta}, \omega_{ii} = \gamma + \boldsymbol{\beta}^{'}\boldsymbol{\Omega}^{-1}_{(-i)(-i)}\boldsymbol{\beta}$ ▷ variable transformation

$\boldsymbol{\lambda}_{(-i)i} \sim \text{InvGamma}(\text{shape} = 1, \text{scale} = \frac{1}{\boldsymbol{\upsilon}_{(-i)i}} + \frac{\boldsymbol{\omega}^2_{(-i)i}}{2\tau^2})$ ▷ sample $\boldsymbol{\lambda}$,where $\boldsymbol{\lambda}_{(-i)i}$ is a vector of length $(p-1)$ with entries $\lambda^2_{ji}, j \neq i$

$\boldsymbol{\upsilon}_{(-i)i} \sim \text{InvGamma}(1, 1 + \frac{1}{\boldsymbol{\lambda}_{(-i)i}})$ ▷ sample $\boldsymbol{\upsilon}$

Save updated $\boldsymbol{\Omega}$

$\boldsymbol{\Sigma}_{(-i)(-i)} = \boldsymbol{\Omega}^{-1}_{(-i)(-i)} + \frac{(\boldsymbol{\Omega}^{-1}_{(-i)(-i)}\boldsymbol{\beta})(\boldsymbol{\Omega}^{-1}_{(-i)(-i)}\boldsymbol{\beta})^{'}}{\gamma}, \boldsymbol{\sigma}_{(-i)i} = \frac{-(\boldsymbol{\Omega}^{-1}_{(-i)(-i)}\boldsymbol{\beta})}{\gamma}, \sigma_{ii} = \frac{1}{\gamma}$

save updated $\boldsymbol{\Sigma}, \boldsymbol{\Lambda}, N$

end for

$\tau^2 \sim \text{InvGamma}((\frac{\binom{p}{2}+1}{2}), \frac{1}{\xi} + \sum_{i,j:i<j}\frac{\omega^2_{ij}}{2\lambda^2_{ij}})$ ▷ sample τ

$\xi \sim \text{InvGamma}(1, 1 + \frac{1}{\tau^2})$ ▷ sample ξ

end for

return MC samples $\boldsymbol{\Omega}$

end function

3.3 Hit and Run sampler (HRS) for Bayesian Graphical LASSO Models

Hit and Run sampler for Bayesian graphical LASSO method, which is proposed by Oya et al.[45], is an approach to improve Wang's block Gibbs sampler method and to

guarantee the generated $\mathbf{\Omega}$ is positive definite. In Wang's method [43], the prior distribution of each diagonal element in $\mathbf{\Omega}$ is exponential distribution and that of each off-diagonal element is Laplace distribution. This prior setting that shows in (3.10) allows λ to vary for each element of the precision matrix $\mathbf{\Omega}$,

$$p(\omega_{ij}) = \begin{cases} \lambda_{ii} e^{\lambda_{ii}\omega_{ii}}, & (i = j); \\ \frac{\lambda_{ij}}{2} e^{-\lambda_{ij}|\omega_{ij}|}, & (i \neq j), \end{cases} \tag{3.10}$$

In the Gibbs sampling algorithm, the Laplace distribution in (3.10) is expressed as a scale mixture of normal distributions with the exponential distribution

$$\omega_{ij}|\tau_{ij} \sim N(0, \tau_{ij}), \tau_{ij} \sim \exp(\frac{\lambda_{ij}^2}{2}), \tag{3.11}$$

and the prior distribution for $\lambda_{ij}(1 \leq i \leq j \leq p)$ is

$$\lambda_{ij} \sim \text{Gamma}(\gamma, s).$$

Then the joint posterior distribution of $\boldsymbol{\omega} = \{\omega_{ij}\}_{i \leq j}$, $\boldsymbol{\tau} = \{\tau_{ij}\}_{i<j}$ and $\boldsymbol{\lambda} = \{\lambda_{ij}\}_{i \leq j}$ as

$$\begin{aligned} p(\boldsymbol{\omega}, \boldsymbol{\tau}, \boldsymbol{\lambda}|\boldsymbol{X}) \propto |\mathbf{\Omega}|^{\frac{n}{2}} \exp[-\frac{1}{2} tr(\boldsymbol{S}\mathbf{\Omega})] \prod_{i=1}^{p} \lambda_{ii} e^{-\lambda_{ii}\omega_{ii}} \\ \times \prod_{i<j} \frac{1}{\sqrt{2\pi\tau_{ij}}} \exp(-\frac{\omega_{ij}^2}{2\tau_{ij}}) \frac{\lambda_{ij}^2}{2} \exp(-\frac{\lambda_{ij}^2}{2}\tau_{ij}) \mathbf{1}_{M^+}(\mathbf{\Omega}) \\ \times \prod_{i \leq j} \lambda_{ij}^{\gamma-1} e^{-s\lambda_{ij}}, \end{aligned}$$

where $\mathbf{1}_{M^+}(\mathbf{\Omega})$ is the indicator function that is equal to one if $\mathbf{\Omega} \in M^+$; otherwise, it is equal to zero. According to the joint posterior distribution, we obtain the full conditional posterior distribution of $\frac{1}{\tau_{ij}}$ that follows the inverse Gaussian distribution and that of λ_{ij} follows the Gamma distribution

$$\frac{1}{\tau_{ij}}|\boldsymbol{\theta}_{-\tau_{ij}} \sim \text{InvGamma}(\frac{\lambda_{ij}}{|\omega_{ij}|}, \lambda_{ij}^2), \lambda_{ij}|\boldsymbol{\theta}_{-\lambda_{ij}} \sim \text{Gamma}(\gamma+1, s+|\omega_{ij}|), \tag{3.12}$$

where $\boldsymbol{\theta}$ represents the vector of all parameters and latent variables in the model and $\boldsymbol{\theta}_{-x}$ indicates that a parameter x is excluded from $\boldsymbol{\theta}$. To generate $\boldsymbol{\omega}$ from the full conditional

posterior distribution, the precision matrix $\boldsymbol{\Omega}$, $\boldsymbol{S}$, $\boldsymbol{\Upsilon}$ and $\boldsymbol{\Lambda}$ are partitioned based on the block Gibbs sampler by Wang et al.[43] as in (3.13)

$$\boldsymbol{\Omega} = \begin{pmatrix} \boldsymbol{\Omega}_{11} & \boldsymbol{\omega}_{12} \\ \boldsymbol{\omega}_{12}^{'} & \omega_{22} \end{pmatrix}, \boldsymbol{S} = \begin{pmatrix} \boldsymbol{S}_{11} & \boldsymbol{s}_{12} \\ \boldsymbol{s}_{12}^{'} & s_{22} \end{pmatrix}, \boldsymbol{\Upsilon} = \begin{pmatrix} \boldsymbol{\Upsilon}_{11} & \boldsymbol{\upsilon}_{12} \\ \boldsymbol{\upsilon}_{12}^{'} & 0 \end{pmatrix}, \boldsymbol{\Lambda} = \begin{pmatrix} \boldsymbol{\Lambda}_{12} \\ \lambda_{22} \end{pmatrix} \tag{3.13}$$

where $\boldsymbol{\Omega}_{11}$ is a $(p-1 \times p-1)$ matrix, $\boldsymbol{\omega}_{12}$ is a $(p-1 \times 1)$ vector and ω_{22} is a scalar, $\boldsymbol{\Upsilon}$ is a $(p \times p)$ symmetric matrix in which the off-diagonal (i, j) element is τ_{ij} and all diagonal elements are equal to zero, and λ_{22} is the element in $\boldsymbol{\Lambda}$ that corresponds with the diagonal element ω_{22} in the prior distribution (3.14). Thus, the likelihood is obtained as

$$\begin{aligned} p(\boldsymbol{X}|\boldsymbol{\Omega}) &\propto |\boldsymbol{\Omega}|^{\frac{n}{2}} \exp[-\frac{1}{2} tr(\boldsymbol{S}\boldsymbol{\Omega})] \\ &\propto |\omega_{22} - \boldsymbol{\omega}_{12}^{'} \boldsymbol{\Omega}_{11}^{-1} \boldsymbol{\omega}_{12}|^{\frac{n}{2}} |\boldsymbol{\Omega}_{11}|^{\frac{n}{2}} \\ &\times \exp[-\frac{1}{2}\{s_{22}\omega_{22} + 2\boldsymbol{s}_{12}^{'}\boldsymbol{\omega}_{12} + tr(\boldsymbol{S}_{11}\boldsymbol{\Omega}_{11})\}]. \end{aligned}$$

To generate $\boldsymbol{\Omega}$ under the positive definiteness constraint, newly generated ω_{22} and $\boldsymbol{\omega}_{12}$ must satisfy

$$\omega_{22} > \boldsymbol{\omega}_{12}^{'} \boldsymbol{\Omega}_{11}^{-1} \boldsymbol{\omega}_{12}, \tag{3.14}$$

that is, the conditional prior distribution of ω_{22} given $\boldsymbol{\omega}_{12}$ and $\boldsymbol{\Omega}_{11}$ must be

$$p(\omega_{22}|\boldsymbol{\omega}_{12}, \boldsymbol{\Omega}_{11}) \propto \lambda_{22} \exp(-\lambda_{22}\omega_{22}) \mathbf{1}_{M_{22}^{+}}(\omega_{22}), \tag{3.15}$$

where $M_{22}^{+} = \{\omega_{22} : \omega_{22} > \boldsymbol{\omega}_{12}^{'} \boldsymbol{\Omega}_{11} \boldsymbol{\omega}_{12}\}$. The full conditional posterior distribution of ω_{22} is generated as follows after ignoring the parts that do not depend on ω_{22},

$$\begin{aligned} p(\omega_{22}|\boldsymbol{\theta}_{-\omega_{22}}, \boldsymbol{X}) &\propto |\omega_{22} - \boldsymbol{\omega}_{12}^{'} \boldsymbol{\Omega}_{11}^{-1} \boldsymbol{\omega}_{12}|^{\frac{n}{2}} \exp(-\frac{s_{22}}{2}\omega_{22}) \times \exp(-\lambda_{22}\omega_{22}) \mathbf{1}_{M_{22}^{+}}(\omega_{22}) \\ &\propto |\omega_{22} - \boldsymbol{\omega}_{12}^{'} \boldsymbol{\Omega}_{11}^{-1} \boldsymbol{\omega}_{12}|^{\frac{n}{2}} \exp[-\frac{s_{22} + 2\lambda_{22}}{2}(\omega_{22} - \boldsymbol{\omega}_{12}^{'} \boldsymbol{\Omega}_{11}^{-1} \boldsymbol{\omega}_{12})] \mathbf{1}_{M_{22}^{+}}(\omega_{22}) \end{aligned}$$

which follows the shifted gamma distribution

$$\omega_{22} = u + \boldsymbol{\omega}_{12}^{'} \boldsymbol{\Omega}_{11}^{-1} \boldsymbol{\omega}_{12}, u \sim \text{Gamma}(\frac{n}{2} + 1, \frac{s_{22}}{2} + \lambda_{22}). \tag{3.16}$$

To derive the full conditional posterior distribution of $\boldsymbol{\omega}_{12}$, the conditional prior distribution of $\boldsymbol{\omega}_{12}$ is the truncated multivariate normal distribution:

$$p(\boldsymbol{\omega}_{12}|\omega_{22}, \boldsymbol{\Omega}_{11}) \propto \exp(-\frac{1}{2}\boldsymbol{\omega}_{12}^{'} \boldsymbol{D}_{\tau}^{-1} \boldsymbol{\omega}_{12}) \mathbf{1}_{M_{12}^{+}}(\boldsymbol{\omega}_{12})$$

where $M_{12}^{+} = \{\omega_{12} : \omega_{22} > \boldsymbol{\omega}_{12}^{'}\boldsymbol{\Omega}_{11}^{-1}\boldsymbol{\omega}_{12}\}$, and the full conditional posterior distribution of $\boldsymbol{\omega}_{12}$ is also the truncated multivariate normal distribution,

$$\boldsymbol{\omega}_{12}|\boldsymbol{\theta}_{-\omega_{12}}, \boldsymbol{X} \sim N(-C\boldsymbol{s}_{12}, C)\mathbf{1}_{M_{12}^{+}}(\boldsymbol{\omega}_{12}). \tag{3.17}$$

In order to generate the $\boldsymbol{\omega}_{12}$ efficiently, the Hit-and-Run algorithm [51] is applied. We first to choose a point $\boldsymbol{\alpha}$ on the unit sphere randomly as $\boldsymbol{\alpha} = \frac{\boldsymbol{z}}{\|\boldsymbol{z}\|}, \boldsymbol{z} \sim N(\mathbf{0}, \boldsymbol{I})$, and then generate a random scalar κ from the distribution:

$$\kappa \sim N(\mu_k, \sigma_k^2)\mathbf{1}_{M_{12}^{+}}(\boldsymbol{\omega}_{12} + \kappa\boldsymbol{\alpha}), \tag{3.18}$$

where $\mu_\kappa = -\frac{\boldsymbol{s}_{12}^{'}\boldsymbol{\alpha} + \boldsymbol{\omega}_{12}^{'}\boldsymbol{C}^{-1}\boldsymbol{\alpha}}{\boldsymbol{\alpha}^{'}\boldsymbol{C}^{-1}\boldsymbol{\alpha}}, \sigma_\kappa^2 = \frac{1}{\boldsymbol{\alpha}^{'}\boldsymbol{C}^{-1}\boldsymbol{\alpha}}$ and the indicator function $\mathbf{1}_{M_{12}^{+}}(\boldsymbol{\omega}_{12} + \kappa\boldsymbol{\alpha})$ is equal to one if and only if $(\boldsymbol{\omega}_{12} + \kappa\boldsymbol{\alpha})^{'}\boldsymbol{\Omega}_{11}^{-1}(\boldsymbol{\omega}_{12} + \kappa\boldsymbol{\alpha}) - \omega_{22} < 0$. Thus, the new $\boldsymbol{\omega}_{12}$ will be set as $\boldsymbol{\omega}_{12} + \kappa\boldsymbol{\alpha}$. The modified block Gibbs sampler is summarized as follows[45].

Algorithm 2 Modified Block Gibbs Sampler

For $i = 1, \cdots, p$, repeat Step 1 to Step 5.

Step 1: Rearrange $\boldsymbol{\Omega}$, $\boldsymbol{S}$, $\boldsymbol{\Gamma}$ and $\boldsymbol{\Lambda}$ so that ω_{ii} is in the place of ω_{22} in $\boldsymbol{\Omega}$ and partition them as in (3.13).

Step 2: $u \leftarrow \text{Gamma}(\frac{n}{2} + 1, \frac{s_{22}}{2} + \lambda_{22})$ and set $\omega_{22} = u + \boldsymbol{\omega}_{12}\boldsymbol{\Omega}_{11}^{-1}\boldsymbol{\omega}_{12}$.

Step 3: If $i \geqslant 2$,

(a) $\boldsymbol{z} \leftarrow N(\mathbf{0}, \boldsymbol{I})$ and set $\boldsymbol{\alpha} = \frac{\boldsymbol{z}}{\|\boldsymbol{z}\|}$.

(b) $\kappa \leftarrow N(\mu_\kappa, \sigma_\kappa^2)\mathbf{1}_{R^{+}(\kappa)}$ and update the old $\boldsymbol{\omega}_{12}$ with $\boldsymbol{\omega}_{12} + \kappa\boldsymbol{\alpha}$.

Step 4: $\lambda_{12} \leftarrow \text{Gamma}(\gamma + 1, \boldsymbol{s} + |\boldsymbol{\omega}_{12}|)$.

Step 5: $\upsilon \leftarrow \text{InvGamma}(\frac{\boldsymbol{\lambda}_{12}}{|\boldsymbol{\omega}_{12}|}, \boldsymbol{\lambda}_{12}^{\mathbf{2}})$ and set $\tau_{12} = \frac{1}{\upsilon}$.

3.4 Simulation study

3.4.1 Simulation scenarios

In this section, simulations are performed to compare the three existing Bayesian network approaches with different graphical structures, and to examine the variable selection consistency and various loss functions in both low- and high-dimensional

settings. In the low-dimensional setting, we fix $p = 10$ and evaluate three methods as $n \in \{30, 50, 70, 100\}$. While in the high-dimensional setting, we consider the cases that $p = 50$ and $n \in \{30, 50, 100, 150\}$. Data is simulated based on Multivariate Normal distribution, with means of zero and standardized covariance matrices. For both settings, we examine the following seven graphical structures:

1. AR(1): $\sigma_{ij} = 0.7^{|i-j|}$, and $\sigma_{ij} = 1$;
2. AR(2): $\omega_{ii} = 1, \omega_{i,i-1} = \omega_{i-1,i} = 0.5$,and $\omega_{i,i-2} = \omega_{i-2,i} = 0.25$ and $\omega_{i,j} = 0$ otherwise;
3. Star: $\omega_{i,i} = 1, \omega_{1,i} = \omega_{i,1} = 0.1$ and $\omega_{i,j} = 0$ otherwise;
4. Circle: $\omega_{i,i} = 1, \omega_{i-1,i} = \omega_{i,i-1} = 0.5, \omega_{1,p} = \omega_{p,1} = 0.4$ and $\omega_{ij} = 0$ otherwise;
5. Cluster: A graph that contains clusters of connections, each of which are randomly structured graphs. The number maximum number of clusters is $max\{1, \frac{p}{20}\}$, with $\omega \sim W_g(3, I_p)$;
6. Random: 90% of the off-diagonal elements set to zero, and $\Omega \sim W_g(3, I_p)$;
7. Scale-free: A graph generated with the B-A algorithm,with $p-1$ edges, and $\Omega \sim W_g(3, I_p)$.

where $\Omega = (\omega_{ij}), 1 \leq i, j \leq p$ is the precision matrix and $\sigma_{ij}, 1 \leq i, j \leq p$ is the (i, j) element of the covariance matrix Σ in the Gaussian graphical model.

In the projection method, we choose the generic value as $\tau_0 = 1$ and for the parameters that are used in (3.5), we set $\alpha = 95\%$ and assess the estimation of covariance matrix by using $\Delta u \in \{2\%, 5\%, 8\%, 10\%, 12\%, 15\%, 18\%, 20\%, 22\%, 25\%\}$ for each graphical structures in low-dimensional settings. The choice of τ_0 follows common practice, whereas the value for α is shown to be a reasonable choice for variable selection [34]. In graphical horseshoe method, since this is a shrinkage method and does not estimate off-diagonal elements of Ω to be exactly 0 [44], we use the symmetric central posterior

credible intervals at $\{40\%, 50\%, 60\%, 70\%, 80\%\}$ in the low-dimensional setting with different graphical structures. That is, if a specific posterior credible interval of an off-diagonal element of Ω does not contain zero, then that element is considered as a connected node, and *vice versa* [44]. Similarly, for the Hit and Run sampler (HRS) of Bayesian graphical lasso method, the criteria to determine whether a pair of nodes are connected or not is based on the point estimate of ω_{ij} by Monte Carlo sampling of Ω. If $\omega_{ij} \geq \varphi$, node i and node j are connected, *vice versa*. We use $\varphi \in \{10^{-4}, 5\times10^{-4}, 10^{-3}, 5\times10^{-3}, 10^{-2}, 5\times10^{-2}, 10^{-1}\}$ in low-dimensional settings. The cut-off points with the best performance are selected to further compare the three methods in both low- and high-dimensional settings. All computations are implemented by using R package GGMprojpred (R v3.6.3) for projection method, Matlab (vR2020b) for horseshoe method and Python (v3.8.1) for the HRS.

To evaluate the estimated graphical structures (edge set identification), the following four statistics are calculated in each replicate,

$$\text{Sensitivity} = \frac{TP}{TP+FN}, \quad \text{Specificity} = \frac{TN}{TN+FP},$$
$$\text{MCC} = \frac{TP\times TN - FP\times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}},$$
$$\text{F1-score} = \frac{2TP}{2TP+FP+FN},$$

where "TP","FP","TN" and "FN" denote the number of true positives, false positives, true negatives and false negatives. The range of sensitivity, specificity and F1-score is between 0 and 1, with 1 denoting perfect identification, and the range of MCC is between -1 and 1, with extreme values -1 and 1 denoting perfect misclassification and perfect classification, respectively. In addition, to evaluate the accuracy in point estimation of the precision matrix Ω, two loss functions, including KL-divergence (KL) and quadratic loss (QL), are calculated as follows:

$$\text{KL}(\Omega, \hat{\Omega}) = tr(\Omega^{-1}\hat{\Omega}) - p - log(\frac{|\hat{\Omega}|}{|\Omega|}),$$
$$\text{QL}(\Omega, \hat{\Omega}) = tr(\Omega^{-1}\hat{\Omega} - I_p)^2.$$

In the Bayesian sampling scheme, we evaluate the three methods by using 50 Monte Carlo (MC) replicates in each scenario. For each MC replicate, we run one chain with 60,000 iterations and the number of burn-in iterations is 30,000, that is, the last 30,000 samples that are drawn from the posterior distribution will be used in Bayesian inference. For each statistics, the mean and the standard deviation (SD) along with the average CPU time seconds for each method across 50 MC replicates are summarized and used to compare the performance of three methods.

3.4.2 Results

Cut-off points determination: In the low-dimensional setting, we use different cut-off points to assess the estimation of graphical structure and precision matrix, which is mentioned in 3.4.1, and select the cut-off points with the best performance to compare the three methods in both low- and high-dimensional settings. The graphs are displayed in Appendix Figure S1. In the projection method, when the cut-off points increase, all conditions show the decreasing or stable sensitivity, MCC and F1 score and slightly increasing or stable specificity, which indicate less connected edges are identified and more edges are estimated as unconnected. For the KL and QL, when the true graph is not sparse, the increasing trend in most of the conditions suggests the more unconnected edges make the estimated Ω far away from the true precision matrix Ω. Notably, almost all conditions have a comparative stable QL, which indicates the estimated Ω does not have much change with a larger cut-off point along with the relative stable sensitivity and specificity. For the Horseshoe and HRS methods, the choice of the cut-off point does not have any influence on the estimated Ω, thus, the KL and QL do not change with the larger cut-off point, and the cut-off point selection only based on the statistics that use for graphical structures estimation. Based on these four statistics, a decreasing or stable sensitivity, increasing specificity, F1 score and MCC are shown across all the conditions in both methods, which indicates more zeros in the off-diagonal element of the precision matrix are identified and the estimated Ω are close to the true value with a larger cut-off

point. In all conditions, each statistics is calculated with different sample sizes, and as sample sizes increase, the accuracy of parameter estimates is improved. However, the trend of some statistics are out of our expectation, for example, in Star, the KL shows a smaller estimation when sample size is 30 compared to that of sample size at 50,70 and 100, it may be due to the unstable estimation for small sample size in low dimensional settings. According to these results, we select the cut-off points in projection method as 10% for AR(1), AR(2), Scale-free and Star, and 5% for Circle and Cluster, 12% for Random conditions. Similar, in the Horseshoe method, the symmetric posterior credible intervals are chosen at 70% for AR(1), 60% for the rest of the conditions, and in the HRS, the cut-off point φ is selected as 0.05 for AR(1), Circle, Cluster, Random, Scale-free, 0.005 for AR(2) and 0.01 for Star conditions.

Edge set identification: The edge set identification is determined based on the cut-off point that is selected from the previous section. In the low-dimensional setting, the simulation results are summarized in Appendix Table S1 and the comparison of the three methods are displayed in Appendix Figure S2. Based on the results, the HRS always has superior performance in sensitivity across all conditions, which indicates the HRS correctly identifies most of connected edges. Compared to horseshoe method, the projection method has a higher sensitivity in Cluster and Star structures, and competitive score in Random and Scale-free conditions, but a lower sensitivity in the other situations. For the specificity, it is mostly uninfluenced by the sample size, whereas performance vary for the other methods. Both the projection and the horseshoe methods always perform well in most of the conditions, and the specificity are consistently around 85% in these two methods, which indicates the ability to identify the zero off-diagonal elements of the precision matrix Ω in low-dimensional setting. While the HRS generally has a much lower specificity than the other two methods. For the F1 score and MCC, both are consistently and substantially increasing with the larger sample sizes. The horseshoe and projection methods have very similar performance scores in most of the conditions, and

the performance scores are always higher than the estimation in HRS method.

In the high-dimensional setting, the results are summarized in Appendix Table S2 and Appendix Figure S2. Among these three methods, the HRS always has the highest sensitivity, which are consistent with the results in low-dimensional settings, and for the projection method, the lowest sensitivity across all scenarios are shown except for Scale-free and Star scenarios. For the specificity, there are almost no difference among the three methods, and most of the specificities are around 95%, which means the three methods can perfectly identify the unconnected edges. Only the HRS has a lower specificity in both AR(2) and Star conditions. To assess the accuracy of the graph structure estimation, both F1 and MCC show that the horseshoe method performs well for AR(1), AR(2) and Circle, while the HRS performs well in other conditions.

Loss on precision matrix estimation: In order to compare the accuracy in point estimation of the precision matrix Ω, the KL-divergence (KL) and quadratic loss (QL) were computed. The results are displayed in Appendix Table S3, Table S4 and Figure S3. While all methods are consistent, in that the loss decrease with larger sample sizes in both low- and high-dimensional settings, there are some notable differences. In the low-dimensional setting, the HRS method has the lowest loss. Compared to the horseshoe method, a higher loss is consistently shown in the projection method across all the conditions.

In high-dimensional setting, the results are displayed in Appendix Figure S3. The HRS always has superior performance in precision matrix estimation, and the estimation of both KL and QL are close to 0 with larger sample size. Most noticeably, the performance of the projection method is consistently reflected by low KL and QL in most conditions. For the horseshoe method, although it has a slightly higher loss than the HRS method, it is still competitive. Only for the QL in AR(2), the projection method has a higher score when $n \leq p$, but a lower score is estimated when $n \geq p$ compared to the horseshoe method.

Based on the results in both low- and high- dimensional settings, the HRS always has

the best performance in constructing precision matrix and the projection method performs worst in all conditions. Taking into account the statistics in identifying graph structures, the HRS seems most suited for low-dimensional settings in all conditions, and in the high-dimensional settings, the horseshoe method performs well in AR(1) and circle structures and the HRS method consistently performs well in others structures.

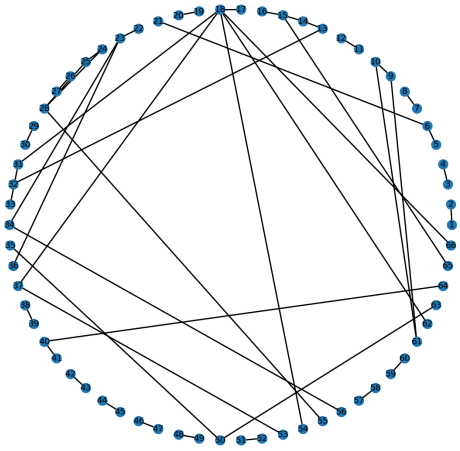
Speed of Estimation We consider the speed of estimation with the times averaged across 50 MC replicates for each graphical structures. Since the computing time of each method is based on different program languages, it may cause different computing speed. The comparison is based on the implementation of each method in the original source language, and for each graphical structure, we compare the three methods on the same computer. In general, Matlab has the fastest computing time for the matrix analysis, and R has the lowest computing speed. The Table 3.1 summarize the averaged times of AR(2) structure with the selected cut-off point and the same sample sizes for both low- and high-dimensional settings. In our result, not surprising the Horseshoe method is much faster with Matlab, the average time for low- and high-dimensional setting are 32.37 seconds and 340.2 seconds, respectively. The HRS method is the slowest in both dimensional settings. The averaged time for the low-dimensional setting is 386.1 seconds and for the high-dimensional setting is 2911 seconds. However, if the HRS method runs in the R program, it may have an even longer computing time.

Table 3.1: Average computing time of AR(2) across 50 MC replicates.

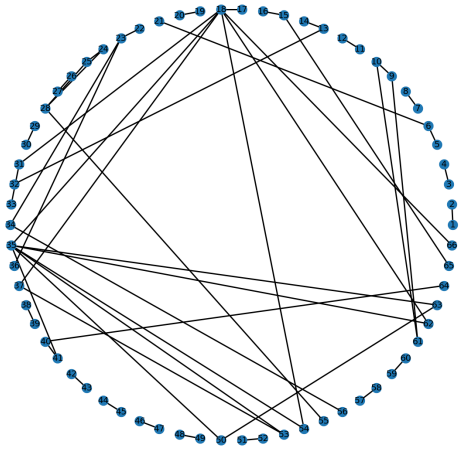
Computing time	**Projection**	**Horseshoe**	**Modified Bayesian graphical lasso**
Low-dimensional	372.1s	32.37s	386.1s
High-dimensional	2659s	340.2s	2911s

3.5 Real data analysis for network comparison

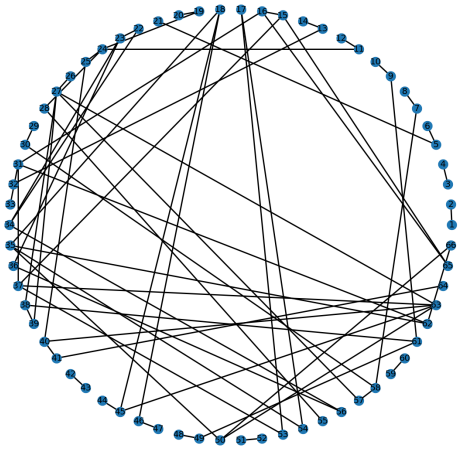
Recall, in Chapter 2, we found 1054 CpGs in cluster 4 that are maternal-transmission dominated. Among these 1054 CpGs, we use DNA methylation in children and select CpGs such that their pairwise correlations in DNA methylation are at the top 50. In total, 66 CpGs are included. Then, the three methods are applied to estimate the graphical structure and to identify the relationship between each two transmitted CpG sites. The cut-off point for each methods is chosen based on simulation findings in 3.4.2. Thus, the cut-off point for the precision method is selected as 5% and 10%, and as 60% in the horseshoe method and 0.05 in the HRS method. The graphical structures of the three methods, which are estimated based on the precision matrix, are showed in Figure 3.1. Compare to seven graphical structures in 3.4.1, the estimated graphs from the projection method and horseshoe method are similar to that of Cluster and Scale-free situations, while the inferred graph from the HRS method is totally different, which is close to the graphical structure in Star situation. Interestingly, the HRS is an extension of the horseshoe method, the different graphs from the horseshoe method may reflect the data uncertainty, which means a small change in the method lead to a huge modification of inferred graph. In addition, based on the simulation results in high-dimensional setting in 3.4.2, the HRS method always has the best performance in edge set identification and loss functions' estimation across all the scenarios. We thus postulate that the graph inferred based on the HRS method may better reflect the underlying truth. The computing time of the HRS method is 4668 seconds, which is the slowest computing speed and it is consistent with the results in the simulation study. For the projection and the horseshoe methods, it takes 2058 seconds and 701 seconds, respectively.



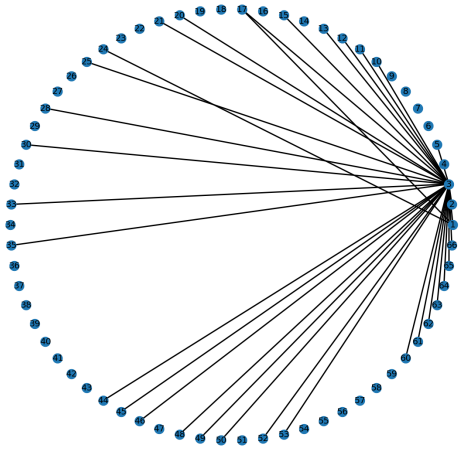
(a) Projection method (cut-off=5%)



(b) Projection method (cut-off=10%)



(c) Horseshoe method



(d) HRS method

Figure 3.1: Estimated graphical structures based on the real data

Chapter 4

Summary and Conclusion

As epigenetic markers such as DNA methylation allow us to better explain the phenomenon of disease heritability. Motivated by this, in this dissertation, I studied the transmission of DNA methylation in two perspectives: 1) developed a nested clustering method to identify the transmitted CpGs, 2) identified the interconnection among the similar variables based on the Gaussian graphical model.

I proposed a nested clustering method in a Bayesian framework to identify the transmitted CpG sites and to study the asymmetric transmission pattern of DNA methylation from parents to offspring in a general population. In the simulation study, our proposed approach can effectively identify the transmitted CpG sites and clusters with high accuracy, sensitivity and specificity. As a demonstration, we applied the method to a triad DNA methylation data set. Among the 3837 CpGs identified as transmitted sites, most of the CpGs were clustered into maternal-transmission dominated clusters, which is consistent with the findings in Han et al.[12] for the transmitted CpGs. The maternal importance in epigenetic transmission between generations has been recognized [52, 53, 54], supporting the findings in our study. It is suggested that the maternal diet before and during the pregnancy is one possible factor to affect the developmental establishment of DNA methylation in the offspring [54, 55, 56]. In addition, during the prenatal period, the biological inheritance of phenotype from one generation to the next can be a result of the transplacental passage of nutrients, metabolic signals and toxins in utero [52, 57]. However, we did not identify any paternal transmitted CpG sites, which was different from Han et al.'s study [12]. Since Han et al. pre-assumed transmitted, their identified paternal-transmission dominated CpGs could have been mis-classified. The strength of the proposed approach exists in its ability to detect inheritability. The idea of nested clustering can be applied to other situation, e.g., in multi-level designs. The method

has the potential to be generalized to incorporate non-linear transmissions, although computationally can be more challenging.

I also proposed a comprehensive comparison of three existing Gaussian graphical models on epigenetic network constructions based on the precision matrix, which were used to identify the interconnection among the CpGs showing similar patterns. The first method is the projection method, which is to estimate Gaussian graphical model with projection predictive selection, and introduces a tolerable predictive loss from a reference to determine conditional relationship. The second method is the horseshoe method, it uses a horseshoe prior to estimate the inverse-covariance matrix, and it is easily implemented by a full Gibbs sampler. The third method is the HRS method, which is an extension of the horseshoe method to guarantees the positive definiteness of the precision matrix. In the simulation study, these three methods were compared in seven different scenarios, and the results showed that in the low-dimensional setting, both the projection method and the horseshoe method had a better performance in edge set identification, but a higher loss in precision matrix estimation. While in the high-dimensional setting, the HRS always performed the best in both graphical structure identification and precision matrix estimation. In the real data analysis, based on findings in simulation studies, the inferred graph from the HRS, which looks similar to the graphical structure in the Star situation, may better reflect the underlying true graph compared to the other two methods, although a further replication assessment is desired.

In conclusion, in this dissertation, I focused on two directions to assess relationships between variables, clustering or networking. It will be an interesting direction in future studies by incorporate variable selections into network constructions. For example, when we want to identify the network of the epigentic markers, some nodes are likely singletons and are not connected to any other nodes. In this case, techniques on variable selections will be needed to identify such singletons to improve network stabilities.

References

[1] Valentina Bollati and Andrea Baccarelli. Environmental epigenetics. *Heredity*, 105(1):105–112, 2010.

[2] Diane E Handy, Rita Castro, and Joseph Loscalzo. Epigenetic modifications: basic mechanisms and role in cardiovascular disease. *Circulation*, 123(19):2145–2156, 2011.

[3] Nelís Soto-Ramírez, Syed Hasan Arshad, John W Holloway, Hongmei Zhang, Eric Schauberger, Susan Ewart, Veeresh Patil, and Wilfried Karmaus. The interaction of genetic variants and dna methylation of the interleukin-4 receptor gene increase the risk of asthma at age 18 years. *Clinical epigenetics*, 5(1):1–8, 2013.

[4] AH Ziyab, W Karmaus, JW Holloway, H Zhang, S Ewart, and SH Arshad. Dna methylation of the filaggrin gene adds to the risk of eczema associated with loss-of-function variants. *Journal of the European Academy of Dermatology and Venereology*, 27(3):e420–e423, 2013.

[5] C Ober and S Hoffjan. Asthma genetics 2006: the long and winding road to gene discovery. *Genes & Immunity*, 7(2):95–100, 2006.

[6] Scott T Weiss, Benjamin A Raby, and Angela Rogers. Asthma genetics and genomics 2009. *Current opinion in genetics & development*, 19(3):279–282, 2009.

[7] Teri A Manolio, Francis S Collins, Nancy J Cox, David B Goldstein, Lucia A Hindorff, David J Hunter, Mark I McCarthy, Erin M Ramos, Lon R Cardon, Aravinda Chakravarti, et al. Finding the missing heritability of complex diseases. *Nature*, 461(7265):747–753, 2009.

[8] Philippe Bégin and Kari C Nadeau. Epigenetic regulation of asthma and allergic disease. *Allergy, Asthma & Clinical Immunology*, 10(1):1–12, 2014.

[9] David Martino, Dörthe A Kesper, Manori Amarasekera, Hani Harb, Harald Renz, and Susan Prescott. Epigenetics in immune development and in allergic and autoimmune diseases. *Journal of reproductive immunology*, 104:43–48, 2014.

[10] Gabrielle A Lockett and John W Holloway. Genome-wide association studies in asthma; perhaps, the end of the beginning. *Current opinion in allergy and clinical immunology*, 13(5):463–469, 2013.

[11] MF Moffatt and WO Cookson. The genetics of asthma. maternal effects in atopic disease. *Clinical and experimental allergy: journal of the British Society for Allergy and Clinical Immunology*, 28:56–61, 1998.

[12] Shengtong Han, Hongmei Zhang, Gabrielle A Lockett, Nandini Mukherjee, John W Holloway, Wilfried Karmaus, et al. Identifying heterogeneous transgenerational dna

methylation sites via clustering in beta regression. *Annals of Applied Statistics*, 9(4):2052–2072, 2015.

[13] Kimberly D Siegmund, Peter W Laird, and Ite A Laird-Offringa. A comparison of cluster analysis methods using dna methylation data. *Bioinformatics*, 20(12):1896–1904, 2004.

[14] E Andres Houseman, Brock C Christensen, Ru-Fang Yeh, Carmen J Marsit, Margaret R Karagas, Margaret Wrensch, Heather H Nelson, Joseph Wiemels, Shichun Zheng, John K Wiencke, et al. Model-based clustering of dna methylation array data: a recursive-partitioning algorithm for high-dimensional data arising as a mixture of beta distributions. *BMC bioinformatics*, 9(1):1–15, 2008.

[15] Hongmei Zhang. *Analyzing High-dimensional Gene Expression and DNA Methylation Data with R*. Chapman and Hall/CRC, 2020.

[16] Douglas R Cox. Note on grouping. *Journal of the American Statistical Association*, 52(280):543–547, 1957.

[17] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the royal statistical society. series c (applied statistics)*, 28(1):100–108, 1979.

[18] Li-Xuan Qin and Steven G Self. The clustering of regression models method with applications in gene expression data. *Biometrics*, 62(2):526–533, 2006.

[19] Sarah Depaoli. Mixture class recovery in gmm under varying degrees of class separation: frequentist versus bayesian estimation. *Psychological methods*, 18(2):186, 2013.

[20] Aki Vehtari, Andrew Gelman, and Jonah Gabry. Practical bayesian model evaluation using leave-one-out cross-validation and waic. *Statistics and computing*, 27(5):1413–1432, 2017.

[21] Rens van de Schoot, Sarah Depaoli, Ruth King, Bianca Kramer, Kaspar Märtens, Mahlet G Tadesse, Marina Vannucci, Andrew Gelman, Duco Veen, Joukje Willemsen, et al. Bayesian statistics and modelling. *Nature Reviews Methods Primers*, 1(1):1–26, 2021.

[22] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.

[23] W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. 1970.

[24] Peter J Green. Reversible jump markov chain monte carlo computation and bayesian model determination. *Biometrika*, 82(4):711–732, 1995.

[25] Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence*, (6):721–741, 1984.

[26] Simon Duane, Anthony D Kennedy, Brian J Pendleton, and Duncan Roweth. Hybrid monte carlo. *Physics letters B*, 195(2):216–222, 1987.

[27] Andrew Gelman, John B Carlin, Hal S Stern, David B Dunson, Aki Vehtari, and Donald B Rubin. *Bayesian data analysis*. CRC press, 2013.

[28] Andrew Gelman and Donald B Rubin. Inference from iterative simulation using multiple sequences. *Statistical science*, 7(4):457–472, 1992.

[29] Andrew Gelman and Donald B Rubin. A single series from the gibbs sampler provides a false sense of security. *Bayesian statistics*, 4:625–631, 1992.

[30] Sheng-An Lee and Kuo-Chuan Huang. Epigenetic profiling of human brain differential dna methylation networks in schizophrenia. *BMC medical genomics*, 9(3):217–228, 2016.

[31] Albert-Laszlo Barabasi and Zoltan N Oltvai. Network biology: understanding the cell's functional organization. *Nature reviews genetics*, 5(2):101–113, 2004.

[32] Trey Ideker and Nevan J Krogan. Differential network biology. *Molecular systems biology*, 8(1):565, 2012.

[33] Abdolreza Mohammadi, Ernst C Wit, et al. Bayesian structure learning in sparse gaussian graphical models. *Bayesian Analysis*, 10(1):109–138, 2015.

[34] Donald R Williams, Juho Piironen, Aki Vehtari, and Philippe Rast. Bayesian estimation of gaussian graphical models with predictive covariance selection. *arXiv preprint arXiv:1801.05725*, 2018.

[35] J Hartlap, Patrick Simon, and P Schneider. Why your model parameter confidences might be too optimistic. unbiased estimation of the inverse covariance matrix. *Astronomy & Astrophysics*, 464(1):399–404, 2007.

[36] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008.

[37] Jianqing Fan, Yang Feng, and Yichao Wu. Network exploration via the adaptive lasso and scad penalties. *The annals of applied statistics*, 3(2):521, 2009.

[38] Onureena Banerjee, Laurent El Ghaoui, and Alexandre d'Aspremont. Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data. *The Journal of Machine Learning Research*, 9:485–516, 2008.

[39] Ming Yuan and Yi Lin. Model selection and estimation in the gaussian graphical model. *Biometrika*, 94(1):19–35, 2007.

[40] Ming Yuan. High dimensional inverse covariance matrix estimation via linear programming. *The Journal of Machine Learning Research*, 11:2261–2286, 2010.

[41] Tony Cai, Weidong Liu, and Xi Luo. A constrained 1 minimization approach to sparse precision matrix estimation. *Journal of the American Statistical Association*, 106(494):594–607, 2011.

[42] Debdeep Pati, Anirban Bhattacharya, Natesh S Pillai, and David Dunson. Posterior contraction in sparse bayesian factor models for massive covariance matrices. *The Annals of Statistics*, 42(3):1102–1130, 2014.

[43] Hao Wang et al. Bayesian graphical lasso models and efficient posterior computation. *Bayesian Analysis*, 7(4):867–886, 2012.

[44] Yunfan Li, Bruce A Craig, and Anindya Bhadra. The graphical horseshoe estimator for inverse covariance matrices. *Journal of Computational and Graphical Statistics*, 28(3):747–757, 2019.

[45] Sakae Oya and Teruo Nakatsuma. An efficient gibbs sampling algorithm for bayesian graphical lasso models with the positive definite constraint on the precision matrix. *arXiv preprint arXiv:2001.04657*, 2020.

[46] Syed Hasan Arshad and David Wallace Hide. Effect of environmental factors on the development of allergic disorders in infancy. *Journal of Allergy and Clinical Immunology*, 90(2):235–241, 1992.

[47] Aki Vehtari and Janne Ojanen. A survey of bayesian predictive methods for model assessment, selection and comparison. *Statistics Surveys*, 6:142–228, 2012.

[48] Aki Vehtari, Daniel Simpson, Andrew Gelman, Yuling Yao, and Jonah Gabry. Pareto smoothed importance sampling. *arXiv preprint arXiv:1507.02646*, 2015.

[49] Nicole Krämer, Juliane Schäfer, and Anne-Laure Boulesteix. Regularized estimation of large-scale gene association networks using graphical gaussian models. *BMC bioinformatics*, 10(1):1–24, 2009.

[50] Enes Makalic and Daniel F Schmidt. A simple sampler for the horseshoe estimator. *IEEE Signal Processing Letters*, 23(1):179–182, 2015.

[51] Claude JP Bélisle, H Edwin Romeijn, and Robert L Smith. Hit-and-run algorithms for generating multivariate distributions. *Mathematics of Operations Research*, 18(2):255–266, 1993.

[52] Dao H Ho. Transgenerational epigenetics: The role of maternal effects in cardiovascular development, 2014.

[53] James P Curley, Rahia Mashoodh, and Frances A Champagne. Epigenetics and the origins of paternal effects. *Hormones and behavior*, 59(3):306–314, 2011.

[54] Paula Dominguez-Salas, Sophie E Moore, Maria S Baker, Andrew W Bergen, Sharon E Cox, Roger A Dyer, Anthony J Fulford, Yongtao Guan, Eleonora Laritsky, Matt J Silver, et al. Maternal nutrition at conception modulates dna methylation of human metastable epialleles. *Nature communications*, 5(1):1–7, 2014.

[55] Robert A Waterland and Randy L Jirtle. Transposable elements: targets for early nutritional effects on epigenetic gene regulation. *Molecular and cellular biology*, 23(15):5293–5300, 2003.

[56] Robert A Waterland, Dana C Dolinoy, Juan-Ru Lin, Charlotte A Smith, Xin Shi, and Kajal G Tahiliani. Maternal methyl supplements increase offspring dna methylation at axin fused. *genesis*, 44(9):401–406, 2006.

[57] Marcus Pembrey, Richard Saffery, Lars Olov Bygren, et al. Human transgenerational responses to early-life experience: potential impact on development, health and biomedical research. *Journal of medical genetics*, 51(9):563–572, 2014.

Appendices

Appendix A

Computing programs utilized in Chapters 2 and 3

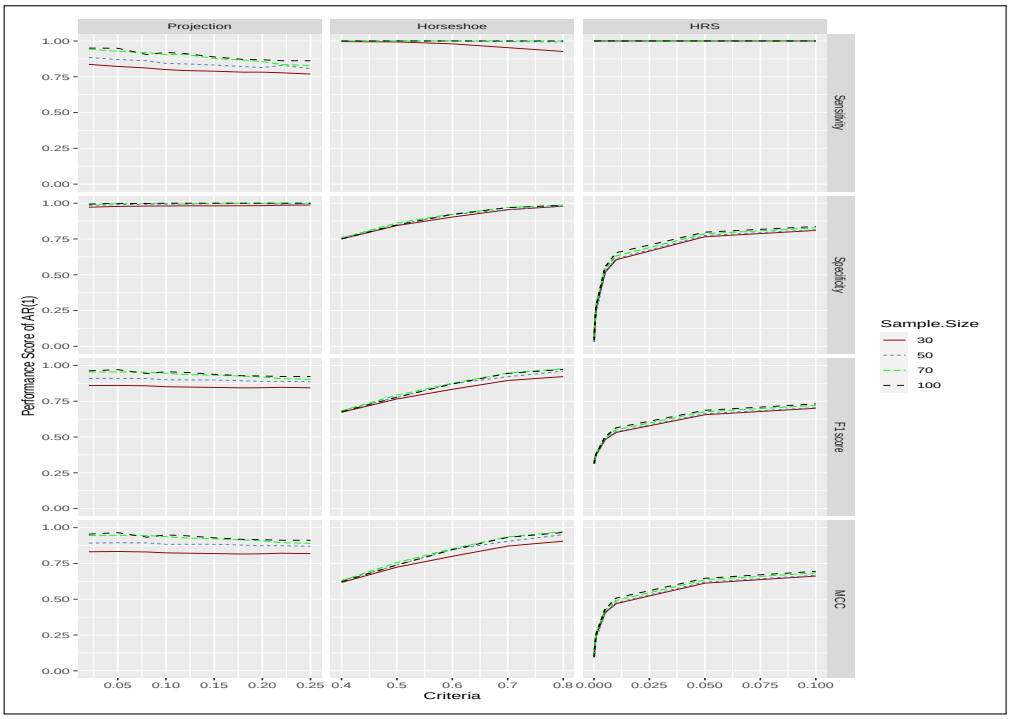


Figure A.1: Performance score for both edge set identification and risk estimation with different cut-off points across all situations

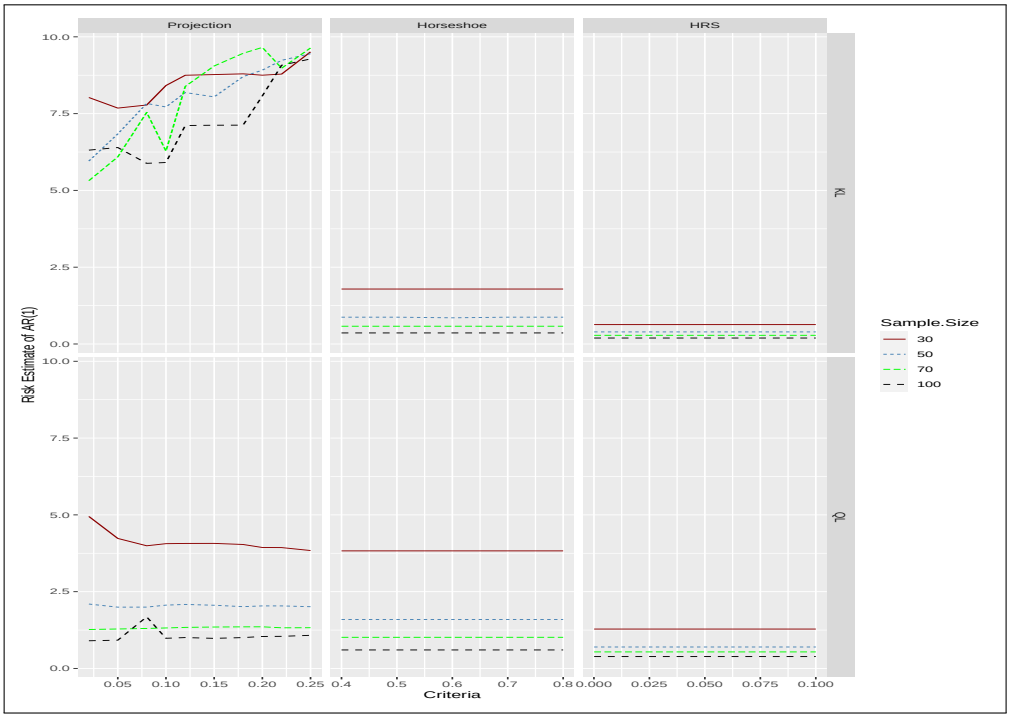


Figure A.1 (Continued)

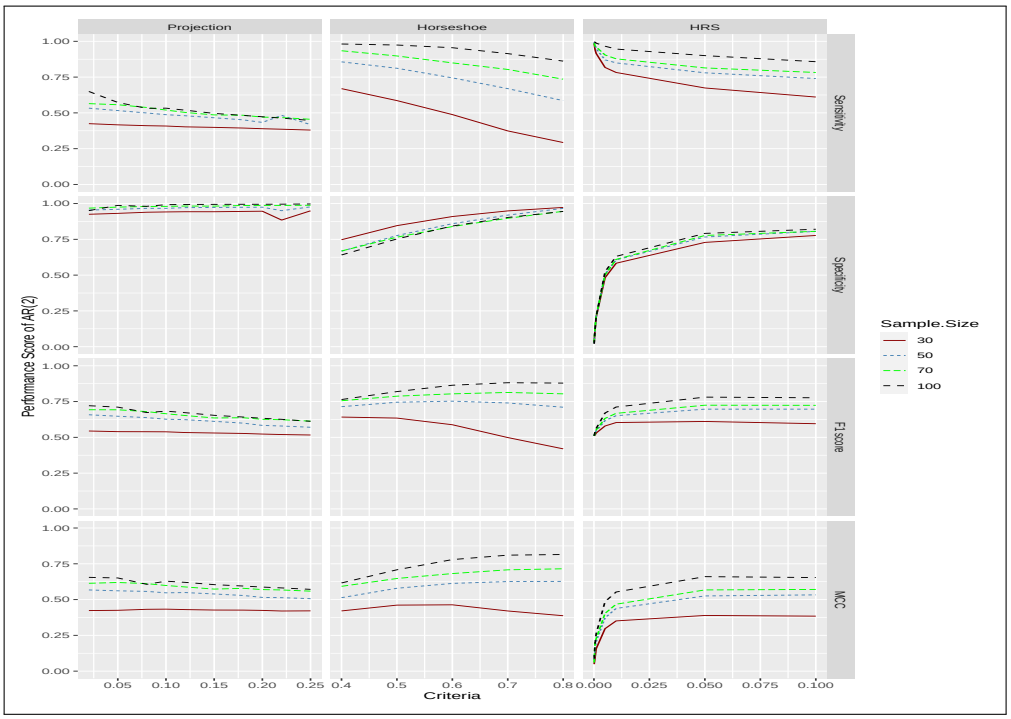


Figure A.1 (Continued)

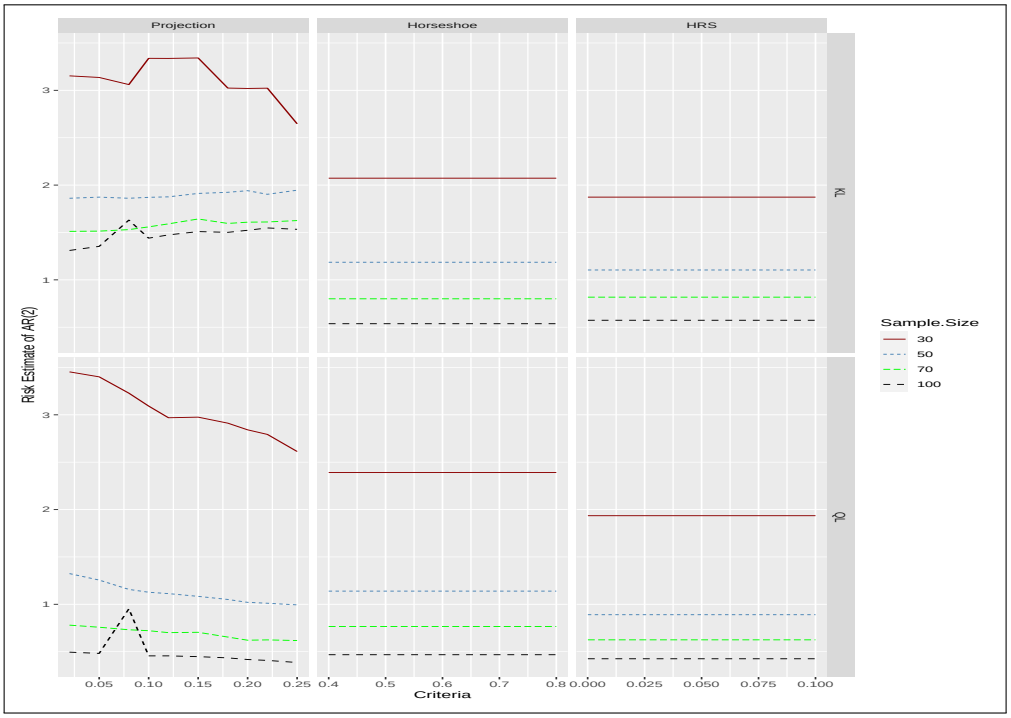


Figure A.1 (Continued)

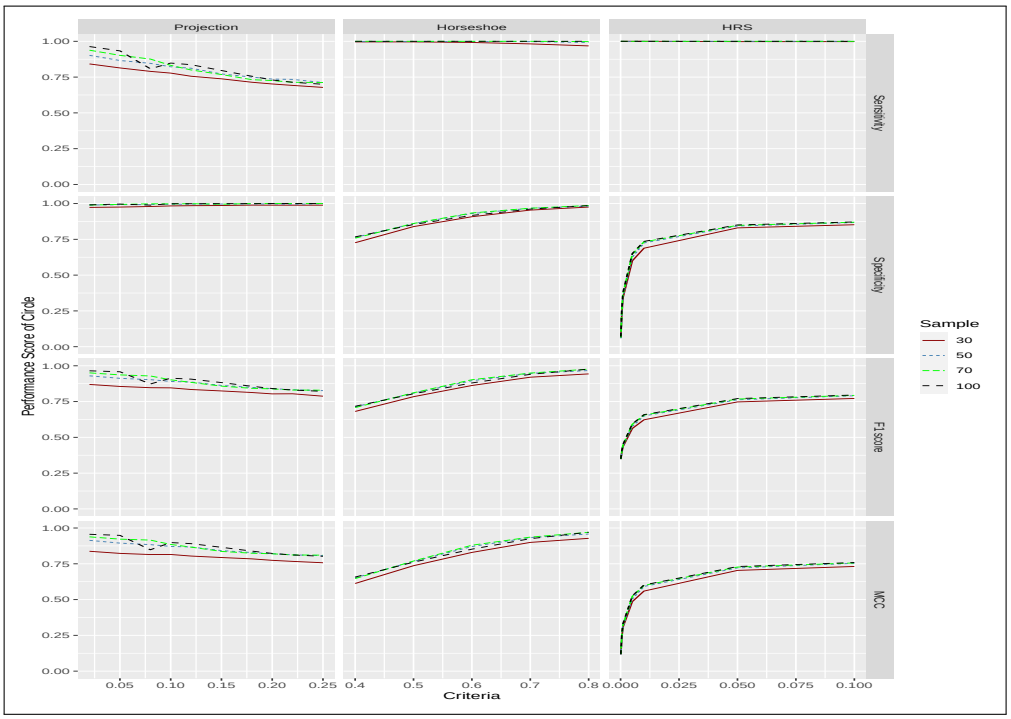


Figure A.1 (Continued)

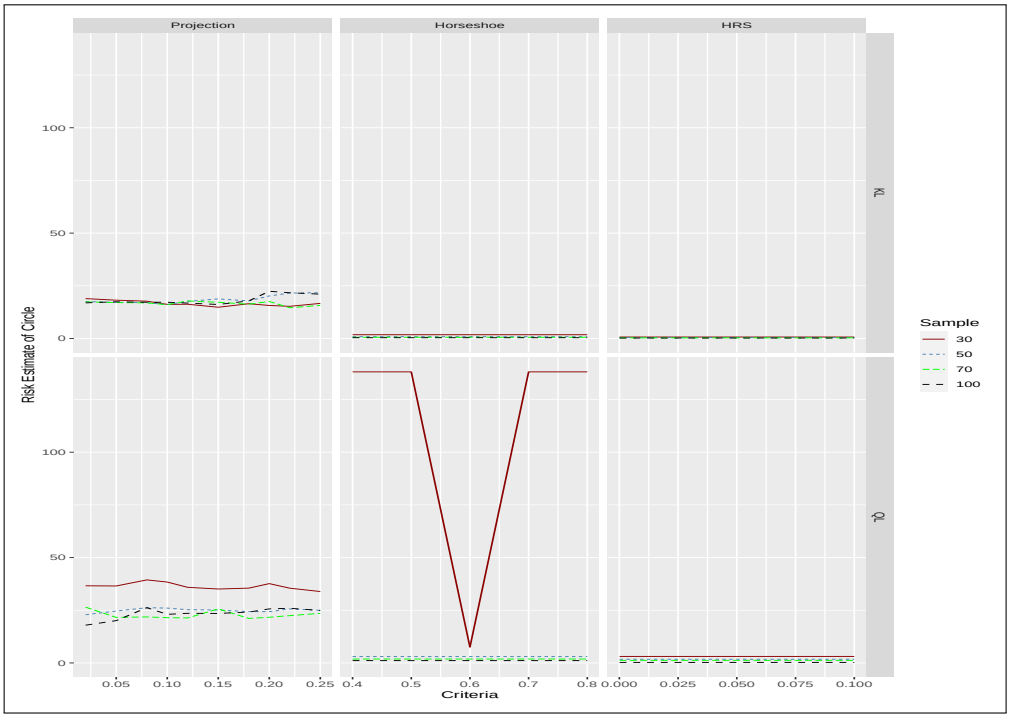


Figure A.1 (Continued)

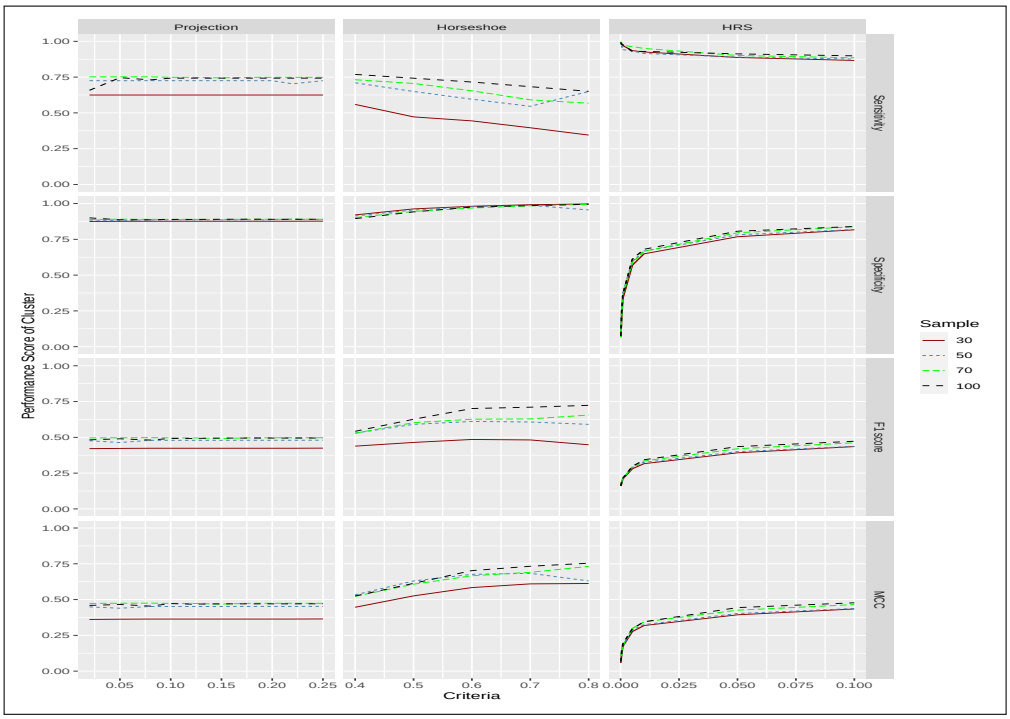


Figure A.1 (Continued)

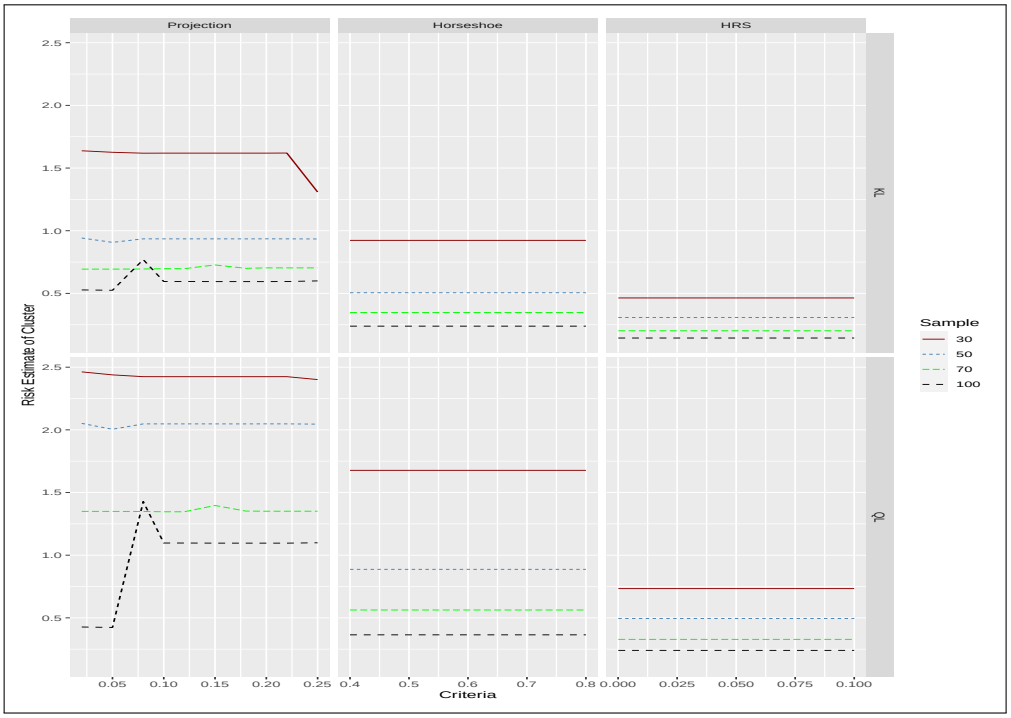


Figure A.1 (Continued)

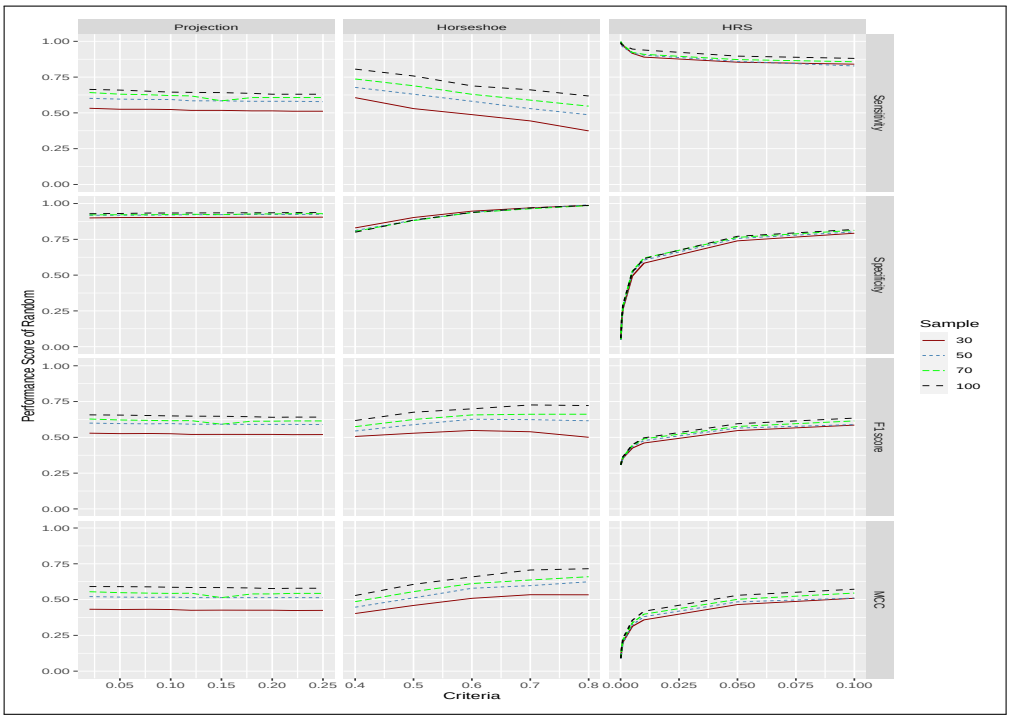


Figure A.1 (Continued)

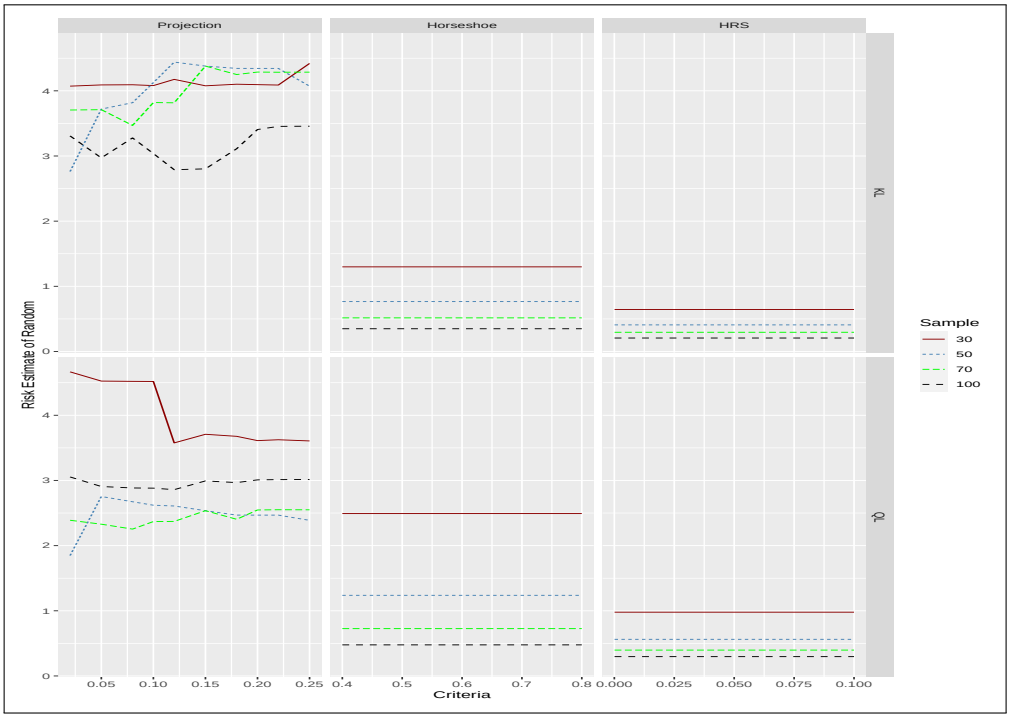


Figure A.1 (Continued)

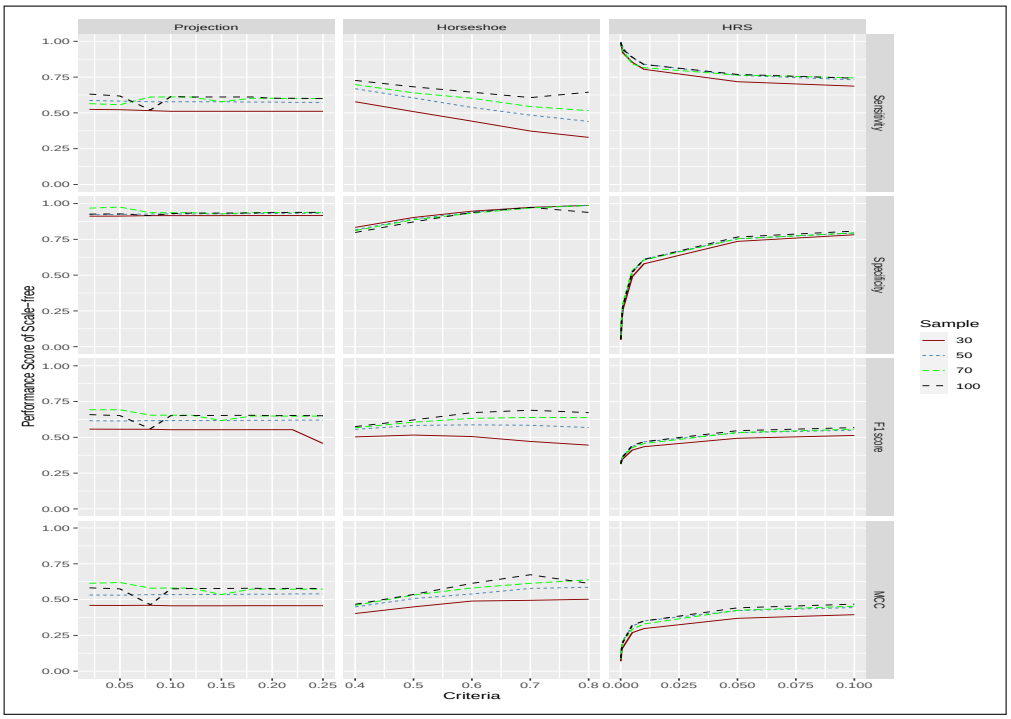


Figure A.1 (Continued)

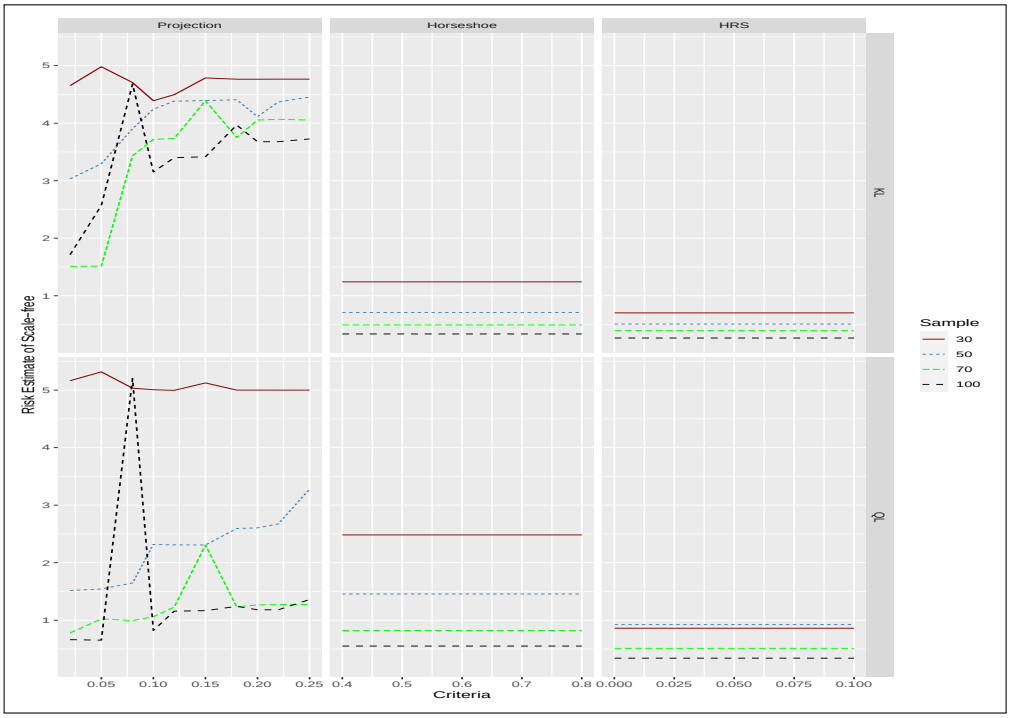


Figure A.1 (Continued)

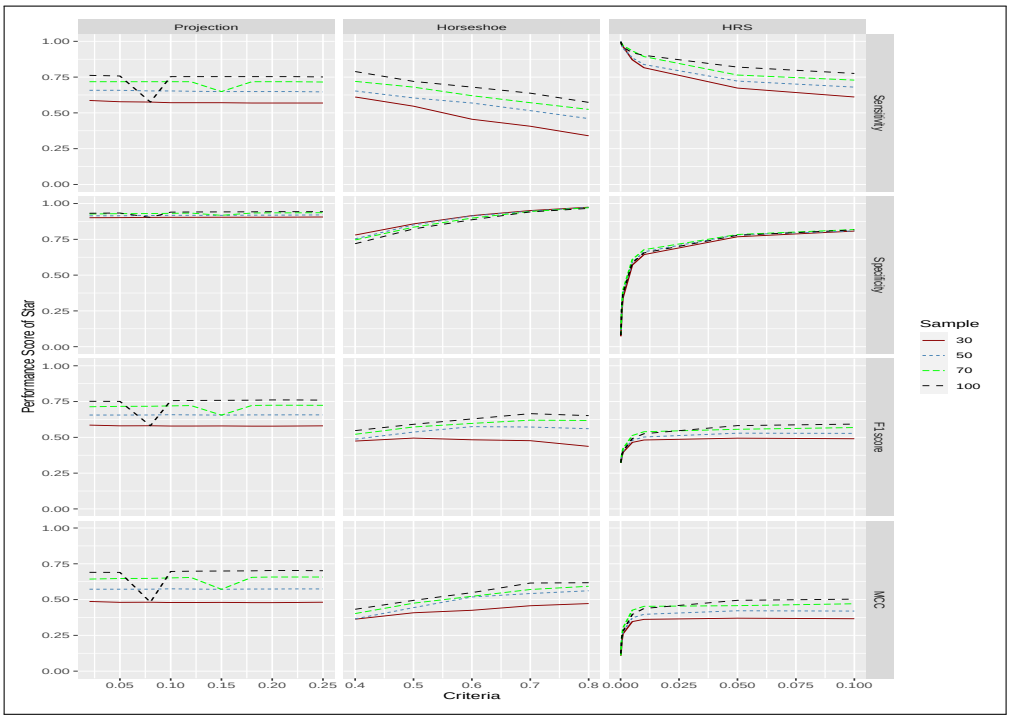


Figure A.1 (Continued)

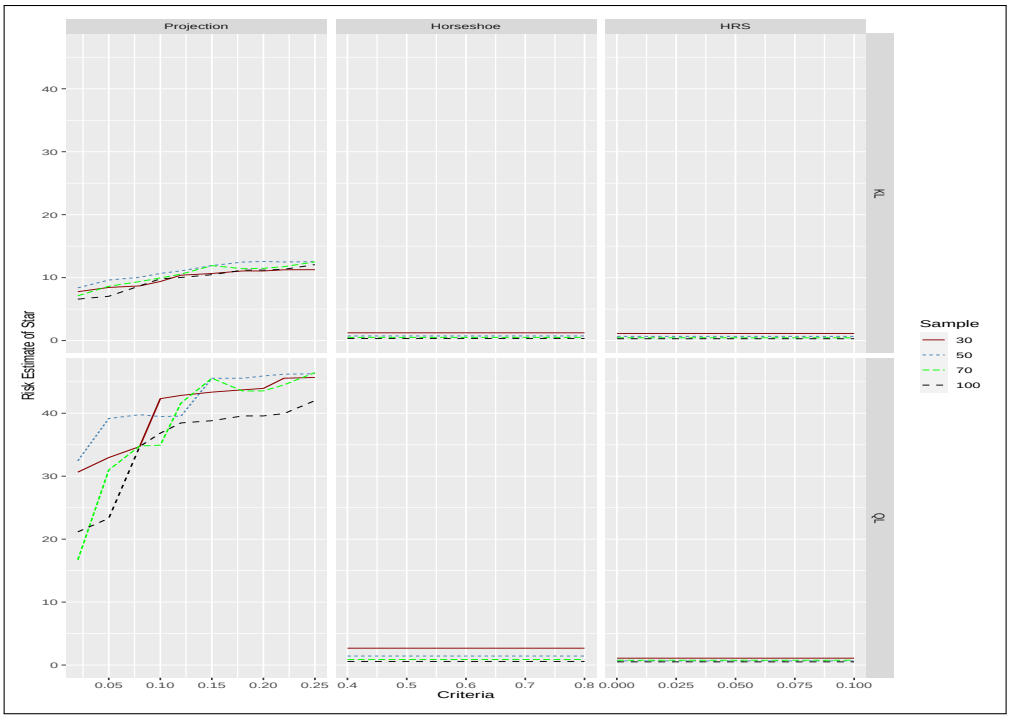


Figure A.1 (Continued)

Table A.1: Result of performance measures for graphical structure learning in low-dimensional setting. Mean and standard deviation (Std) are summarized across 50 MCMC replicates.

AR(1): n=30			
Statistics	**Projection**	**Horseshoe**	**Modified Bayesian graphical lasso**
Sensitivity	0.8000 (0.0952)	0.9533 (0.0780)	1.000 (0)
Specificity	0.9806 (0.0246)	0.9544 (0.0345)	0.7659 (0.0490)
F1-score	0.8514 (0.0718)	0.8947 (0.0692)	0.6556 (0.0480)
MCC	0.8240 (0.0871)	0.8713 (0.0848)	0.6121 (0.0531)
n=50			
Sensitivity	0.8435 (0.1012)	0.9956 (0.0220)	1.0000 (0)
Specificity	0.9932 (0.0156)	0.9550 (0.0466)	0.7741 (0.0420)
F1-score	0.8995 (0.0635)	0.9212 (0.0752)	0.6628 (0.0406)
MCC	0.8837 (0.0722)	0.9050 (0.0897)	0.6202 (0.0450)
n=70			
Sensitivity	0.9051 (0.0972)	1.000 (0)	1.000 (0)
Specificity	0.9988 (0.0056)	0.9694 (0.0373)	0.7863 (0.0561)
F1-score	0.9452 (0.0541)	0.9467 (0.0619)	0.6771 (0.0536)
MCC	0.9370 (0.0608)	0.9358 (0.0737)	0.6357 (0.0596)
n=100			
Sensitivity	0.9206 (0.0935)	1.0000 (0)	1.0000 (0)
Specificity	1.0000 (0)	0.9694 (0.0233)	0.7976 (0.0376)
F1-score	0.9561 (0.0537)	0.9442 (0.0412)	0.6867 (0.0393)
MCC	0.9496 (0.0599)	0.9322 (0.0497)	0.6464 (0.0432)

Table A.1.(Continued)

AR(2): n=30			
Statistics	**Projection**	**Horseshoe**	**Modified Bayesian graphical lasso**
Sensitivity	0.4082 (0.0976)	0.4882 (0.1236)	0.8176 (0.0930)
Specificity	0.9414 (0.0437)	0.9093 (0.0764)	0.4818 (0.0961)
F1-score	0.5387 (0.0978)	0.5881 (0.0954)	0.5793 (0.0452)
MCC	0.4331 (0.1254)	0.4635 (0.0962)	0.2967 (0.0968)
n=50			
Sensitivity	0.4874 (0.1060)	0.7447 (0.0970)	0.8706 (0.0928)
Specificity	0.9665 (0.0367)	0.8593 (0.0709)	0.5061 (0.1090)
F1-score	0.6268 (0.0931)	0.7527 (0.0678)	0.6168 (0.0584)
MCC	0.5472 (0.1086)	0.6130 (0.1048)	0.3723 (0.1166)
n=70			
Sensitivity	0.5198 (0.0978)	0.8494 (0.0922)	0.9047 (0.0692)
Specificity	0.9803 (0.0310)	0.8393 (0.0775)	0.5018 (0.1016)
F1-score	0.6650 (0.0861)	0.8039 (0.0675)	0.6323 (0.0560)
MCC	0.5984 (0.0899)	0.6818 (0.1120)	0.4034 (0.1106)
n=100			
Sensitivity	0.5329 (0.1124)	0.9553 (0.0564)	0.9659 (0.0396)
Specificity	0.9921 (0.0166)	0.8429 (0.0629)	0.5230 (0.0872)
F1-score	0.6830 (0.0971)	0.8641 (0.0549)	0.6700 (0.0429)
MCC	0.6286 (0.0944)	0.7794 (0.0924)	0.4854 (0.0809)

Table A.1.(Continued)

Circle: n=30			
Statistics	**Projection**	**Horseshoe**	**Modified Bayesian graphical lasso**
Sensitivity	0.8140 (0.1010)	0.9920 (0.0274)	0.9980 (0.0141)
Specificity	0.9754 (0.0265)	0.9080 (0.0511)	0.8300 (0.0375)
F1-score	0.8557 (0.0795)	0.8622 (0.0682)	0.7475 (0.0425)
MCC	0.8227 (0.0985)	0.8295 (0.0837)	0.7038 (0.0498)
n=50			
Sensitivity	0.8660 (0.0961)	0.9980 (0.0141)	1.000 (0)
Specificity	0.9920 (0.0153)	0.9303 (0.0500)	0.8430 (0.0299)
F1-score	0.9124 (0.0642)	0.8955 (0.0707)	0.7625 (0.0338)
MCC	0.8945 (0.0759)	0.8707 (0.0868)	0.7213 (0.0383)
n=70			
Sensitivity	0.9020 (0.0820)	1.000 (0)	1.000 (0)
Specificity	0.9937 (0.0145)	0.9343 (0.0523)	0.8475 (0.0296)
F1-score	0.9364 (0.0565)	0.9026 (0.0720)	0.7677 (0.0330)
MCC	0.9223 (0.0694)	0.8798 (0.0877)	0.7272 (0.0374)
n=100			
Sensitivity	0.9333 (0.0724)	1.000 (0)	1.000 (0)
Specificity	0.9964 (0.0095)	0.9171 (0.0598)	0.8505 (0.0240)
F1-score	0.9582 (0.0483)	0.8800 (0.0751)	0.7708 (0.0276)
MCC	0.9486 (0.0591)	0.8523 (0.0906)	0.7307 (0.0313)

Table A.1.(Continued)

Cluster: n=30			
Statistics	**Projection**	**Horseshoe**	**Modified Bayesian graphical lasso**
Sensitivity	0.6248 (0.2741)	0.4441 (0.3029)	0.8871 (0.1863)
Specificity	0.8760 (0.0351)	0.9810 (0.0306)	0.7680 (0.0694)
F1-score	0.4227 (0.1642)	0.4852 (0.2733)	0.3919 (0.1503)
MCC	0.3629 (0.2069)	0.5839 (0.1829)	0.3936 (0.1564)
n=50			
Sensitivity	0.7248 (0.2481)	0.5954 (0.3178)	0.8898 (0.1869)
Specificity	0.8824 (0.0411)	0.9781 (0.0299)	0.7815 (0.0650)
F1-score	0.4638 (0.1908)	0.6111 (0.2758)	0.4001 (0.1349)
MCC	0.4396 (0.2028)	0.6764 (0.1913)	0.4027 (0.1439)
n=70			
Sensitivity	0.7519 (0.2332)	0.6542 (0.3229)	0.9013 (0.1651)
Specificity	0.8906 (0.0345)	0.9687 (0.0411)	0.7960 (0.0648)
F1-score	0.4970 (0.1851)	0.6266 (0.2610)	0.4203 (0.1452)
MCC	0.4743 (0.1918)	0.6665 (0.2040)	0.4259 (0.1439)
n=100			
Sensitivity	0.7460 (0.2300)	0.7155 (0.2846)	0.9123 (0.1705)
Specificity	0.8866 (0.0346)	0.9752 (0.0267)	0.8068 (0.0558)
F1-score	0.4898 (0.1809)	0.7014 (0.2085)	0.4352 (0.1469)
MCC	0.4658 (0.1882)	0.7024 (0.1949)	0.4440 (0.1436)

Table A.1.(Continued)

Random: n=30			
Statistics	**Projection**	**Horseshoe**	**Modified Bayesian graphical lasso**
Sensitivity	0.5175 (0.1702)	0.4878 (0.2100)	0.8541 (0.1336)
Specificity	0.9027 (0.0432)	0.9469 (0.0450)	0.7394 (0.0758)
F1-score	0.5204 (0.1523)	0.5481 (0.1970)	0.5471 (0.1291)
MCC	0.4253 (0.1813)	0.5088 (0.1760)	0.4652 (0.1410)
n=50			
Sensitivity	0.5842 (0.1671)	0.5810 (0.2052)	0.8597 (0.1061)
Specificity	0.9222 (0.0414)	0.9388 (0.0658)	0.7561 (0.0689)
F1-score	0.5918 (0.1407)	0.6266 (0.1842)	0.5649 (0.0956)
MCC	0.5136 (0.1745)	0.5791 (0.1827)	0.4841 (0.1071)
n=70			
Sensitivity	0.6187 (0.1727)	0.6296 (0.2060)	0.8724 (0.1089)
Specificity	0.9253 (0.0376)	0.9368 (0.0624)	0.7638 (0.0640)
F1-score	0.6168 (0.1324)	0.6566 (0.1740)	0.5760 (0.1104)
MCC	0.5447 (0.1593)	0.6118 (0.1692)	0.5018 (0.1141)
n=100			
Sensitivity	0.6433 (0.1412)	0.6886 (0.1895)	0.8966 (0.1114)
Specificity	0.9344 (0.0390)	0.9380 (0.0560)	0.7713 (0.0676)
F1-score	0.6480 (0.1284)	0.6991 (0.1598)	0.5951 (0.1181)
MCC	0.5845 (0.1464)	0.6584 (0.1453)	0.5300 (0.1203)

Table A.1.(Continued)

Scale-free: n=30			
Statistics	**Projection**	**Horseshoe**	**Modified Bayesian graphical lasso**
Sensitivity	0.5111 (0.1365)	0.4422 (0.2228)	0.7178 (0.1423)
Specificity	0.9156 (0.0385)	0.9467 (0.0561)	0.7356 (0.0608)
F1-score	0.5532 (0.1434)	0.5055 (0.2053)	0.4935 (0.0985)
MCC	0.4565 (0.1782)	0.4897 (0.1692)	0.3691 (0.1414)
n=50			
Sensitivity	0.5778 (0.1634)	0.5378 (0.2012)	0.7622 (0.1347)
Specificity	0.9272 (0.0423)	0.9400 (0.0497)	0.7537 (0.0652)
F1-score	0.6170 (0.1645)	0.5873 (0.1744)	0.5324 (0.1031)
MCC	0.5346 (0.2022)	0.5391 (0.1687)	0.4232 (0.1423)
n=70			
Sensitivity	0.6111 (0.1527)	0.6022 (0.1906)	0.7644 (0.1449)
Specificity	0.9361 (0.0410)	0.9322 (0.0533)	0.7537 (0.0711)
F1-score	0.6548 (0.1583)	0.6325 (0.1597)	0.5336 (0.1031)
MCC	0.5801 (0.1960)	0.5809 (0.1613)	0.4252 (0.1471)
n=100			
Sensitivity	0.6133 (0.1575)	0.6445 (0.1851)	0.7689 (0.1601)
Specificity	0.9300 (0.0570)	0.9378 (0.0478)	0.7663 (0.0723)
F1-score	0.6512 (0.1694)	0.6726 (0.1482)	0.5467 (0.1152)
MCC	0.5747 (0.2122)	0.6133 (0.1695)	0.4423 (0.1619)

Table A.1.(Continued)

Star: n=30			
Statistics	**Projection**	**Horseshoe**	**Modified Bayesian graphical lasso**
Sensitivity	0.5711 (0.1995)	0.4556 (0.2168)	0.8156 (0.1762)
Specificity	0.9039 (0.0394)	0.9156 (0.0632)	0.6434 (0.1051)
F1-score	0.5790 (0.1909)	0.4830 (0.1907)	0.4815 (0.1212)
MCC	0.4794 (0.2322)	0.4252 (0.1752)	0.3618 (0.1848)
n=50			
Sensitivity	0.6533 (0.2149)	0.5689 (0.2195)	0.8378 (0.1947)
Specificity	0.9178 (0.0538)	0.9133 (0.0649)	0.6629 (0.1093)
F1-score	0.6581 (0.2128)	0.5748 (0.1653)	0.5025 (0.1204)
MCC	0.5751 (0.2641)	0.5183 (0.1523)	0.3971 (0.1811)
n=70			
Sensitivity	0.7178 (0.1854)	0.6200 (0.2246)	0.8933 (0.1679)
Specificity	0.9317 (0.0450)	0.8983 (0.0649)	0.6776 (0.1093)
F1-score	0.7198 (0.1823)	0.5974 (0.1579)	0.5392 (0.1182)
MCC	0.6514 (0.2258)	0.5228 (0.1760)	0.4528 (0.1666)
n=100			
Sensitivity	0.7533 (0.1669)	0.6800 (0.2003)	0.9022 (0.1597)
Specificity	0.9400 (0.0447)	0.8867 (0.0614)	0.6551 (0.1048)
F1-score	0.7563 (0.1664)	0.6287 (0.1347)	0.5254 (0.1071)
MCC	0.6965 (0.2081)	0.5485 (0.1549)	0.4378 (0.1557)

Table A.2: Result of performance measures for graphical structure learning in high-dimensional setting. Mean and standard deviation (Std) are summarized across 50 MCMC replicates.

AR(1): n=30			
Statistics	**Projection**	**Horseshoe**	**Modified Bayesian graphical lasso**
Sensitivity	0.6641 (0.0636)	0.8604 (0.0423)	1.000 (0)
Specificity	0.9974 (0.0026)	0.9979 (0.0014)	0.9660 (0.0135)
F1-score	0.7684 (0.0506)	0.9000 (0.0308)	0.7148 (0.0756)
MCC	0.7727 (0.0495)	0.8975 (0.0316)	0.7343 (0.0646)
n=50			
Sensitivity	0.8139 (0.0351)	0.9857 (0.0171)	1.000 (0)
Specificity	0.9994 (0.0012)	0.9987 (0.0011)	0.9698 (0.0104)
F1-score	0.8904 (0.0264)	0.9776 (0.0187)	0.7358 (0.0621)
MCC	0.8909 (0.0266)	0.9768 (0.0194)	0.7523 (0.0534)
n=100			
Sensitivity	0.8830 (0.0382)	1.000 (0)	1.000 (0)
Specificity	0.9999 (0.0002)	0.9993 (0.0007)	0.9740 (0.0066)
F1-score	0.9368 (0.0213)	0.9926 (0.0080)	0.7612 (0.0428)
MCC	0.9365 (0.0209)	0.9923 (0.0083)	0.7741 (0.0371)
n=150			
Sensitivity	0.9008 (0.0357)	1.000 (0)	1.000 (0)
Specificity	1.000 (0.0002)	0.9994 (0.0008)	0.9762 (0.0050)
F1-score	0.9471 (0.0197)	0.9928 (0.0094)	0.7756 (0.0341)
MCC	0.9466 (0.0193)	0.9926 (0.0097)	0.7867 (0.0297)

Table A.2.(Continued)

AR(2): n=30			
Statistics	**Projection**	**Horseshoe**	**Modified Bayesian graphical lasso**
Sensitivity	0.1666 (0.0341)	0.0938 (0.0379)	1.000 (0)
Specificity	0.9886 (0.0039)	0.9976 (0.0021)	0.8640 (0.0374)
F1-score	0.2551 (0.0431)	0.1649 (0.0603)	0.5613 (0.0686)
MCC	0.2761 (0.0434)	0.2500 (0.0616)	0.5818 (0.0620)
n=50			
Sensitivity	0.3412 (0.0257)	0.3299 (0.0529)	1.000 (0)
Specificity	0.9892 (0.0037)	0.9920 (0.0036)	0.8805 (0.0391)
F1-score	0.4652 (0.0277)	0.4614 (0.0498)	0.5954 (0.0819)
MCC	0.4742 (0.0322)	0.4839 (0.0391)	0.6127 (0.0743)
n=100			
Sensitivity	0.4674 (0.0345)	0.7932 (0.0437)	1.000 (0)
Specificity	0.9966 (0.0024)	0.9793 (0.0041)	0.9004 (0.0398)
F1-score	0.6196 (0.0297)	0.7798 (0.0300)	0.6420 (0.0962)
MCC	0.6386 (0.0269)	0.7611 (0.0324)	0.6550 (0.0873)
n=150			
Sensitivity	0.5216 (0.0431)	0.9470 (0.0244)	1.000 (0)
Specificity	0.9964 (0.0027)	0.9781 (0.0040)	0.9126 (0.0312)
F1-score	0.6665 (0.0312)	0.8604 (0.0220)	0.6680 (0.0821)
MCC	0.6784 (0.0261)	0.8517 (0.0234)	0.6784 (0.0748)

Table A.2.(Continued)

Circle: n=30			
Statistics	**Projection**	**Horseshoe**	**Modified Bayesian graphical lasso**
Sensitivity	0.7380 (0.0704)	0.9864 (0.0178)	1.000 (0)
Specificity	0.9990 (0.0011)	0.9985 (0.0011)	0.9792 (0)
F1-score	0.8365 (0.0508)	0.9767 (0.0191)	0.8000 (0)
MCC	0.8400 (0.0463)	0.9758 (0.0199)	0.8079 (0)
n=50			
Sensitivity	0.9044 (0.0407)	0.9984 (0.0055)	1.000 (0)
Specificity	0.9998 (0.0006)	0.9995 (0.0006)	0.9792 (0)
F1-score	0.9472 (0.0221)	0.9941 (0.0080)	0.8000 (0)
MCC	0.9466 (0.0218)	0.9940 (0.0083)	0.8079 (0)
n=100			
Sensitivity	0.9556 (0.0226)	1.000 (0)	1.000 (0)
Specificity	1.000 (0.0001)	0.9996 (0.0006)	0.9792 (0)
F1-score	0.9770 (0.0117)	0.9953 (0.0066)	0.8000 (0)
MCC	0.9764 (0.0119)	0.9951 (0.0069)	0.8079 (0)
n=150			
Sensitivity	0.9628 (0.0252)	1.000 (0)	1.000 (0)
Specificity	1.000 (0)	0.9998 (0.0005)	0.9792 (0)
F1-score	0.9809 0.0132)	0.9972 (0.0056)	0.8000 (0)
MCC	0.9804 (0.0134)	0.9971 (0.0058)	0.8079 (0)

Table A.2.(Continued)

Cluster: n=30			
Statistics	**Projection**	**Horseshoe**	**Modified Bayesian graphical lasso**
Sensitivity	0.2910 (0.0621)	0.3067 (0.0985)	0.9792 (0.0185)
Specificity	0.9885 (0.0040)	0.9961 (0.0024)	0.9545 (0.0220)
F1-score	0.3971 (0.0776)	0.4415 (0.1129)	0.7480 (0.0876)
MCC	0.4044 (0.0838)	0.4870 (0.0913)	0.7531 (0.0794)
n=50			
Sensitivity	0.3861 (0.0589)	0.4061 (0.0898)	0.9791 (0.0179)
Specificity	0.9897 (0.0032)	0.9955 (0.0020)	0.9598 (0.0205)
F1-score	0.5004 (0.0641)	0.5464 (0.0888)	0.7695 (0.0819)
MCC	0.5037 (0.0665)	0.5712 (0.0746)	0.7728 (0.0744)
n=100			
Sensitivity	0.4580 (0.0522)	0.5524 (0.0796)	0.9789 (0.0185)
Specificity	0.9932 (0.0031)	0.9945 (0.0027)	0.9667 (0.0156)
F1-score	0.5866 (0.0527)	0.6746 (0.0676)	0.7973 (0.0686)
MCC	0.5948 (0.0549)	0.6786 (0.0629)	0.7981 (0.0628)
n=150			
Sensitivity	0.5019 (0.0535)	0.6286 (0.0701)	0.9789 (0.0181)
Specificity	0.9943 (0.0035)	0.9941 (0.0035)	0.9699 (0.0111)
F1-score	0.6315 (0.0494)	0.7322 (0.0538)	0.8099 (0.0544)
MCC	0.6394 (0.0501)	0.7302 (0.0524)	0.8097 (0.0502)

Table A.2.(Continued)

Random: n=30			
Statistics	**Projection**	**Horseshoe**	**Modified Bayesian graphical lasso**
Sensitivity	0.1236 (0.0253)	0.1998 (0.0316)	0.9778 (0.0103)
Specificity	0.9832 (0.0051)	0.9789 (0.0054)	0.9442 (0.0296)
F1-score	0.2070 (0.0371)	0.3129 (0.0404)	0.8897 (0.0470)
MCC	0.2230 (0.0417)	0.3111 (0.0361)	0.8656 (0.0553)
n=50			
Sensitivity	0.1730 (0.0282)	0.3144 (0.0375)	0.9784 (0.0100)
Specificity	0.9831 (0.0064)	0.9723 (0.0070)	0.9517 (0.0242)
F1-score	0.2784 (0.0393)	0.4405 (0.0385)	0.9019 (0.0399)
MCC	0.2920 (0.0476)	0.4113 (0.0349)	0.8800 (0.0471)
n=100			
Sensitivity	0.2203 (0.0327)	0.4812 (0.0443)	0.9780 (0.0101)
Specificity	0.9844 (0.0064)	0.9620 (0.0077)	0.9563 (0.0195)
F1-score	0.3427 (0.0434)	0.5890 (0.0358)	0.9087 (0.0336)
MCC	0.3545 (0.0492)	0.5335 (0.0358)	0.8881 (0.0398)
n=150			
Sensitivity	0.2435 (0.0330)	0.5686 (0.0423)	0.9778 (0.0102)
Specificity	0.9839 (0.0089)	0.9570 (0.0078)	0.9609 (0.0165)
F1-score	0.3720 (0.0428)	0.6536 (0.0318)	0.9164 (0.0286)
MCC	0.3796 (0.0520)	0.5924 (0.0356)	0.8972 (0.0341)

Table A.2.(Continued)

Scale-free: n=30			
Statistics	**Projection**	**Horseshoe**	**Modified Bayesian graphical lasso**
Sensitivity	0.3396 (0.1074)	0.2559 (0.1034)	0.9714 (0.0292)
Specificity	0.9852 (0.0047)	0.9973 (0.0023)	0.9470 (0.0198)
F1-score	0.3984 (0.1162)	0.3762 (0.1230)	0.6068 (0.0896)
MCC	0.3862 (0.1206)	0.4352 (0.0993)	0.6372 (0.0776)
n=50			
Sensitivity	0.4653 (0.0847)	0.3927 (0.1117)	0.9739 (0.0264)
Specificity	0.9858 (0.0043)	0.9962 (0.0024)	0.9563 (0.0197)
F1-score	0.5160 (0.0955)	0.5205 (0.1064)	0.6536 (0.0940)
MCC	0.5016 (0.1003)	0.5507 (0.0860)	0.6775 (0.0809)
n=100			
Sensitivity	0.5646 (0.0831)	0.5490 (0.1096)	0.9722 (0.0276)
Specificity	0.9897 (0.0034)	0.9953 (0.0028)	0.9562 (0.0202)
F1-score	0.6225 (0.0857)	0.6553 (0.0839)	0.6529 (0.0957)
MCC	0.6125 (0.0891)	0.6637 (0.0755)	0.6767 (0.0826)
n=150			
Sensitivity	0.6146 (0.0678)	0.6233 (0.0962)	0.9727 (0.0296)
Specificity	0.9910 (0.0034)	0.9947 (0.0030)	0.9599 (0.0179)
F1-score	0.6708 (0.0733)	0.7088 (0.0710)	0.6701 (0.0860)
MCC	0.6620 (0.0764)	0.7095 (0.0688)	0.6913 (0.0738)

Table A.2.(Continued)

Star: n=30			
Statistics	**Projection**	**Horseshoe**	**Modified Bayesian graphical lasso**
Sensitivity	0.1274 (0.0476)	0.0898 (0.0558)	0.9049 (0.0397)
Specificity	0.9814 (0.0062)	0.9914 (0.0068)	0.9198 (0.0407)
F1-score	0.1629 (0.0599)	0.1335 (0.0811)	0.4987 (0.1283)
MCC	0.1466 (0.0680)	0.1617 (0.0833)	0.5325 (0.1115)
n=50			
Sensitivity	0.3004 (0.0763)	0.1694 (0.0720)	0.9127 (0.0406)
Specificity	0.9824 (0.0073)	0.9876 (0.0087)	0.9178 (0.0402)
F1-score	0.3535 (0.1003)	0.2234 (0.0870)	0.4924 (0.1212)
MCC	0.3395 (0.1130)	0.2405 (0.0725)	0.5286 (0.1061)
n=100			
Sensitivity	0.3673 (0.0628)	0.3196 (0.0989)	0.9090 (0.0419)
Specificity	0.9870 (0.0070)	0.9849 (0.0093)	0.9291 (0.0409)
F1-score	0.4430 (0.0937)	0.3741 (0.1004)	0.5329 (0.1356)
MCC	0.4396 (0.1074)	0.3717 (0.0891)	0.5626 (0.1183)
n=150			
Sensitivity	0.3865 (0.0467)	0.4098 (0.1138)	0.9167 (0.0430)
Specificity	0.9885 (0.0076)	0.9841 (0.0093)	0.9339 (0.0308)
F1-score	0.4711 (0.0812)	0.4501 (0.0959)	0.5415 (0.1153)
MCC	0.4703 (0.0988)	0.4454 (0.0773)	0.5716 (0.1006)

Table A.3: Result of loss functions in low-dimensional setting. Mean and standard deviation (Std) are summarized across 50 MCMC replicates. KL: KL-divergence; QL: quadratic loss.

AR(1): n=30			
Statistics	**Projection**	**Horseshoe**	**Modified Bayesian graphical lasso**
KL	8.4150 (7.3755)	1.7890 (0.6476)	0.6314 (0.3302)
QL	4.0625 (2.1172)	3.8279 (2.0575)	1.2824 (0.7488)
n=50			
KL	7.7165 (7.3873)	0.8725 (0.3764)	0.3932 (0.2258)
QL	2.0616 (1.5147)	1.5928 (1.0045)	0.6985 (0.4714)
n=70			
KL	6.2799 (7.1008)	0.5780 (0.2266)	0.2821 (0.1155)
QL	1.3183 (0.7784)	1.0138 (0.5728)	0.5385 (0.3000)
n=100			
KL	5.9086 (6.9686)	0.3602 (0.0995)	0.1941 (0.0777)
QL	0.9819 (0.5423)	0.6024 (0.2728)	0.3872 (0.2391)

Table A.3.(Continued)

AR(2): n=30			
Statistics	**Projection**	**Horseshoe**	**Modified Bayesian graphical lasso**
KL	3.3380 (3.3030)	2.0730 (0.4276)	1.8730 (0.4812)
QL	3.0923 (1.7535)	2.3908 (1.1569)	1.9353 (0.9157)
n=50			
KL	1.8710 (0.3309)	1.1856 (0.2678)	1.1040 (0.2973)
QL	1.1273 (0.7008)	1.1392 (0.5519)	0.8904 (0.4195)
n=70			
KL	1.5592 (0.2869)	0.8000 (0.1984)	0.8172 (0.2225)
QL	0.7214 (0.4137)	0.7656 (0.4269)	0.6251 (0.3447)
n=100			
KL	1.4400 (0.3378)	0.5371 (0.1275)	0.5733 (0.1780)
QL	0.4558 (0.2298)	0.4680 (0.2097)	0.4253 (0.2065)

Table A.3.(Continued)

Circle: n=30			
Statistics	**Projection**	**Horseshoe**	**Modified Bayesian graphical lasso**
KL	18.1597 (5.7973)	1.7733 (0.6228)	0.6838 (0.3599)
QL	36.5220 (28.1204)	7.5050 (6.4050)	3.0819 (1.8244)
n=50			
KL	17.1422 (5.2629)	0.8524 (0.2856)	0.3865 (0.1442)
QL	24.6310 (14.7921)	3.0049 (1.7751)	1.7520 (0.9605)
n=70			
KL	17.0275 (4.8914)	0.5638 (0.2066)	0.2854 (0.1260)
QL	21.5970 (10.8850)	1.8480 (1.2738)	1.2259 (0.7248)
n=100			
KL	17.3971 (3.4254)	0.3671 (0.1030)	0.2087 (0.0817)
QL	20.0760 (9.7387)	1.1070 (0.5435)	0.2404 (0.1606)

Table A.3.(Continued)

Cluster: n=30			
Statistics	**Projection**	**Horseshoe**	**Modified Bayesian graphical lasso**
KL	1.6258 (3.2869)	0.9230 (0.4256)	0.4637 (0.2261)
QL	2.4386 (4.2105)	1.6769 (1.0385)	0.7334 (0.4375)
n=50			
KL	0.9071 (2.4994)	0.5052 (0.2612)	0.3074 (0.1457)
QL	2.0046 (6.9230)	0.8870 (0.5459)	0.4946 (0.3108)
n=70			
KL	0.6935 (2.3658)	0.3475 (0.1536)	0.2019 (0.0890)
QL	1.3495 (5.2652)	0.5628 (0.3374)	0.3282 (0.1740)
n=100			
KL	0.5248 (1.9225)	0.2386 (0.1030)	0.1439 (0.0550)
QL	0.4234 (0.6957)	0.3645 (0.2449)	0.2404 (0.1606)

Table A.3.(Continued)

Random: n=30			
Statistics	**Projection**	**Horseshoe**	**Modified Bayesian graphical lasso**
KL	4.1756 (6.0887)	1.2987 (0.5053)	0.6441 (0.3423)
QL	3.2636 (4.8469)	2.4920 (1.8198)	0.9785 (0.6801)
n=50			
KL	4.4408 (6.6648)	0.7662 (0.3174)	0.4084 (0.2236)
QL	2.2609 (4.0945)	1.2371 (0.8476)	0.5619 (0.2816)
n=70			
KL	3.8161 (6.2976)	0.5159 (0.1678)	0.2948 (0.1503)
QL	1.8697 (4.3907)	0.7264 (0.4133)	0.3967 (0.2115)
n=100			
KL	2.7893 (5.3444)	0.3491 (0.1053)	0.2066 (0.1043)
QL	1.8679 (6.0988)	0.4771 (0.3389)	0.2967 (0.2392)

Table A.3.(Continued)

Scale-free: n=30			
Statistics	**Projection**	**Horseshoe**	**Modified Bayesian graphical lasso**
KL	4.3900 (6.5497)	1.2419 (0.4618)	0.7028 (0.2889)
QL	5.0062 (9.5662)	2.4830 (1.8757)	0.8609 (0.5368)
n=50			
KL	4.2431 (6.7165)	0.7102 (0.2469)	0.5079 (0.2297)
QL	2.3167 (5.1336)	1.4559 (1.7603)	0.9244 (2.0100)
n=70			
KL	3.7185 (6.8916)	0.4937 (0.1696)	0.3938 (0.1701)
QL	1.0602 (1.4162)	0.8210 (0.7190)	0.5112 (0.5880)
n=100			
KL	3.1518 (6.4948)	0.3363 (0.1157)	0.2656 (0.0989)
QL	0.8231 (1.4082)	0.5511 (0.4676)	0.3405 (0.2975)

Table A.3.(Continued)

Star: n=30			
Statistics	**Projection**	**Horseshoe**	**Modified Bayesian graphical lasso**
KL	9.3869 (11.8001)	1.2318 (0.4249)	1.1337 (0.4927)
QL	30.2840 (86.6764)	2.6794 (2.0988)	1.0932 (0.8729)
n=50			
KL	10.6668 (11.9631)	0.7375 (0.2803)	0.6720 (0.2905)
QL	29.7791 (101.4884)	1.4311 (1.2146)	0.6555 (0.4451)
n=70			
KL	9.9519 (11.0803)	0.5120 (0.1554)	0.4468 (0.2597)
QL	26.2034 (87.7785)	0.8825 (0.5371)	0.7659 (2.1462)
n=100			
KL	9.8102 (11.8114)	0.3442 (0.1005)	0.2964 (0.1645)
QL	24.3418 (74.6403)	0.5726 (0.4065)	0.5077 (1.1274)

Table A.4: Result of loss functions in high-dimensional setting. Mean and standard deviation (Std) are summarized across 50 MCMC replicates. KL: KL-divergence; QL: quadratic loss.

AR(1): n=30			
Statistics	**Projection**	**Horseshoe**	**Modified Bayesian graphical lasso**
KL	26.1600 (8.6145)	14.3000 (2.6939)	1.9460 (0.3763)
QL	27.1600 (18.7939)	28.3400 (8.1384)	6.2220 (1.2300)
n=50			
KL	19.4690 (8.4831)	5.9120 (1.1010)	1.1780 (0.2934)
QL	10.4960 (3.3298)	10.1070 (2.7416)	3.6670 (0.8904)
n=100			
KL	16.4320 (12.2765)	2.0610 (0.2788)	0.5602 (0.1240)
QL	5.7620 (1.3140)	3.1330 (0.5896)	1.7004 (0.3696)
n=150			
KL	14.2496 (12.9208)	1.2426 (0.1646)	0.3686 (0.0802)
QL	5.2820 (1.1132)	1.8410 (0.4353)	1.1328 (0.2895)

Table A.4.(Continued)

AR(2): n=30			
Statistics	**Projection**	**Horseshoe**	**Modified Bayesian graphical lasso**
KL	19.0800 (6.7418)	15.9500 (1.2697)	2.1010 (0.5618)
QL	16.5970 (15.8884)	14.4840 (4.8188)	5.0600 (1.8568)
n=50			
KL	13.4300 (2.7481)	11.5420 (0.7929)	1.1753 (0.3045)
QL	7.6290 (2.7713)	8.1390 (1.9958)	2.5350 (0.7626)
n=100			
KL	9.0210 (0.5024)	5.1910 (0.4443)	0.5698 (0.1311)
QL	2.2400 (0.4379)	3.5320 (0.6557)	1.1175 (0.2825)
n=150			
KL	7.9120 (0.6159)	2.9410 (0.2713)	0.3856 (0.0851)
QL	1.3756 (0.3057)	1.9540 (0.3601)	0.7426 (0.1612)

Table A.4.(Continued)

Circle: n=30			
Statistics	**Projection**	**Horseshoe**	**Modified Bayesian graphical lasso**
KL	63.7709 (50.4438)	11.6210 (2.1768)	2.2860 (0.6556)
QL	2012.00 (752.6661)	69.5000 (19.5284)	37.0800 (12.1474)
n=50			
KL	59.2300 (27.0504)	4.9760 (0.8768)	1.2486 (0.3379)
QL	1961.20 (603.3898)	30.7600 (9.3331)	21.7400 (6.5833)
n=100			
KL	45.4200 (18.9776)	1.9960 (0.3208)	0.5874 (0.1605)
QL	1789.50 (476.6465)	13.3200 (3.5615)	10.9230 (3.0168)
n=150			
KL	49.4700 (20.4378)	1.2341 (0.1668)	0.3832 (0.1028)
QL	1606.10 (359.9216)	8.3580 (2.1458)	7.2610 (2.0190)

Table A.4.(Continued)

Cluster: n=30			
Statistics	**Projection**	**Horseshoe**	**Modified Bayesian graphical lasso**
KL	25.4370 (16.6449)	12.2240 (2.5161)	1.9550 (0.4291)
QL	145.4410 (269.6536)	24.3400 (11.3659)	8.2090 (5.6101)
n=50			
KL	20.5063 (16.1369)	6.8170 (1.2034)	1.1261 (0.2353)
QL	66.4450 (140.2168)	10.4490 (4.9833)	4.7440 (3.6803)
n=100			
KL	17.4245 (14.2820)	2.9970 (0.4254)	0.5432 (0.1112)
QL	35.6540 (70.7918)	3.5800 (1.9779)	2.1960 (1.8408)
n=150			
KL	14.3580 (12.9571)	1.9070 (0.2781)	0.3647 (0.0645)
QL	30.9011 (75.4363)	2.1568 (1.3444)	1.4467 (1.1921)

Table A.4.(Continued)

Random: n=30			
Statistics	**Projection**	**Horseshoe**	**Modified Bayesian graphical lasso**
KL	45.7590 (13.0165)	24.1600 (3.8380)	1.8750 (0.5171)
QL	317.7300 (676.2177)	60.1500 (52.8117)	14.8050 (22.2438)
n=50			
KL	44.9160 (91.3314)	14.3700 (1.7701)	1.1042 (0.2912)
QL	264.5920 (837.2126)	25.0100 (17.5673)	8.4600 (10.3008)
n=100			
KL	33.7510 (72.4596)	6.8680 (0.7736)	0.5447 (0.1241)
QL	263.4460 (800.0613)	9.0080 (6.8188)	4.6710 (8.0472)
n=150			
KL	26.6511 (72.4952)	4.3740 (0.4195)	0.3504 (0.0714)
QL	88.6150 (171.8838)	5.2880 (4.4390)	3.0926 (5.3955)

Table A.4.(Continued)

Scale-free: n=30			
Statistics	**Projection**	**Horseshoe**	**Modified Bayesian graphical lasso**
KL	15.3950 (11.0631)	8.9610 (1.8298)	1.9250 (0.3468)
QL	25.2650 (28.6937)	16.0910 (6.7530)	5.6930 (6.2733)
n=50			
KL	15.7091 (14.7793)	4.9350 (0.8235)	0.5605 (0.1134)
QL	21.5300 (63.2493)	7.3150 (3.9851)	1.5703 (1.5237)
n=100			
KL	13.8578 (14.0106)	2.1920 (0.4048)	0.5621 (0.1153)
QL	16.7230 (63.7257)	2.6550 (1.6012)	1.5693 (1.5212)
n=150			
KL	12.8580 (11.3878)	1.3910 (0.1973)	0.3875 (0.0702)
QL	11.1676 (29.0119)	1.6292 (0.9858)	1.0774 (1.1088)

Table A.4.(Continued)

Star: n=30			
Statistics	**Projection**	**Horseshoe**	**Modified Bayesian graphical lasso**
KL	23.7080 (17.5092)	8.6040 (2.3933)	2.8090 (1.5820)
QL	64.9100 (110.2868)	19.5490 (11.1543)	8.7600 (16.5585)
n=50			
KL	22.2140 (12.2961)	4.7910 (1.0365)	1.8036 (0.9332)
QL	53.7640 (83.6636)	7.9220 (3.8689)	4.9280 (7.8631)
n=100			
KL	24.6840 (14.7878)	2.2840 (0.4889)	1.0709 (0.6083)
QL	57.8408 (97.0576)	2.9557 (1.7581)	2.8257 (5.5285)
n=150			
KL	21.3460 (12.5221)	1.4672 (0.3012)	1.0532 (1.7258)
QL	56.6180 (96.2333)	1.9947 (2.7445)	2.4523 (6.6675)

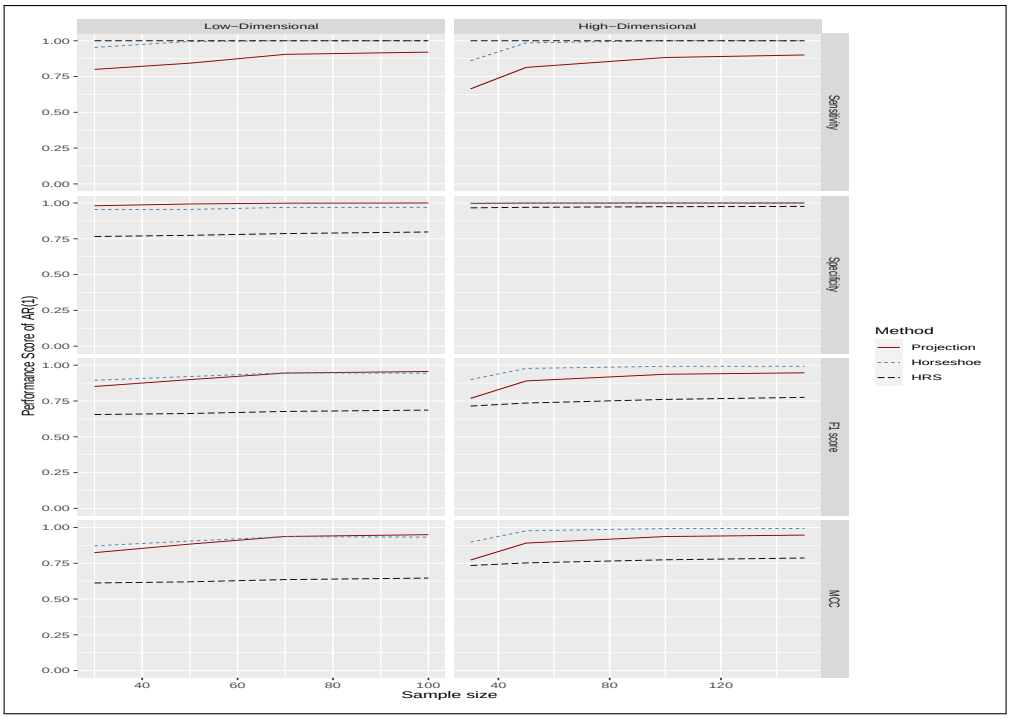


Figure A.2: Comparison of accuracy for edge set identification

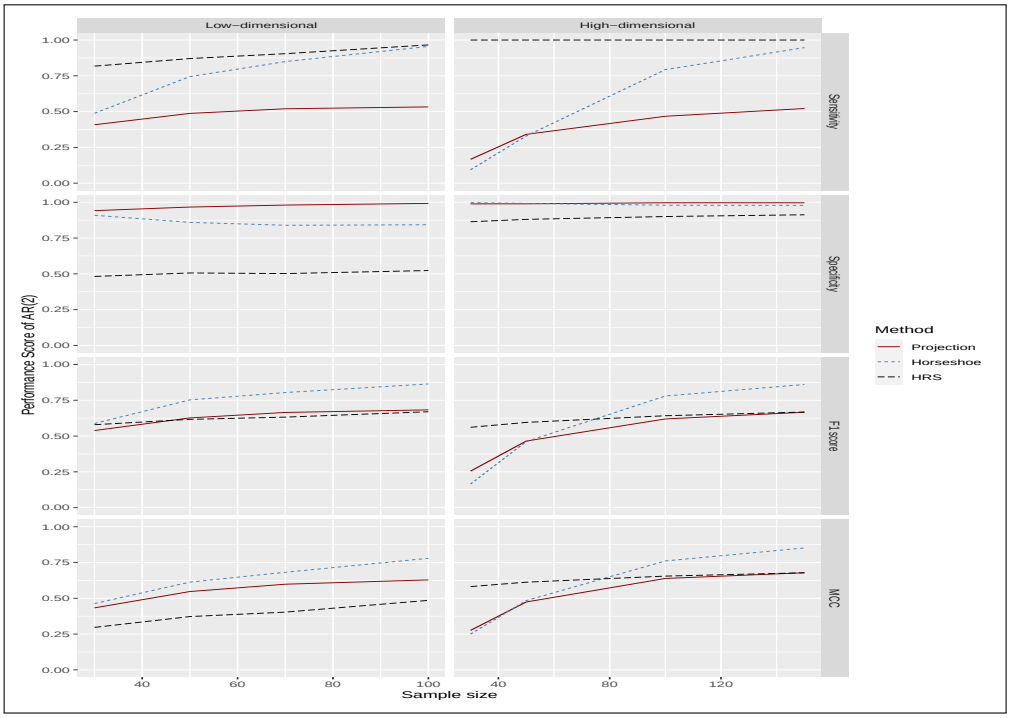


Figure A.2.(Continued)

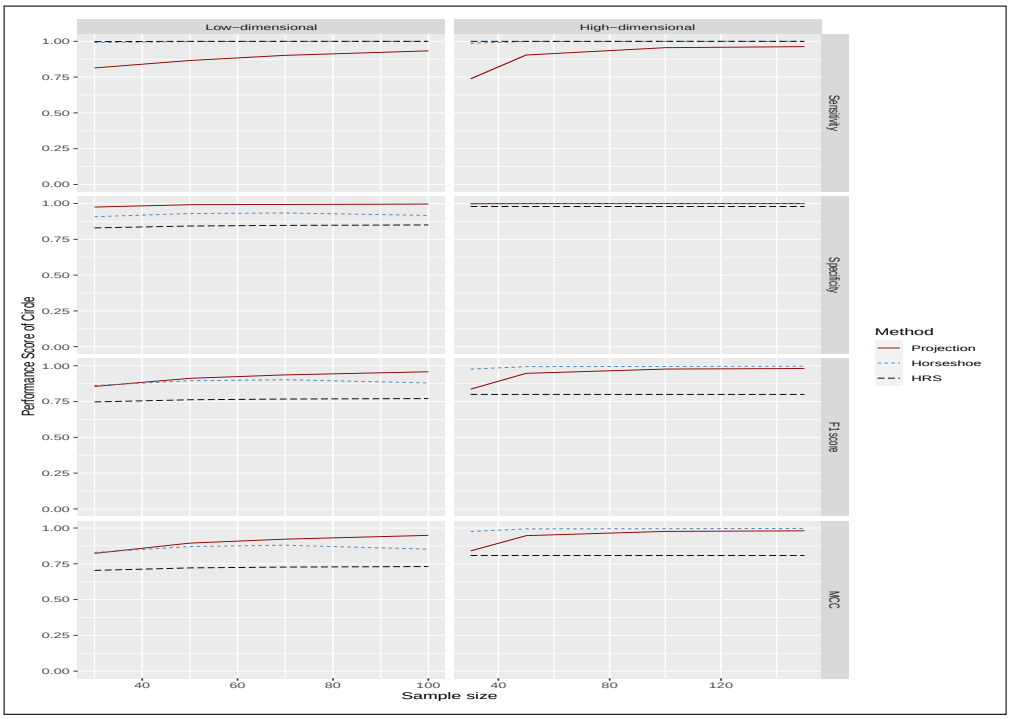


Figure A.2.(Continued)

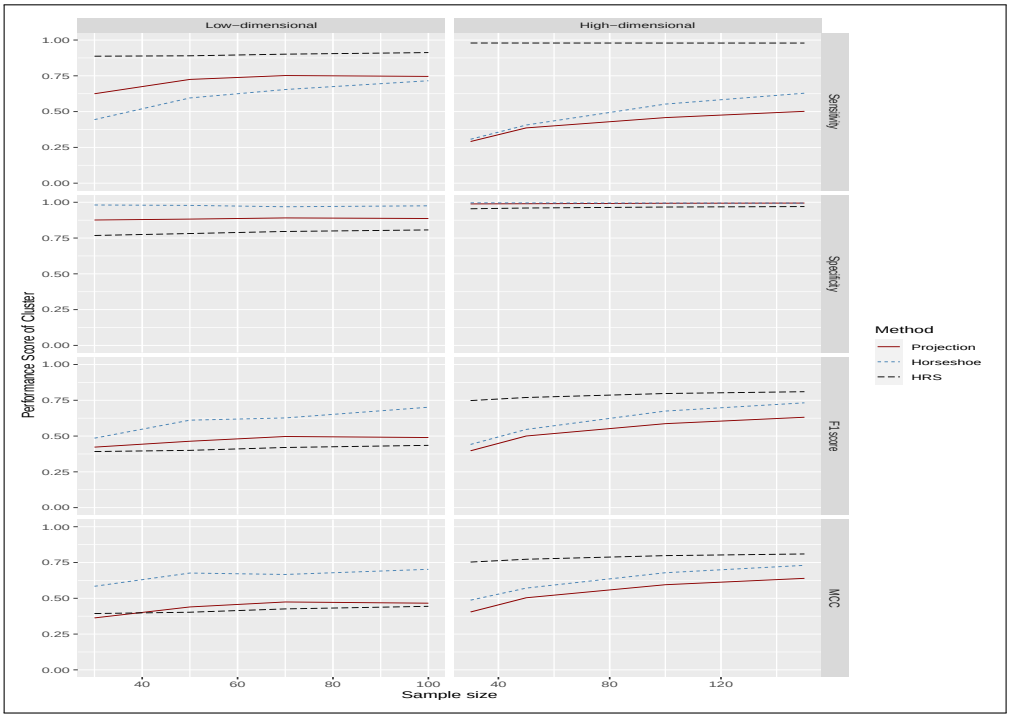


Figure A.2.(Continued)

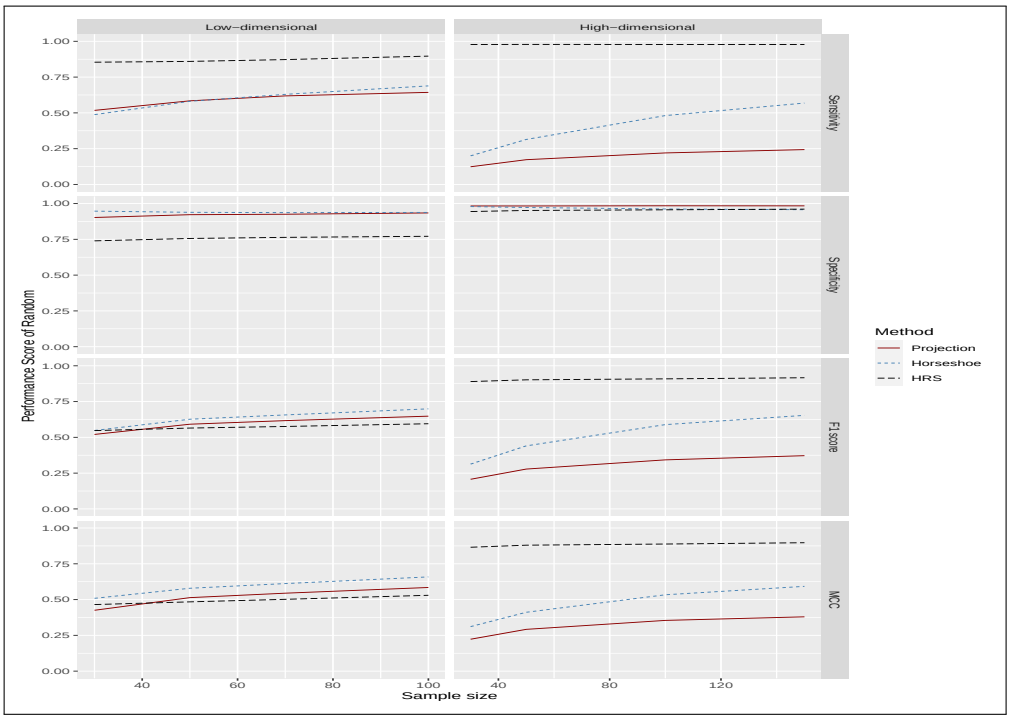


Figure A.2.(Continued)

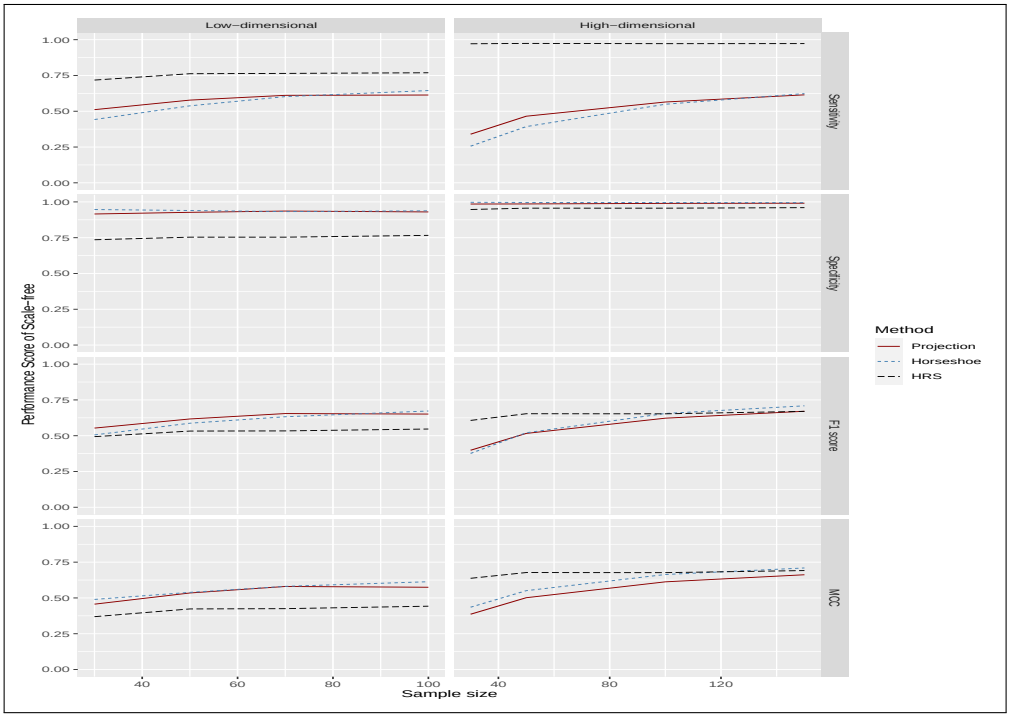


Figure A.2.(Continued)

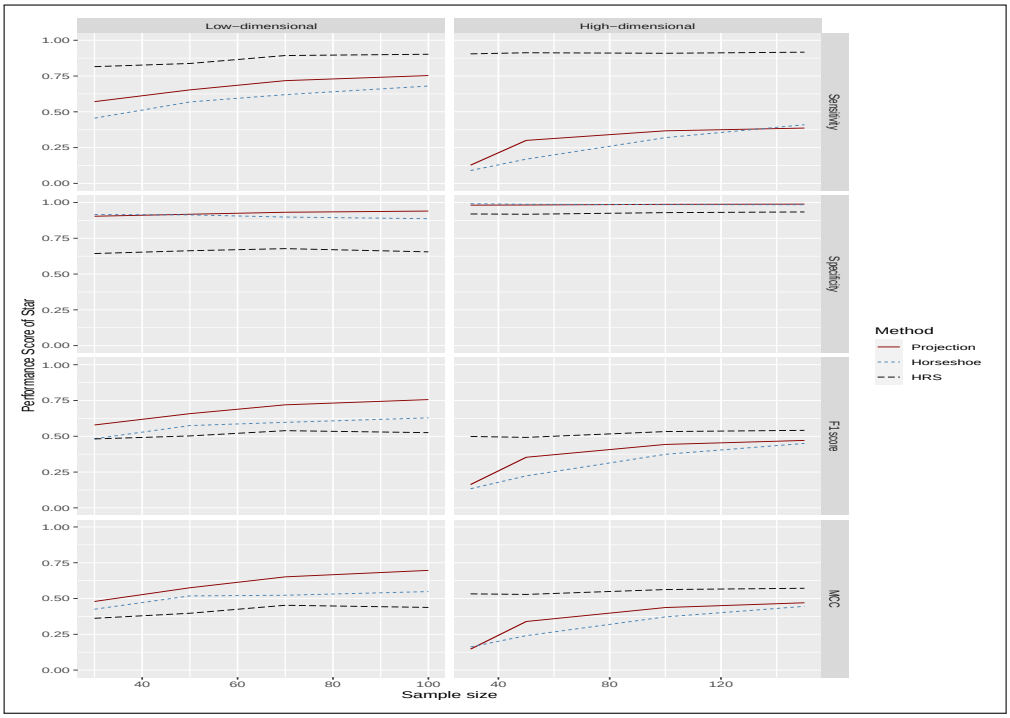


Figure A.2.(Continued)

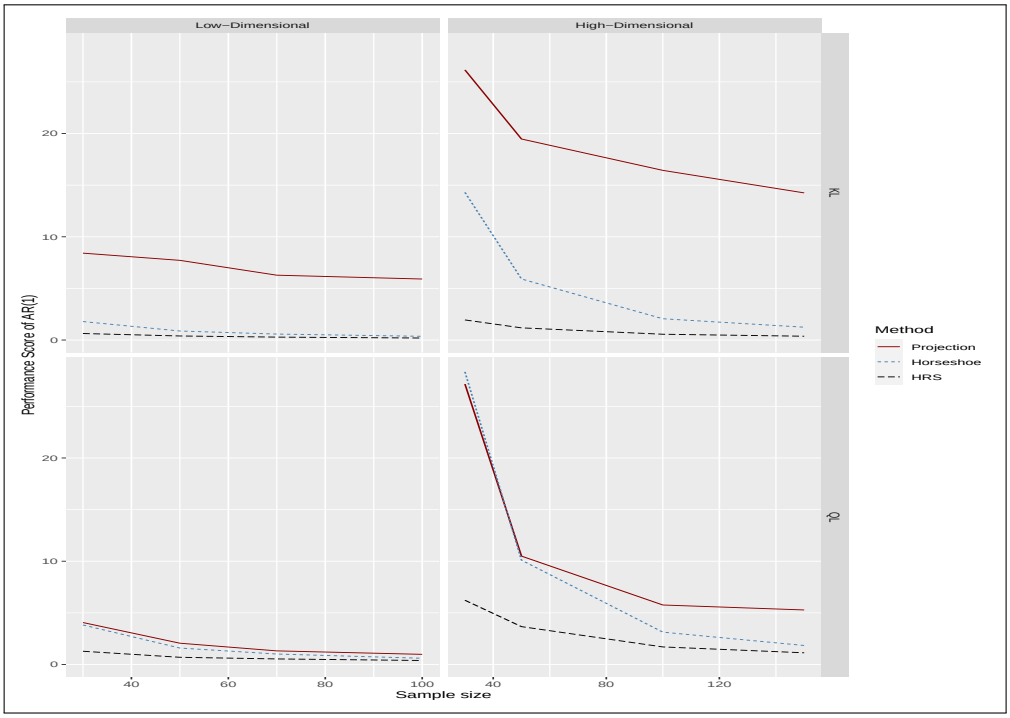


Figure A.3: Comparison of loss functions' estimation

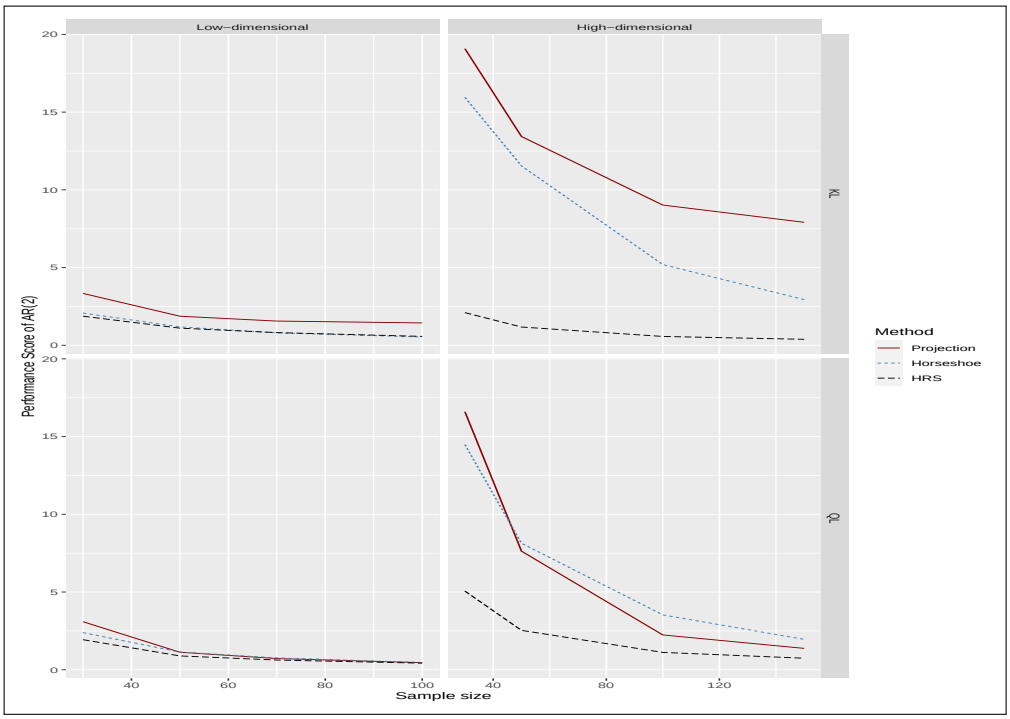


Figure A.3.(Continued)

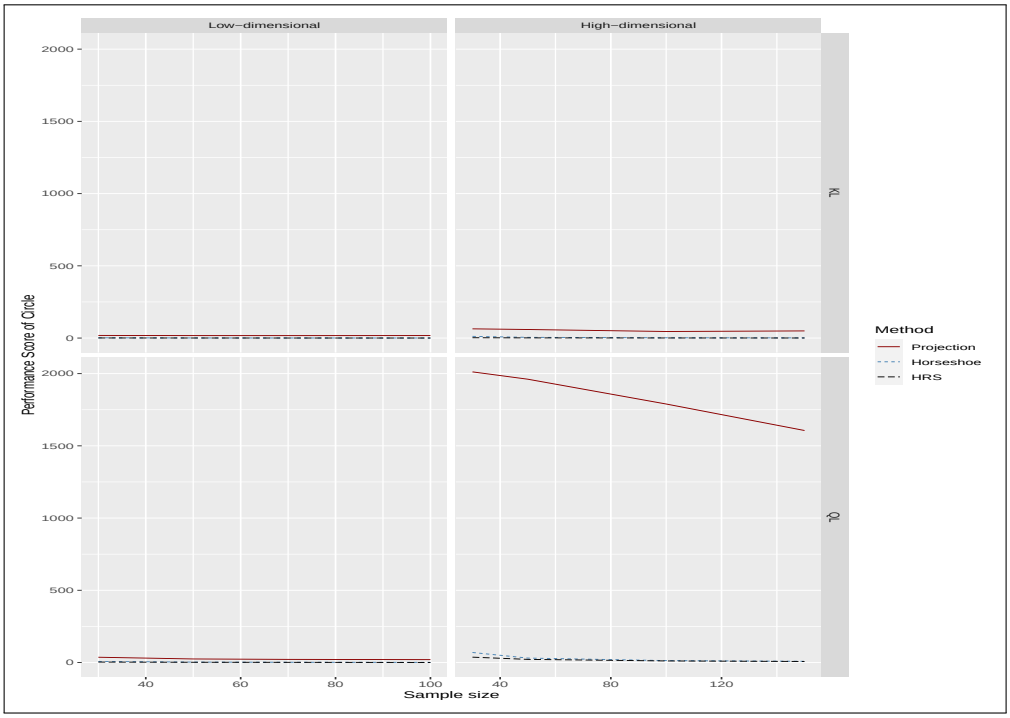


Figure A.3.(Continued)

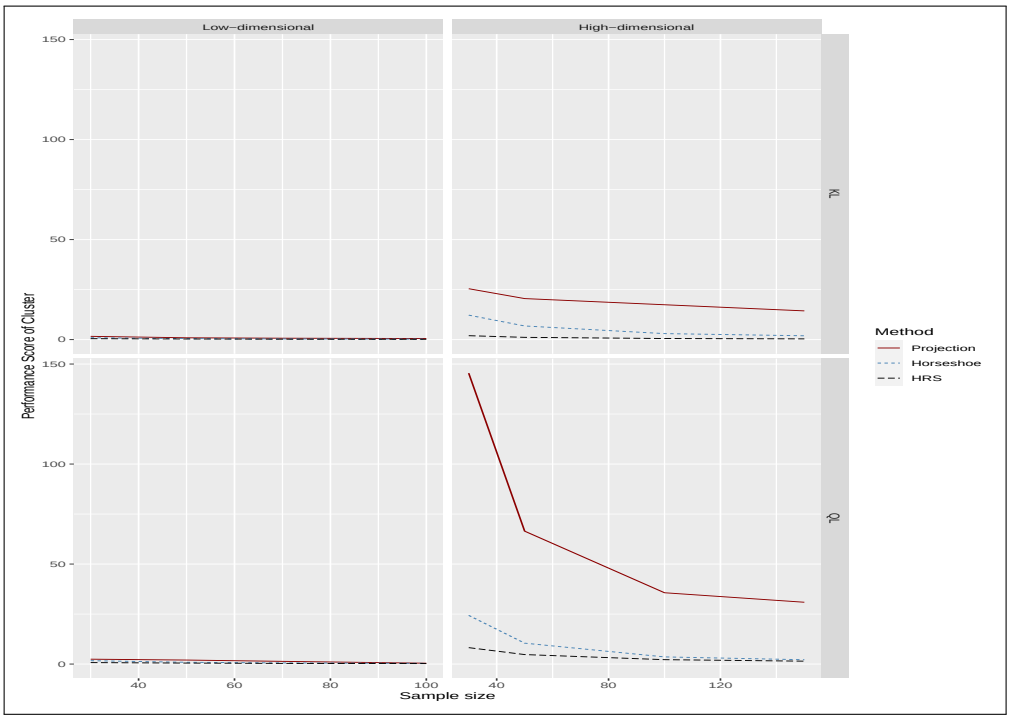


Figure A.3.(Continued)

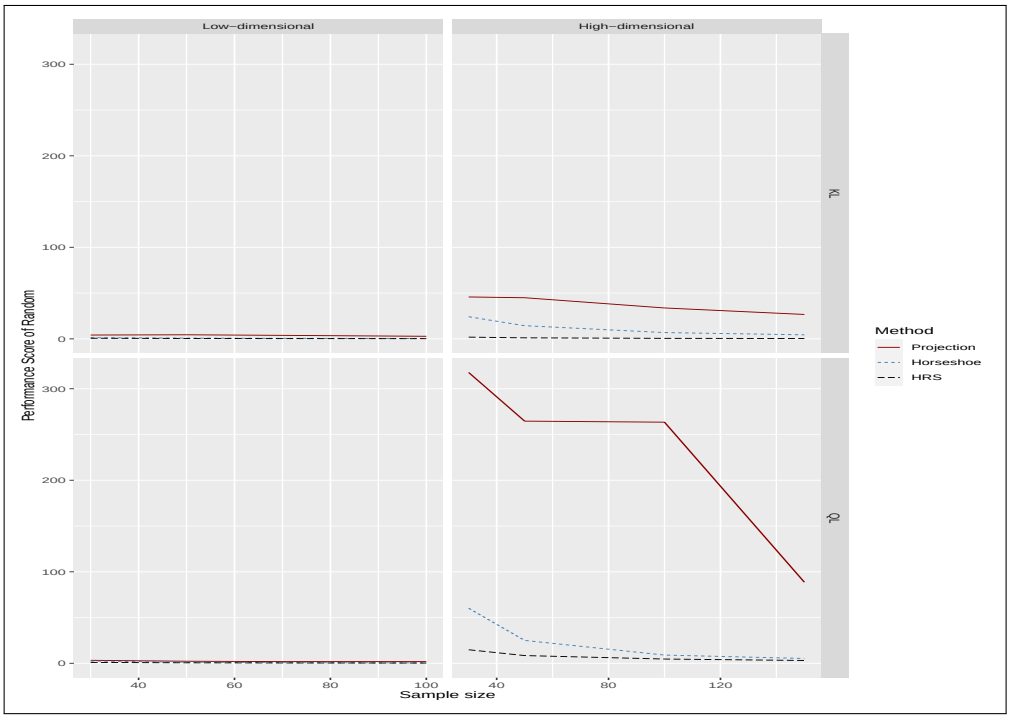


Figure A.3.(Continued)

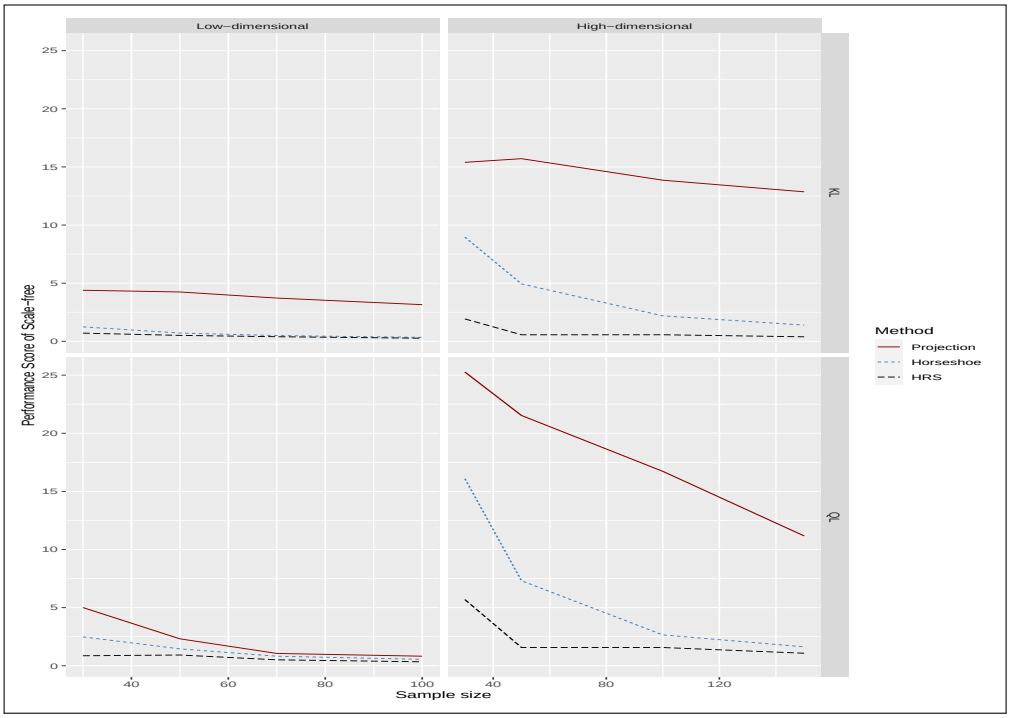


Figure A.3.(Continued)

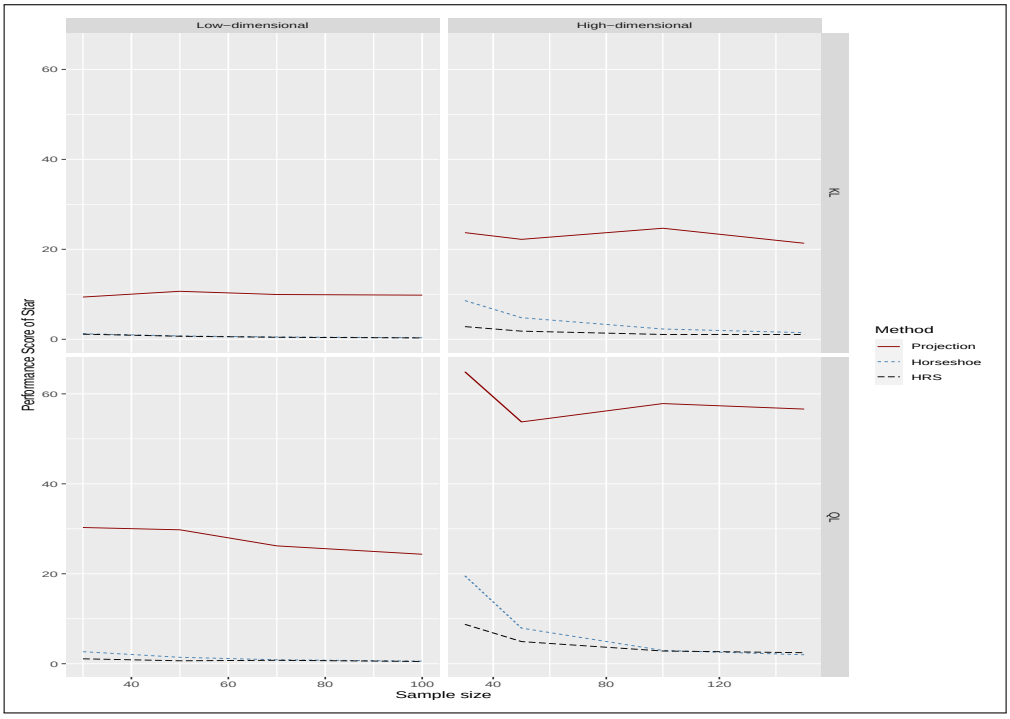


Figure A.3.(Continued)

Appendix B

Code

B.1 Clustering project

```
#########################################################
#################### Parameters ####################
# no.iter: number of iteration in MCMC
# no.site: number of CpG sites
# no.sub: number of sample size
# no.cluster: number of cluster
# sigma2: variance in prior distribution of gamma
# pi: parameter in prior distribution of delta
# delta: transimission status
# delta=1: transmit; delta=0: nontransmit
# mu: clustering
# gamma_k: transmission patterns
# gamma_0j: average DNA methylation of a child

#################### function ####################
log.child.trans.func=function(beta_child.test,no.site,all.
   alpha.child,child.data){
  return(sapply(1:no.site, function(j){
    sum(sapply(1:no.sub, function(i){
      (all.alpha.child[j]-1)*log(child.data[i,j])+(beta_child.
         test[j]-1)*log(1-child.data[i,j])-lbeta(all.alpha.
         child[j],beta_child.test[j])
    }))
```

```
  }))
}

log.moth.func=function(all.alpha.moth,all.beta.moth,no.site,
   moth.data){
  return(sapply(1:no.site, function(j){
    sum(sapply(1:no.sub, function(i){
      (all.alpha.moth[j]-1)*log(moth.data[i,j])+(all.beta.moth
         [j]-1)*log(1-moth.data[i,j])-lbeta(all.alpha.moth[j],
         all.beta.moth[j])
    }))
  })
  )
}

log.fath.func=function(all.alpha.fath,all.beta.fath,no.site,
   fath.data){
  return(
    sapply(1:no.site, function(j){
      sum(sapply(1:no.sub, function(i){
        (all.alpha.fath[j]-1)*log(fath.data[i,j])+(all.beta.
           fath[j]-1)*log(1-fath.data[i,j])-lbeta(all.alpha.
           fath[j],all.beta.fath[j])
      }))
    })
  )
}

##################### update gamma ####################
```

```
gamma.trans_update=function(gamma_k.test,gamma_0j.new,no.
   notrans,site.notrans,delta,no.site){
  result = t(sapply(1:no.site, function(j) {
    if(delta[j]==1) return(gamma_k.test)
    else return(rep(0,3))
  }))
  for (t in 1:no.notrans){
    if (!length(site.notrans)==0){
      result[site.notrans[t],]=gamma_0j.new[t,]
    }
  }
  return(result)
}

###### log of positerior distribution of gamma_0j ######
log.posterior_gamma_0j=function(gamma_0j,child.part,no.notrans
   ){
  return(sapply(1:no.notrans, function(t) {
    child.part[t]-t(gamma_0j[t,1])%*%gamma_0j[t,1]/(2*sigma2)
  }))
}

############### function of clustering ###############
gamma.clst_update=function(gamma_k.test,gamma_0j.new,no.
   notrans,site.notrans,mu.new){
  result = t(sapply(1:no.site, function(j) {
    for (m in 1:no.cluster){
      if (mu.new[j,m]==1) return(gamma_k.test[m,])
    }
```

```
  }))
  for (t in 1:no.notrans){
    if (!length(site.notrans)==0){
      result[site.notrans[t],]=gamma_0j.new[t,]
    }
  }
  return(result)
}

############### function for update beta ###############
beta.child_update=function(gamma_k,gamma,site.trans){
  return(sapply(1:no.cluster, function(k){
    sapply(1:no.site,function(j){
      ifelse(j %in% site.trans,(all.alpha.child[j]/exp(gamma_k
         [k,1]))*(all.beta.moth[j]/all.alpha.moth[j])^gamma_k[
         k,2]*(all.beta.fath[j]/all.alpha.fath[j])^gamma_k[k,3
         ],
             exp(log(all.alpha.child[j])-gamma[j,1]))
    })
  }))
}

log.child.func=function(beta_child.test){
  return(sapply(1:no.cluster, function(k){
    sapply(1:no.site, function(j){
      sum(sapply(1:no.sub, function(i){
        (all.alpha.child[j]-1)*log(child.data[i,j])+(beta_
           child.test[j,k]-1)*log(1-child.data[i,j])-lbeta(all
```

```
            .alpha.child[j],beta_child.test[j,k])
      }))
    })
  }))
}

############### parameters setting ###############
no.iter=20000
no.site=650
no.sub=100
############### update if more than 2 clusters #############
no.cluster=2
sigma2=100
p=numeric()
p=500/no.site
pi=matrix(rep(1/no.cluster,no.site*no.cluster),nrow=no.site,
   ncol=no.cluster)
threshold=10^(-7)

############### set up transmission patterns ###############
############### update if more than 2 clusters #############
gamma_k.true=matrix(0,nrow=no.cluster,ncol=3)
gamma_k.true[1,]=c(-4.2, 1.2, 2.5)
gamma_k.true[2,]=c(-2.7, 3, 2.5)

############# data simulation  #############
all.data=list()
max.data=150
delta_true=list()
```

```
distribute.site_true=list()
for (num.data in 1:max.data) {
  set.seed(seed[num.data])
  delta_true[[num.data]]=numeric()
  delta_true[[num.data]]=rbinom(no.site,1,p)
  site.order=seq(1:no.site)
  site.notrans_true=site.order[which(delta_true[[num.data]]==0
     )]
  site.trans_true=site.order[which(delta_true[[num.data]]==1)]
  no.notrans_true=length(site.notrans_true)
  cluster.size_true=floor(length(site.trans_true)/no.cluster)
     # site in cluster
  distribute.site_true.sub=c()
  for (k in 1:(no.cluster)){
    distribute.site_true.sub[[k]]=site.trans_true[((k-1)*
       cluster.size_true+1):(k*cluster.size_true)]
    distribute.site_true.sub[[no.cluster]]=site.trans_true[((
       no.cluster-1)*cluster.size_true+1):length(site.trans_
       true)]
  }
  distribute.site_true[[num.data]]=distribute.site_true.sub
###### update if there are more than 2 clusters ######
  coff.para=matrix(rep(0,(no.cluster+no.notrans_true)*3),nrow=
     no.cluster+no.notrans_true,ncol=3)
  coff.para[1,]=c(-4.2, 1.2, 2.5)
  coff.para[2,]=c(-2.7, 3, 2.5)
  for (i in 1:no.notrans_true){
    coff.para[i+no.cluster,1]=rnorm(1,7,0.5)
  }
```

```
  gamma_true=matrix(nrow=no.site,ncol=3)
  for (j in 1:no.site){
    for (m in 1:no.cluster){
      if (j %in% distribute.site_true[[num.data]][[m]]) gamma_
         true[j,]=coff.para[m,]
    }

    for (t in 1:no.notrans_true){
      if ( j==site.notrans_true[t]) gamma_true[j,]=coff.para[
         no.cluster+t,]
    }
  }

  mu_true=matrix(nrow=no.site,ncol=no.cluster)
  for (k in 1:no.cluster){
    for (j in 1:no.site){
      if (j %in% distribute.site_true[[num.data]][[k]]){
        mu_true[j,k]=1
      } else {
        mu_true[j,k]=0
      }
    }
  }

#######  each CpG site has its own parameters #######
  para.moth=matrix(nrow=no.site,ncol=2)
  para.fath=matrix(nrow=no.site,ncol=2)
  para.child=matrix(nrow=no.site,ncol=2)
```

```
######### generate parameters for each site #########
  for (j in 1:no.site){
    para.moth[j,1]=abs(rnorm(1,5,0.5))
    para.moth[j,2]=abs(rnorm(1,7,0.5))
    para.fath[j,1]=abs(rnorm(1,5,0.5))
    para.fath[j,2]=abs(rnorm(1,15,0.5))
    para.child[j,1]=abs(rnorm(1,25,0.5))
  }

  for (j in 1:no.site){
##### calculate beta_child based on the equation #####
    for (m in 1:no.cluster){
      if(j %in% distribute.site_true[[num.data]][[m]]) k=m
    }

    for (t in 1:no.notrans_true){
      if ( j==site.notrans_true[t]) k=no.cluster+t
    }
    para.child[j,2]=ifelse(j %in% site.trans_true,(para.child[
       j,1]/exp(coff.para[k,1]))*(para.moth[j,2]/para.moth[j,1
       ])^coff.para[k,2]*(para.fath[j,2]/para.fath[j,1])^coff.
       para[k,3],
                           exp(log(para.child[j,1])-coff.para[
                              k,1]))
  }

  moth.data=matrix(nrow=no.sub,ncol=no.site)
  fath.data=matrix(nrow=no.sub,ncol=no.site)
  child.data=matrix(nrow=no.sub,ncol=no.site)
```

```
  for (j in 1:no.site){
    moth.data[,j]=rbeta(no.sub,para.moth[j,1],para.moth[j,2])
    fath.data[,j]=rbeta(no.sub,para.fath[j,1],para.fath[j,2])
    child.data[,j]=rbeta(no.sub,para.child[j,1],para.child[j,2
       ])
    for(i in 1:no.sub){
      if(child.data[i,j]==1){
        child.data[i,j]=child.data[i,j]-threshold
      }
    }
  }
  Data=list()
  Data[[1]]=moth.data
  Data[[2]]=fath.data
  Data[[3]]=child.data
  all.data[[num.data]]=Data
  num.data=num.data+1
}

############## main code ##############
#### identify the transmission status  ####
all.conf_matrix.sub=list()
all.conf_matrix.mean=list()
all.prob.corr.trans.mean=c()
all.mu.sen.median=matrix(nrow=max.data,ncol=no.cluster)
all.mu.spec.median=matrix(nrow=max.data,ncol=no.cluster)

for (num.data in 1:max.data){
  moth.data=all.data[[num.data]][[1]]
```

```
  fath.data=all.data[[num.data]][[2]]
  child.data=all.data[[num.data]][[3]]

#### use sample mean and variance to estimate alpha, beta ####
  all.alpha.moth=sapply(1:no.site,function(j){
    if (var(moth.data[,j])==0){
      return( mean(moth.data[,j])*(mean(moth.data[,j])*(1-mean
         (moth.data[,j]))/(var(moth.data[,j])+threshold)-1))
    } else {
      return( mean(moth.data[,j])*(mean(moth.data[,j])*(1-mean
         (moth.data[,j]))/var(moth.data[,j])-1))
    }
  })

  all.beta.moth=sapply(1:no.site,function(j){
    all.alpha.moth[j]/(mean(moth.data[,j]))-all.alpha.moth[j]
  })
  all.alpha.fath=sapply(1:no.site,function(j){
    if (var(fath.data[,j])==0){
      return( mean(fath.data[,j])*(mean(fath.data[,j])*(1-mean
         (fath.data[,j]))/(var(fath.data[,j])+threshold)-1))
    } else {
      return(mean(fath.data[,j])*(mean(fath.data[,j])*(1-mean(
         fath.data[,j]))/var(fath.data[,j])-1))
    }
  })
  all.beta.fath=sapply(1:no.site,function(j){
    all.alpha.fath[j]/(mean(fath.data[,j]))-all.alpha.fath[j]
  })
```

```
  all.alpha.child=sapply(1:no.site,function(j){
    if (var(child.data[,j])==0){
      return(mean(child.data[,j])*(mean(child.data[,j])*(1-
         mean(child.data[,j]))/(var(child.data[,j])+threshold)
         -1))
    } else {
      return(mean(child.data[,j])*(mean(child.data[,j])*(1-
         mean(child.data[,j]))/(var(child.data[,j]))-1))
    }
  })
  all.beta.child=sapply(1:no.site,function(j){
    all.alpha.child[j]/(mean(child.data[,j]))-all.alpha.child[
       j]
  })

############### initial value  ###############
  set.seed(seed[num.data])
  site.order=seq(1:no.site)
  delta.init=rbinom(650,1,p)

  site.notrans.init=site.order[which(delta.init==0)]
  site.trans.init=site.order[which(delta.init==1)]
  no.notrans.init=length(site.notrans.init)

  gamma_k.trans.init=t(as.matrix(rnorm(3)))
  gamma_0j.trans.init=t(sapply(1:no.notrans.init,function(t){
    c(rnorm(1,7,2),0,0)
  }))
```

```
  gamma.trans.init=matrix(nrow=no.site,ncol=3)
  for (j in 1:no.site){
    if(delta.init[j]==1) gamma.trans.init[j,]=gamma_k.trans.
       init
    for (t in 1:no.notrans.init){
      if ( j==site.notrans.init[t]) gamma.trans.init[j,]=gamma
         _0j.trans.init[t,]
    }
  }

  all.delta.simple=matrix(nrow=no.iter,ncol=no.site)
  all.gamma.trans.simple=list()
  all.gamma_k.trans.simple=matrix(nrow=no.iter,ncol=3)

  all.delta.simple[1,]=delta.init
  all.gamma.trans.simple[[1]]=gamma.trans.init
  all.gamma_k.trans.simple[1,]=gamma_k.trans.init

  gamma_intercept.chain=matrix(rep(0,no.iter*no.site),no.iter,
     no.site)
  gamma_intercept.chain[1,]=gamma.trans.init[,1]

############### main estimate delta ###############
# step1: Use all the CpGs to estimate delta
# step2: select the nontransmitted CpGs from step1 and find
   the transmitted CpGs, this process will stop until there's
   no transmitted CpGs can be identified.
# step3: Combine all the transmitted CpGs in iteration.
```

```
  iter=0
  cond=0
  no.notrans.sub=list()
  site.notrans.sub=list()
  site.trans_cmb=list()
  gamma_0j.final=list()


  while (cond==0){
    iter=iter+1
################  Begin step1 ###############
    if (iter==1){
############ with all the CpGs #############
      for (N in 2:no.iter){
        delta=all.delta.simple[N-1,]
        gamma=all.gamma.trans.simple[[N-1]]
        gamma_k=t(as.matrix(all.gamma_k.trans.simple[N-1,]))

#### calculate beta_child based on equation  ####
        beta.child.trans = sapply(1:no.site, function(j){
          return((all.alpha.child[j]/exp(gamma_k[,1]))*(all.
             beta.moth[j]/all.alpha.moth[j])^gamma_k[,2]*(all.
             beta.fath[j]/all.alpha.fath[j])^gamma_k[,3])
        })

        child.part_trans=log.child.trans.func(beta.child.trans
           ,no.site,all.alpha.child,child.data)
        log.moth=log.moth.func(all.alpha.moth,all.beta.moth,no
           .site,moth.data)
```

```
        log.fath=log.fath.func(all.alpha.fath,all.beta.fath,no
           .site,fath.data)

############## Estimate delta #############
##### calculate posterior distribution of delta  #####
##### loga: assume CpGs are transmitted  #####
        loga = sapply (1:no.site, function(j){
          log.moth[j]+log.fath[j]+log(p)+(child.part_trans[j]-
             gamma_k%*%t(gamma_k)/(2*sigma2))
        })

##### logb: assume CpGs are nontransmitted  #####
        logb = sapply (1:no.site, function(j){
          if (delta[j]==1){
            gamma_est = gamma_k[,1]
            beta.child.notrans=exp(log(all.alpha.child[j])-
               gamma_est)
            return(sum(sapply(1:no.sub, function(i) {
              (all.alpha.child[j]-1)*log(child.data[i,j])+(
                 beta.child.notrans-1)*log(1-child.data[i,j])-
                 lbeta(all.alpha.child[j],beta.child.notrans)
            })) + log(1-p) - gamma_est*gamma_est/(2*sigma2))
          }
          else {
            beta.child.notrans=exp(log(all.alpha.child[j])-
               gamma[j,1])
            if (is.nan(beta.child.notrans)==T) {
              beta.child.notrans=mean(all.beta.moth[j], all.
                 beta.fath[j])
```

```
        }
        return(sum(sapply(1:no.sub, function(i) {
          (all.alpha.child[j]-1)*log(child.data[i,j])+(
             beta.child.notrans-1)*log(1-child.data[i,j])-
             lbeta(all.alpha.child[j],beta.child.notrans)
        })) + log(1-p) - gamma[j,1]*gamma[j,1] / (2*sigma2
           ))
      }
    })

    M = sapply(1:no.site, function(j) {max(loga[j],logb[j
       ])})
    p_post = sapply(1:no.site, function(j) {exp(loga[j]-M[
       j])/(exp(loga[j]-M[j])+exp(logb[j]-M[j]))})
    delta_new = sapply(1:no.site, function(j) {rbinom(1,1,
       p_post[j])})

##### updated transmitted CpGs and non-transmitted CpGs #####
    site.order=seq(1:no.site)
    site.notrans=site.order[which(delta_new==0)]
    site.trans=site.order[which(delta_new==1)]
    no.notrans=length(site.notrans)

    if (length(site.trans)<3) {
      delta_new1=all.delta.simple[N-1,]
    } else {
      delta_new1=delta_new
    }
```

```
        site.order=seq(1:no.site)
        site.notrans1=site.order[which(delta_new1==0)]
        site.trans1=site.order[which(delta_new1==1)]
        no.notrans1=length(site.notrans1)

############## update gamma_k by regression #############
        child.mean.trans=logit(colMeans(child.data[,site.trans
           1]))
        moth.mean.trans=logit(colMeans(moth.data[,site.trans1
           ]))
        fath.mean.trans=logit(colMeans(fath.data[,site.trans1
           ]))
        fit=lm(child.mean.trans~moth.mean.trans+fath.mean.
           trans)
        gamma_k.new=t(as.matrix(coefficients(fit)))
        colnames(gamma_k.new)=NULL

        gamma.process=matrix(rep(0,no.site*3),nrow=no.site,
           ncol=3)
        for (j in 1:no.site){
          if(delta_new1[j]==1) gamma.process[j,]=gamma_k.new
          else gamma.process[j,1]=gamma[j,1]
        }

##### update gamma_0j for non_transmitted CpGs
##### (metropolis hasting)
############## old gamma_0j #############
        beta_child_update.process=sapply(1:no.site,function(j)
           {
```

```
      ifelse(j %in% site.trans1,(all.alpha.child[j]/exp(
         gamma_k.new[,1]))*(all.beta.moth[j]/all.alpha.
         moth[j])^gamma_k.new[,2]*(all.beta.fath[j]/all.
         alpha.fath[j])^gamma_k.new[,3],
             exp(log(all.alpha.child[j])-gamma.process[j,1
                ]))
    })
    child.part_update.process=log.child.trans.func(beta_
       child_update.process,no.site,all.alpha.child,child.
       data)

    if (!no.notrans1==0){
      gamma_0j.old=matrix(rep(0,no.notrans1*3),nrow=no.
         notrans1,ncol=3)
    }else {
      gamma_0j.old=rep(0,3)
    }
    gamma_0j.old=as.matrix(gamma.process[site.notrans1,])
    child.part_0j.old = sapply(1:no.notrans1, function(t)
       {
      if( !length(site.notrans1)==0){
        child.part_update.process[site.notrans1[t]]
      } else {0}
    })

############## candidate gamma_0j #############
    if (!no.notrans1==0){
      gamma_0j.test=matrix(rep(0,no.notrans1*3),nrow=no.
         notrans1,ncol=3)
```

```
} else {
  gamma_0j.test=rep(0,3)
}
gamma_0j.test=t(sapply(1:no.notrans1,function(j){
  if (!length(site.notrans1)==0) {
    c(rnorm(1,gamma_0j.old[j,1],0.08),0,0)
  }
}))

gamma.test_0j=gamma.trans_update(gamma_k.new,gamma_0j.
   test,no.notrans1,site.notrans1,delta_new1,no.site)

beta_child.test_0j=sapply(1:no.site,function(j){
  ifelse(j %in% site.trans1,(all.alpha.child[j]/exp(
     gamma_k.new[,1]))*(all.beta.moth[j]/all.alpha.
     moth[j])^gamma_k.new[,2]*(all.beta.fath[j]/all.
     alpha.fath[j])^gamma_k.new[,3],
         exp(log(all.alpha.child[j])-gamma.test_0j[j,1
            ]))
})
child.part_update_0j= log.child.trans.func(beta_child.
   test_0j,no.site,all.alpha.child,child.data)

child.part_0j.test = sapply(1:no.notrans1, function(t)
    {
  if( !length(site.notrans1)==0){
    child.part_update_0j[site.notrans1[t]]
  } else {0}
})
```

```
############## updated gamma_0j ##############
        if (!no.notrans1==0){
          gamma_0j.new=matrix(nrow=no.notrans1,ncol=3)
        } else {
          gamma_0j.new=rep(0,3)
        }
        for (j in 1:no.notrans1){
          if (!length(site.notrans1)==0){
            u=runif(1)
            prob=exp(log.posterior_gamma_0j(gamma_0j.test,
               child.part_0j.test,no.notrans1)[j] - log.
               posterior_gamma_0j(gamma_0j.old,child.part_0j.
               old,no.notrans1)[j])
            if (u<=min(1,prob)){
              gamma_0j.new[j,]=gamma_0j.test[j,]
            } else {
              gamma_0j.new[j,]=gamma_0j.old[j,]
            }
          }
        }
############## update all the parameters #############
############## delta #############
        all.delta.simple[N,]=delta_new1
############## gamma #############
        gamma.new=gamma.trans_update(gamma_k.new,gamma_0j.new,
           no.notrans1,site.notrans1,delta_new1,no.site)
        all.gamma.trans.simple[[N]]=gamma.new
        all.gamma_k.trans.simple[N,]=gamma_k.new
        for (j in 1:no.site){
```

```
          gamma_intercept.chain[N,j]=gamma.new[j,1]
        }
        N=N+1
      } # end MCMC
##### use the last 1000 iterations to draw inferences
##### on the parameters
############## delta  ##############
      delta_burin=all.delta.simple[19000:20000,]

      delta_burin_mean=colMeans(delta_burin)
      delta_est=sapply(1:no.site,function(j){
        ifelse(delta_burin_mean[j]>0.5,1,0)
      })

#### transmitted and nontransmitted CpGs  ####
      site.notrans_burnin=site.order[which(delta_est==0)]
      site.trans_burnin=site.order[which(delta_est==1)]
      no.notrans_burnin=length(site.notrans_burnin)
      no.trans_burnin=length(site.trans_burnin)
      no.notrans.sub[[iter]]=no.notrans_burnin
      site.notrans.sub[[iter]]=site.notrans_burnin
      site.trans_cmb[[iter]]=site.trans_burnin

############## gamma_0j  ##############
      gamma_0j.burnin=gamma_intercept.chain[19000:20000,site.
         notrans_burnin]
      gamma_0j.final[[iter]]=gamma_0j.burnin
    } # end step1
```

```
############# begin step2 #############

    if (iter>1) {
      set.seed(seed[num.data])
## only select nontransmitted CpGs from previous iteration  ##
      delta.init_new=rbinom(no.notrans.sub[[iter-1]],1,p)
      site.order_new=seq(1:no.notrans.sub[[iter-1]])

############ initial values  ############
      site.notrans.init_new=site.order_new[which(delta.init_
         new==0)]
      site.trans.init_new=site.order_new[which(delta.init_new
         ==1)]
      no.notrans.init_new=length(site.notrans.init_new)
      no.trans.init_new=length(site.trans.init_new)

      gamma_k.trans.init_new=t(as.matrix(rnorm(3)))
      gamma_0j.trans.init_new=t(sapply(1:no.notrans.init_new,
         function(t){
        c(rnorm(1,7,2),0,0)
      }))

      gamma.trans.init_new=matrix(nrow=no.notrans.sub[[iter-1
         ]],ncol=3)
      for (j in 1:no.notrans.sub[[iter-1]]){
        if (delta.init_new[j]==1) {gamma.trans.init_new[j,]=
           gamma_k.trans.init_new
        } else {
          for (t in 1:no.notrans.init_new){
```

```
      if ( j==site.notrans.init_new[t]) gamma.trans.init
         _new[j,]=gamma_0j.trans.init_new[t,]
    }
  }
}

all.alpha.child_new=all.alpha.child[-site.trans_cmb[[
   iter-1]]]
all.beta.child_new=all.beta.child[-site.trans_cmb[[iter-
   1]]]
all.alpha.fath_new=all.alpha.fath[-site.trans_cmb[[iter-
   1]]]
all.beta.fath_new=all.beta.fath[-site.trans_cmb[[iter-1
   ]]]
all.alpha.moth_new=all.alpha.moth[-site.trans_cmb[[iter-
   1]]]
all.beta.moth_new=all.beta.moth[-site.trans_cmb[[iter-1
   ]]]

moth.data_new=as.matrix(moth.data[,-site.trans_cmb[[iter
   -1]]])
fath.data_new=as.matrix(fath.data[,-site.trans_cmb[[iter
   -1]]])
child.data_new=as.matrix(child.data[,-site.trans_cmb[[
   iter-1]]])

all.delta.simple_new=matrix(nrow=no.iter,ncol=no.notrans
   .sub[[iter-1]])
all.gamma.trans.simple_new=list()
```

```
      all.gamma_k.trans.simple_new=matrix(nrow=no.iter,ncol=3)

      all.delta.simple_new[1,]=delta.init_new
      all.gamma.trans.simple_new[[1]]=gamma.trans.init_new
      all.gamma_k.trans.simple_new[1,]=gamma_k.trans.init_new

      gamma_intercept.chain_new=matrix(rep(0,no.iter*no.
         notrans.sub[[iter-1]]),no.iter,no.notrans.sub[[iter-1
         ]])
      gamma_intercept.chain_new[1,]=gamma.trans.init_new[,1]

############## MCMC  #############
      for (N in 2:no.iter){
        delta=all.delta.simple_new[N-1,]
        gamma=all.gamma.trans.simple_new[[N-1]]
        gamma_k=t(as.matrix(all.gamma_k.trans.simple_new[N-1
           ,]))

        if (!all(is.na(gamma_k)==TRUE)) {

############## update beta_child  #############
          beta.child.trans = sapply(1:no.notrans.sub[[iter-1
             ]], function(j){
            return((all.alpha.child_new[j]/exp(gamma_k[,1]))*(
               all.beta.moth_new[j]/all.alpha.moth_new[j])^
               gamma_k[,2]*(all.beta.fath_new[j]/all.alpha.
               fath_new[j])^gamma_k[,3])
          })
```

```
            child.part_trans=log.child.trans.func(beta.child.
               trans,no.notrans.sub[[iter-1]],all.alpha.child_
               new,child.data_new)
            log.moth=log.moth.func(all.alpha.moth_new,all.beta.
               moth_new,no.notrans.sub[[iter-1]],moth.data_new)
            log.fath=log.fath.func(all.alpha.fath_new,all.beta.
               fath_new,no.notrans.sub[[iter-1]],fath.data_new)

############## Estimate delta   ##############
#### loga: assume CpGs are transmmitted  ####
            loga = sapply (1:no.notrans.sub[[iter-1]], function(
               j){
              log.moth[j]+log.fath[j]+log(p)+(child.part_trans[j
                 ]-gamma_k%*%t(gamma_k)/(2*sigma2))
            })

#### logb: assume CpGs are nontransmmitted  ####
            logb = sapply (1:no.notrans.sub[[iter-1]], function(
               j){
              if (delta[j]==1){
                gamma_est = gamma_k[,1]
                beta.child.notrans=exp(log(all.alpha.child_new[j
                   ])-gamma_est)
                return(sum(sapply(1:no.sub, function(i) {
                  (all.alpha.child_new[j]-1)*log(child.data_new[
                     i,j])+(beta.child.notrans-1)*log(1-child.
                     data_new[i,j])-lbeta(all.alpha.child_new[j
                     ],beta.child.notrans)
                })) + log(1-p) - gamma_est*gamma_est/(2*sigma2))
```

```
    }
    else {
      beta.child.notrans=exp(log(all.alpha.child_new[j
         ])-gamma[j,1])
      if (is.nan(beta.child.notrans)==T) {
        beta.child.notrans=mean(all.beta.moth_new[j],
           all.beta.fath_new[j])
      }
      return(sum(sapply(1:no.sub, function(i) {
        (all.alpha.child_new[j]-1)*log(child.data_new[
           i,j])+(beta.child.notrans-1)*log(1-child.
           data_new[i,j])-lbeta(all.alpha.child_new[j
           ],beta.child.notrans)
      })) + log(1-p) - gamma[j,1]*gamma[j,1] / (2*
         sigma2))
    }
  })

  M = sapply(1:no.notrans.sub[[iter-1]], function(j) {
     max(loga[j],logb[j])})
  p_post = sapply(1:no.notrans.sub[[iter-1]], function
     (j) {exp(loga[j]-M[j])/(exp(loga[j]-M[j])+exp(
     logb[j]-M[j]))})
  delta_new = sapply(1:no.notrans.sub[[iter-1]],
     function(j) {rbinom(1,1,p_post[j])})

  site.order=seq(1:no.notrans.sub[[iter-1]])

##### Update transmitted and nontransmitted CpGs #####
```

```
          if (all(delta_new==0)) {
###### if all CpGs are estimated as nontransmitted CpGs ######
####### only update gamma_0j ########
            site.notrans=site.order[which(delta_new==0)]
            no.notrans=length(site.notrans)

##### update gamma_0j for non_transmitted CpGs
##### (metropolis hasting)
############## old gamma_j #############
            beta_child_update.process=sapply(1:no.notrans.sub
               [[iter-1]],function(j){
              exp(log(all.alpha.child_new[j])-gamma[j,1])
            })
            child.part_update.process=log.child.trans.func(
               beta_child_update.process,no.notrans.sub[[iter-
               1]],all.alpha.child_new,child.data_new)

            gamma_0j.old=matrix(rep(0,no.notrans*3),nrow=no.
               notrans,ncol=3)
            gamma_0j.old[,1]=gamma[,1]
            child.part_0j.old=child.part_update.process

############ candidate gamma_0j ############
            gamma_0j.test=matrix(rep(0,no.notrans*3),nrow=no.
               notrans,ncol=3)
            gamma_0j.test=t(sapply(1:no.notrans,function(j){
              c(rnorm(1,gamma_0j.old[j,1],0.08),0,0)
            }))
```

```
            gamma.test=gamma_0j.test
            beta_child.test_0j=sapply(1:no.notrans.sub[[iter-1
               ]],function(j){
              exp(log(all.alpha.child_new[j])-gamma.test[j,1])
            })
            child.part_update_0j= log.child.trans.func(beta_
               child.test_0j,no.notrans.sub[[iter-1]],all.
               alpha.child_new,child.data_new)

            child.part_0j.test = child.part_update_0j

############ updated gamma_0j ############
            gamma_0j.new=matrix(nrow=no.notrans,ncol=3)

            for (j in 1:no.notrans){
              if (!length(site.notrans)==0){
                u=runif(1)
                prob=exp(log.posterior_gamma_0j(gamma_0j.test,
                   child.part_0j.test,no.notrans)[j] - log.
                   posterior_gamma_0j(gamma_0j.old,child.part_
                   0j.old,no.notrans)[j])
                if (u<=min(1,prob)){
                  gamma_0j.new[j,]=gamma_0j.test[j,]
                } else {
                  gamma_0j.new[j,]=gamma_0j.old[j,]
                }
              }
            }
############ update all the parameters ############
```

```
            all.delta.simple_new[N,]=delta_new
            all.gamma.trans.simple_new[[N]]=gamma_0j.new
            all.gamma_k.trans.simple_new[N,]=NA
            for (j in 1:no.notrans.sub[[iter-1]]){
              gamma_intercept.chain_new[N,j]=gamma_0j.new[j,1]
            }

          } else if(all(delta_new==1)){
##### if all CpGs are estimated as transmitted CpGs #####
############## only update gamma_k #############
            site.order=seq(1:no.notrans.sub[[iter-1]])
            site.trans=site.order[which(delta_new==1)]

#### if the number of transmitted CpGs smaller than 3,
#### use the previous step's parameter in MCMC
            if (length(site.trans)<3) {
              all.delta.simple_new[N,]=all.delta.simple_new[N-
                 1,]
              all.gamma.trans.simple_new[[N]]=all.gamma.trans.
                 simple_new[[N-1]]
              all.gamma_k.trans.simple_new[N,]=all.gamma_k.
                 trans.simple_new[N-1,]
              for (j in 1:no.notrans.sub[[iter-1]]){
                gamma_intercept.chain_new[N,j]=gamma_intercept
                   .chain_new[N-1,j]
              }

            } else {
              delta_new1=delta_new
```

```
                site.order=seq(1:no.notrans.sub[[iter-1]])
                site.trans1=site.order[which(delta_new1==1)]

############## update gamma_k by regression #############
                child.mean.trans=logit(colMeans(child.data_new[,
                   site.trans1]))
                moth.mean.trans=logit(colMeans(moth.data_new[,
                   site.trans1]))
                fath.mean.trans=logit(colMeans(fath.data_new[,
                   site.trans1]))
                fit=lm(child.mean.trans~moth.mean.trans+fath.
                   mean.trans)
                gamma_k.new=t(as.matrix(coefficients(fit)))
                colnames(gamma_k.new)=NULL

                gamma.all_trans=matrix(nrow=no.notrans.sub[[iter
                   -1]],ncol=3)
                for (j in 1:no.notrans.sub[[iter-1]]){
                  if (delta_new1[j]==1) {gamma.all_trans[j,]=
                     gamma_k.new
                  }
                }
                all.delta.simple_new[N,]=delta_new1
                all.gamma.trans.simple_new[[N]]=gamma.all_trans
                all.gamma_k.trans.simple_new[N,]=gamma_k.new

                for (j in 1:no.notrans.sub[[iter-1]]){
                  gamma_intercept.chain_new[N,j]=gamma_k.new[,1]
                }
```

```
            }

          } else {
#### updated transmitted CpGs and non-transmitted CpGs ####
            site.order=seq(1:no.notrans.sub[[iter-1]])
            site.notrans=site.order[which(delta_new==0)]
            site.trans=site.order[which(delta_new==1)]
            no.notrans=length(site.notrans)

            if (length(site.trans)<3) {
              delta_new1=all.delta.simple_new[N-1,]
            } else {
              delta_new1=delta_new
            }

            site.order=seq(1:no.notrans.sub[[iter-1]])
            site.notrans1=site.order[which(delta_new1==0)]
            site.trans1=site.order[which(delta_new1==1)]
            no.notrans1=length(site.notrans1)

############## update gamma_k by regression #############
            child.mean.trans=logit(colMeans(child.data_new[,
               site.trans1]))
            moth.mean.trans=logit(colMeans(moth.data_new[,site
               .trans1]))
            fath.mean.trans=logit(colMeans(fath.data_new[,site
               .trans1]))
```

```
            fit=lm(child.mean.trans~moth.mean.trans+fath.mean.
               trans)
            gamma_k.new=t(as.matrix(coefficients(fit)))
            colnames(gamma_k.new)=NULL

            gamma.process=matrix(rep(0,no.notrans.sub[[iter-1
               ]]*3),nrow=no.notrans.sub[[iter-1]],ncol=3)
            for (j in 1:no.notrans.sub[[iter-1]]){
              if(delta_new1[j]==1) gamma.process[j,]=gamma_k.
                 new
              else gamma.process[j,1]=gamma[j,1]
            }

##### update gamma_0j for non_transmitted CpGs
##### (metropolis hasting)
            beta_child_update.process=sapply(1:no.notrans.sub
               [[iter-1]],function(j){
              ifelse(j %in% site.trans1,(all.alpha.child_new[j
                 ]/exp(gamma_k.new[,1]))*(all.beta.moth_new[j
                 ]/all.alpha.moth_new[j])^gamma_k.new[,2]*(all
                 .beta.fath_new[j]/all.alpha.fath_new[j])^
                 gamma_k.new[,3],
                     exp(log(all.alpha.child_new[j])-gamma.
                        process[j,1]))
            })
            child.part_update.process=log.child.trans.func(
               beta_child_update.process,no.notrans.sub[[iter-
               1]],all.alpha.child_new,child.data_new)
############## old gamma_0j #############
```

```
            if (!no.notrans1==0){
              gamma_0j.old=matrix(rep(0,no.notrans1*3),nrow=no
                 .notrans1,ncol=3)
            }else {
              gamma_0j.old=rep(0,3)
            }
            gamma_0j.old=as.matrix(gamma.process[site.notrans1
               ,])
            child.part_0j.old = sapply(1:no.notrans1, function
               (t) {
              if( !length(site.notrans1)==0){
                child.part_update.process[site.notrans1[t]]
              } else {0}
            })

############ candidate gamma_0j #############
            if (!no.notrans1==0){
              gamma_0j.test=matrix(rep(0,no.notrans1*3),nrow=
                 no.notrans1,ncol=3)
            } else {
              gamma_0j.test=rep(0,3)
            }
            gamma_0j.test=t(sapply(1:no.notrans1,function(j){
              if (!length(site.notrans1)==0) {
                c(rnorm(1,gamma_0j.old[j,1],0.08),0,0)
              }
            }))
```

```
            gamma.test_0j=gamma.trans_update(gamma_k.new,gamma
               _0j.test,no.notrans1,site.notrans1,delta_new1,
               no.notrans.sub[[iter-1]])
            beta_child.test_0j=sapply(1:no.notrans.sub[[iter-1
               ]],function(j){
              ifelse(j %in% site.trans1,(all.alpha.child_new[j
                 ]/exp(gamma_k.new[,1]))*(all.beta.moth_new[j
                 ]/all.alpha.moth_new[j])^gamma_k.new[,2]*(all
                 .beta.fath_new[j]/all.alpha.fath_new[j])^
                 gamma_k.new[,3],
                     exp(log(all.alpha.child_new[j])-gamma.
                        test_0j[j,1]))
            })
            child.part_update_0j= log.child.trans.func(beta_
               child.test_0j,no.notrans.sub[[iter-1]],all.
               alpha.child_new,child.data_new)

            child.part_0j.test = sapply(1:no.notrans1,
               function(t) {
              if( !length(site.notrans1)==0){
                child.part_update_0j[site.notrans1[t]]
              } else {0}
            })
############## update gamma_0j #############
            if (!no.notrans1==0){
              gamma_0j.new=matrix(nrow=no.notrans1,ncol=3)
            } else {
              gamma_0j.new=rep(0,3)
            }
```

```
            for (j in 1:no.notrans1){
              if (!length(site.notrans1)==0){
                u=runif(1)
                prob=exp(log.posterior_gamma_0j(gamma_0j.test,
                   child.part_0j.test,no.notrans1)[j] - log.
                   posterior_gamma_0j(gamma_0j.old,child.part_
                   0j.old,no.notrans1)[j])
                if (u<=min(1,prob)){
                  gamma_0j.new[j,]=gamma_0j.test[j,]
                } else {
                  gamma_0j.new[j,]=gamma_0j.old[j,]
                }
              }
            }

############# update all the parameters in MCMC #############
############# update delta #############
            all.delta.simple_new[N,]=delta_new1
############# update gamma #############
            gamma.new=gamma.trans_update(gamma_k.new,gamma_0j.
               new,no.notrans1,site.notrans1,delta_new1,no.
               notrans.sub[[iter-1]])
            all.gamma.trans.simple_new[[N]]=gamma.new
            all.gamma_k.trans.simple_new[N,]=gamma_k.new

            for (j in 1:no.notrans.sub[[iter-1]]){
              gamma_intercept.chain_new[N,j]=gamma.new[j,1]
            }
```

```
          }
        } # end when gamma_k exist
        else {
##### if all the CpGs in previous step are nontransmitted CpGs
##### only update gamma_0j
############# update gamma_0j #############
############# old gamma_0j #############
          beta_child_update.old=sapply(1:no.notrans.sub[[iter-
             1]],function(j){
            exp(log(all.alpha.child_new[j])-gamma[j,1])
          })
          child.part_update.old=log.child.trans.func(beta_
             child_update.old,no.notrans.sub[[iter-1]],all.
             alpha.child_new,child.data_new)

############# candidate gamma_0j #############
          gamma_0j_update.test=t(sapply(1:no.notrans.sub[[iter
             -1]],function(j){
            c(rnorm(1,gamma_0j.old[j,1],0.08),0,0)
          }))

          beta_child_update.test=sapply(1:no.notrans.sub[[iter
             -1]],function(j){
            exp(log(all.alpha.child_new[j])-gamma_0j_update.
               test[j,1])
          })

          child.part_update.test= log.child.trans.func(beta_
             child_update.test,no.notrans.sub[[iter-1]],all.
```

```
            alpha.child_new,child.data_new)

############## updated gamma_0j #############
          gamma_0j_update.new=matrix(nrow=no.notrans.sub[[iter
             -1]],ncol=3)
          for (j in 1:no.notrans.sub[[iter-1]]){
            u=runif(1)
            prob=exp(log.posterior_gamma_0j(gamma_0j_update.
               test,child.part_update.test,no.notrans.sub[[
               iter-1]])[j] - log.posterior_gamma_0j(gamma,
               child.part_update.old,no.notrans.sub[[iter-1]])
               [j])
            if (u<=min(1,prob)){
              gamma_0j_update.new[j,]=gamma_0j_update.test[j,]
            } else {
              gamma_0j_update.new[j,]=gamma[j,]
            }
          }
          all.delta.simple_new[N,]=0
          all.gamma.trans.simple_new[[N]]=gamma_0j_update.new
          all.gamma_k.trans.simple_new[N,]=NA

          for (j in 1:no.notrans.sub[[iter-1]]){
            gamma_intercept.chain_new[N,j]=gamma_0j_update.new
               [j,1]
          }

        } # end when gamma_k do not exist
        N=N+1
```

```
      }  # end MCMC
##### use the last 1000 iterations to draw inferences
##### on the parameters
      delta_burin_new=all.delta.simple_new[19000:20000,]
      if (no.notrans.sub[[iter-1]]==1) {
        delta_burin_mean_new=mean(delta_burin_new)
      } else {
        delta_burin_mean_new=colMeans(delta_burin_new)
      }
      delta_est_new=sapply(1:no.notrans.sub[[iter-1]],function
         (j){
        ifelse(delta_burin_mean_new[j]>0.5,1,0)
      })

      site.order_new=seq(1:no.notrans.sub[[iter-1]])

      if (all(delta_est_new==0)){
##### if all the CpGs are nontransmiteed CpGs,
##### then stop the iteration
        site.notrans_burnin_new=site.order_new[which(delta_est
           _new==0)]
        no.notrans_burnin_new=length(site.notrans_burnin_new)

############## gamma_0j #############
        gamma_0j.burnin_new=gamma_intercept.chain_new[19000:20
           000,site.notrans_burnin_new]
        gamma_0j.final[[iter]]=gamma_0j.burnin_new

############## update the parameters  #############
```

```
        no.notrans.sub[[iter]]=no.notrans_burnin_new
        site.notrans.sub[[iter]]=site.notrans.sub[[iter-1]]
        site.trans_cmb[[iter]]=site.trans_cmb[[iter-1]]
        cond=1
      } else {
        site.notrans_burnin_new=site.order_new[which(delta_est
           _new==0)]
        site.trans_burnin_new=site.order_new[which(delta_est_
           new==1)]
        no.notrans_burnin_new=length(site.notrans_burnin_new)
        no.trans_burnin_new=length(site.trans_burnin_new)

##### if the number of nontransmitted CpGs is small,
##### then stop the iteration
        if (no.notrans_burnin_new<=20) {
          no.notrans.sub[[iter]]=no.notrans.sub[[iter-1]]
          site.notrans.sub[[iter]]=site.notrans.sub[[iter-1]]
          site.trans_cmb[[iter]]=site.trans_cmb[[iter-1]]
          gamma_0j.final[[iter]]=gamma_0j.final[[iter-1]]
          cond=1
        } else {
##### update the parameter in current iteration
##### and continuous for the next iteration
############## gamma_0j #############
          gamma_0j.burnin_new=gamma_intercept.chain_new[19000:
             20000,site.notrans_burnin_new]
          gamma_0j.final[[iter]]=gamma_0j.burnin_new

############## summary  #############
```

```
          no.notrans.sub[[iter]]=no.notrans_burnin_new
          site.notrans.sub[[iter]]=site.notrans.sub[[iter-1
             ]][-site.trans_burnin_new]
#### find the site of trans in original order ####
          site.trans_new=site.notrans.sub[[iter-1]][site.trans
             _burnin_new]
          site.trans_cmb[[iter]]=c(site.trans_cmb[[iter-1]],
             site.trans_new)
        }
      }
    }
  } # end iteration (no more transmitted CpGs can be detected)

#### final estimate of transimission status ####

  delta_est.final=rep(0,no.site)
  delta_est.final=sapply(1:no.site,function(j){
    ifelse(j %in% site.trans_cmb[[length(site.trans_cmb)]],1,0
       )
  })

############# final estimate of gamma_0j #############
  gamma_0j.mean=matrix(rep(0,no.notrans.sub[[length(no.notrans
     .sub)]]*3),nrow=no.notrans.sub[[length(no.notrans.sub)]])

  if (no.notrans.sub[[length(no.notrans.sub)]]==1) {
    gamma_0j.mean[,1]=mean(gamma_0j.final[[length(gamma_0j.
       final)]])
  } else {
```

```
    gamma_0j.mean[,1]=colMeans(gamma_0j.final[[length(gamma_0j
        .final)]])
  }

############## confusion matrix of delta #############
  conf_matrix=table(delta_est.final,delta_true[[num.data]])

#### clustering for identified transmitted CpG sites ####
############## initial value  #############
  set.seed(seed[num.data])
  site.order=seq(1:no.site)

##### select the identified transmitted CpG sites #####
  site.notrans.clst=site.order[which(delta_est.final==0)]
  site.trans.clst=site.order[which(delta_est.final==1)]
  no.notrans.clst=length(site.notrans.clst)

############## initial value for clustering #############
  mu.init_test=t(sapply(1:no.site,function(j){
    if (j %in% site.trans.clst) sample(c(rep(0,no.cluster-1),1
        ),no.cluster,replace=FALSE)
    else c(rep(0,no.cluster))
  }))

  distribute.site.init.clst=lapply(1:no.cluster,function(k){
    which(mu.init_test[,k]==1)
  })

##### each cluster should have more than 3 CpGs #####
```

```
  mu.init.clst=matrix(nrow=no.site,ncol=no.cluster)
  if (any(sapply(distribute.site.init.clst,length)<3)) {
    mu.init.clst=t(sapply(1:no.site,function(j){
      if (j %in% site.trans.clst) sample(c(rep(0,no.cluster-1)
        ,1),no.cluster,replace=FALSE)
      else c(rep(0,no.cluster))
    }))
  } else {
    mu.init.clst=mu.init_test
  }

  distribute.site.init.clst1=lapply(1:no.cluster,function(k){
    which(mu.init.clst[,k]==1)
  })

##### initial gamma_k (estimated by regression) #####
  child.mean.init.clst=list()
  moth.mean.init.clst=list()
  fath.mean.init.clst=list()
  gamma_k.init.clst=t(sapply(1:no.cluster,function(k){
    child.mean.init.clst[[k]]=logit(colMeans(child.data[,
      distribute.site.init.clst1[[k]]]))
    moth.mean.init.clst[[k]]=logit(colMeans(moth.data[,
      distribute.site.init.clst1[[k]]]))
    fath.mean.init.clst[[k]]=logit(colMeans(fath.data[,
      distribute.site.init.clst1[[k]]]))
    fit=lm(child.mean.init.clst[[k]]~moth.mean.init.clst[[k]]+
      fath.mean.init.clst[[k]])
    #chain_gamma_k[[k]][N,]=coefficients(fit)
```

```
    coefficients(fit)
  })
  )
  colnames(gamma_k.init.clst)=NULL

  gamma.init.clst=matrix(rep(0,no.site*3),nrow=no.site,ncol=3)
  gamma.init.clst=gamma.clst_update(gamma_k.init.clst,gamma_0j
     .mean,no.notrans.clst,site.notrans.clst,mu.init.clst)

  chain_gamma_k.clst=list()
  for (k in 1:no.cluster){
    chain_gamma_k.clst[[k]]=array(data=NA,c(no.iter,3))
    chain_gamma_k.clst[[k]][1,]=c(t(gamma_k.init.clst[k,]))
  }

  all.mu.simple.clst=list()
  all.gamma_k.simple.clst=list()
  all.gamma.simple.clst=list()
  all.distribute.site.clst=list()
  DS=rep(0,no.iter)

  all.mu.simple.clst[[1]]=mu.init.clst
  all.distribute.site.clst[[1]]=distribute.site.init.clst1
  all.gamma_k.simple.clst[[1]]=gamma_k.init.clst
  all.gamma.simple.clst[[1]]=gamma.init.clst

############## MCMC for clustering #############
  for (N in 2:no.iter){
    delta=delta_est.final
```

```
    mu=all.mu.simple.clst[[N-1]]
    gamma=all.gamma.simple.clst[[N-1]]
    gamma_k=all.gamma_k.simple.clst[[N-1]]

    log.moth=log.moth.func(all.alpha.moth,all.beta.moth,no.
       site,moth.data)
    log.fath=log.fath.func(all.alpha.fath,all.beta.fath,no.
       site,fath.data)
    beta_child_update=beta.child_update(gamma_k,gamma,site.
       trans.clst)
    child.part_update=log.child.func(beta_child_update)
    log_mu_post = matrix(rep(log.moth,no.cluster),no.site,no.
       cluster) + matrix(rep(log.fath,no.cluster),no.site,no.
       cluster) + child.part_update - sapply(1:no.cluster,
       function(k){
      sapply(1:no.site, function(j){
        (t(gamma_k[k,])%*%gamma_k[k,]/(2*sigma2)-log(pi[j,k]))
      })
    })

############# update mu #############
    mu_post=sapply(1:no.cluster,function(k){
      sapply(1:no.site,function(j){
        M=max(log_mu_post[j,])
        return(exp(log_mu_post[j,k]-M)/sum(exp(log_mu_post[j
           ,]-M)))
      })
    })
```

```
    mu.new = t(sapply(1:no.site, function(j) {
      if (j %in% site.trans.clst) {
        c(rmultinom(1,1,prob=mu_post[j,]))
      }
      else {rep(0,no.cluster)}
    }))

############## find the CpGs in each cluster #############
    distribute.site=lapply(1:no.cluster,function(k){
      which(mu.new[,k]==1)
    })

    mu.new1=matrix(nrow=no.site,ncol=no.cluster)
    if (any(sapply(distribute.site,length)<3)) {
      mu.new1=all.mu.simple.clst[[N-1]]
    } else {
      mu.new1=mu.new
    }

    distribute.site1=lapply(1:no.cluster,function(k){
      which(mu.new1[,k]==1)
    })

############## update gamma_k by regression #############
    child.mean.trans=list()
    moth.mean.trans=list()
    fath.mean.trans=list()
    gamma_k.new=t(sapply(1:no.cluster,function(k){
```

```
    child.mean.trans[[k]]=logit(colMeans(child.data[,
       distribute.site1[[k]]]))
    moth.mean.trans[[k]]=logit(colMeans(moth.data[,
       distribute.site1[[k]]]))
    fath.mean.trans[[k]]=logit(colMeans(fath.data[,
       distribute.site1[[k]]]))
    fit=lm(child.mean.trans[[k]]~moth.mean.trans[[k]]+fath.
       mean.trans[[k]])
    #chain_gamma_k[[k]][N,]=coefficients(fit)
    coefficients(fit)
  }))
  colnames(gamma_k.new)=NULL
  for (k in 1:no.cluster){
    chain_gamma_k.clst[[k]][N,]=gamma_k.new[k,]
  }


  all.mu.simple.clst[[N]]=mu.new1
  all.distribute.site.clst[[N]]=distribute.site1

  gamma.new=matrix(rep(0,no.site*3),nrow=no.site,ncol=3)
  gamma.new=gamma.clst_update(gamma_k.new,gamma_0j.mean,no.
     notrans.clst,site.notrans.clst,mu.new1)
  all.gamma.simple.clst[[N]]=gamma.new
  all.gamma_k.simple.clst[[N]]=gamma_k.new

  N=N+1
} # end of clustering
```

```
############## estimated mu after burnin #############
  all.mu.burn=all.mu.simple.clst[19000:20000]
  all.distribute.site.burn=all.distribute.site.clst[19000:2000
     0]
  all.mu_mean<-Reduce("+", all.mu.burn) / length(all.mu.burn)
  all.mu_est=t(sapply(1:no.site,function(j){
    sapply(1:no.cluster,function(k){
      return(ifelse(all.mu_mean[j,k]>0.5,1,0))
    })
  }))

  for (j in 1:no.site){
    if (sum(all.mu_est[j,])>1) {
      pos=which(all.mu_est[j,]==1)
      pos_change=sample(pos,1,FALSE)
      all.mu_est[j,-pos_change]=0
    }
  }

  distribute.site.final=lapply(1:no.cluster,function(k){
    which(all.mu_est[,k]==1)
  })

############## check the clustering process #############
########### sensitivity and specificity of mu ##########

  all.accu.sub=list()
  for (i in 1:length(all.mu.burn)){
    result=all.distribute.site.burn[[i]]
```

```
    in.accu=matrix(rep(0,no.cluster*no.cluster),nrow=no.
       cluster,ncol=no.cluster)
    for (j in 1:no.cluster){
      res=result[[j]]
      for (k in 1:no.cluster){
        in.accu[j,k]=length(res[res %in% distribute.site_true
           [[num.data]][[k]]])
      }
    }
    all.accu.sub[[i]]=in.accu
  }

############# sort the clustering #############
  all.gamma_k.simple.burn=all.gamma_k.simple.clst[19000:20000]
  sort.accu=list()
  for (i in 1:length(all.accu.sub)){
    dif=matrix(rep(0,no.cluster*no.cluster),nrow=no.cluster,
       ncol=no.cluster)
    sort.dist=matrix(0,nrow=no.cluster,ncol=no.cluster)
    dist.matrix=all.accu.sub[[i]]
    coff=all.gamma_k.simple.burn[[i]]
    for (ii in 1:nrow(dist.matrix)){
      if(sum(dist.matrix[ii,])>0) {
        for (j in 1:nrow(gamma_k.true))
          dif[ii,j]=sum((coff[ii,]-gamma_k.true[j,])^2)
        row.index=which(dif[ii,]==min(dif[ii,]))
        sort.dist[row.index[1],]=dist.matrix[ii,]
      }
    }
```

```
    sort.accu[[i]]=sort.dist
  }

## sensitivity and specificity for all the burnin samples ##
  mu.sen.burnin=matrix(nrow=length(all.accu.sub),ncol=no.
     cluster)
  mu.spec.burnin=matrix(nrow=length(all.accu.sub),ncol=no.
     cluster)
  for (i in 1:length(all.accu.sub)){
    sort.dist=sort.accu[[i]]
    for (ii in 1:no.cluster){
      TP=sort.dist[ii,ii]
      FN=sum(sort.dist[,ii])-TP
      FP=sum(sort.dist[ii,])-TP
      TN=sum(diag(sort.dist))-TP
      mu.sen.burnin[i,ii]=TP/(TP+FN)
      mu.spec.burnin[i,ii]=TN/(TN+FP)
    }
  }

###### confusion matrix of delta and senstivitiy and
###### specificity of clsutering for 50 datasets
############ delta: confusion matrix ############
  all.conf_matrix.mean[[num.data]]=conf_matrix
  all.prob.corr.trans.mean[num.data]=sum(diag(all.conf_matrix.
     mean[[num.data]]))/no.site

### mu: median of sensitivity
### and specificity for each dataset
```

```
  for (i in 1:no.cluster){
    all.mu.sen.median[num.data,i]=median(mu.sen.burnin[,i])
    all.mu.spec.median[num.data,i]=median(mu.spec.burnin[,i])
  }
  num.data=num.data+1
}

##### summary of prob of detect delta for all the dataset ####
prob.corr.trans.mean.final=median(all.prob.corr.trans.mean)

## summary sensitivity and specificity for all the dataset ##
all.mu.sen.median.final=c()
all.mu.spec.median.final=c()

for (i in 1:no.cluster){
  sort.mu.sen.median=sort(all.mu.sen.median[,i])
  sort.mu.spec.median=sort(all.mu.spec.median[,i])
  all.mu.sen.median.final[i]=median(all.mu.sen.median[,i])
  all.mu.spec.median.final[i]=median(all.mu.spec.median[,i])
}
```

B.2 Network project

- The R code of projection method [34] is available at `https://github.com/donaldRwilliams/GGMprojpred`.
- The MATLAB code of the Horseshoe method [44] is available at `http://github.com/liyf1988/GHS`.
- The Python code of the HRS method [45] is available from `https://github.com/sakaesaserunrun/onglasso`.