A Covariate-Dependent Approach to Gaussian Graphical Modeling in R

Journal:	*Transactions on Mathematical Software*
Manuscript ID	Draft
Manuscript Type:	2 Algorithm Papers
Date Submitted by the Author:	n/a
Complete List of Authors:	Helwig, Jacob; Texas A&M University, Computer Science Dasgupta, Sutanoy; Texas A&M University, Statistics Zhao, Peng; Texas A&M University, Statistics Mallick, Bani; Texas A&M University, Statistics Pati, Debdeep; Texas A&M University, Statistics
Computing Classification Systems:	Bayesian computation, Variational methods, Multivariate statistics, Statistical software, Probabilistic algorithms

Algorithm xxx: A Covariate-Dependent Approach to Gaussian Graphical Modeling in R

JACOB HELWIG, SUTANOY DASGUPTA, PENG ZHAO, BANI K. MALLICK, and DEBDEEP PATI, Texas A&M Department of Statistics, USA

Graphical models are used to capture complex multivariate relationships and have applications in diverse disciplines such as in biology, physics, and economics. Within this field, Gaussian graphical models aim to identify the pairs of variables whose dependence is maintained even after conditioning on the remaining variables in the data, known as the *conditional dependence structure* of the data. There are many existing software packages for Gaussian graphical modeling, however, these packages make the restrictive assumption that the data are identically distributed or can be partitioned into identically distributed subgroups. Conversely, `covdepGE` is a R implementation of a variational weighted pseudo-likelihood algorithm for modeling the conditional dependence structure as a continuous function of an extraneous covariate. To build on the efficiency of this algorithm, `covdepGE` leverages parallelism and C++ integration with R. Additionally, `covdepGE` provides fully-automated and data-driven hyperparameter specification while maintaining flexibility for the user to decide key components of the estimation procedure. Through an extensive simulation study spanning diverse settings, `covdepGE` is demonstrated to be top of its class in recovering the ground-truth conditional dependence structure while managing time complexity.

CCS Concepts: • **Mathematics of computing → Bayesian computation**; **Variational methods**; **Multivariate statistics**; **Statistical software**; **Probabilistic algorithms**; *Regression analysis*; *Kernel density estimators*; *Continuous functions*; *Approximation algorithms*.

Additional Key Words and Phrases: Gaussian graphical models, variational inference, structure learning, pseudo likelihood, heterogeneous graphs

ACM Reference Format:
Jacob Helwig, Sutanoy Dasgupta, Peng Zhao, Bani K. Mallick, and Debdeep Pati. 2022. Algorithm xxx: A Covariate-Dependent Approach to Gaussian Graphical Modeling in R. 1, 1 (February 2022), 18 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

We present the R [Team 2021] package `covdepGE`, an implementation of the Gaussian graphical modeling (GGM) algorithm introduced by Dasgupta et al. [2022]. GGM seeks to capture the conditional dependence structure (CDS) of the data under the assumption that the data are normally distributed. This distributional assumption is convenient for inference, as the CDS is given by the sparsity structure of the precision matrix [Lauritzen 1996]. There are more than 35 existing R packages for GGM, for example Marchetti [2006]; Vinciotti et al. [2016]; Williams and Mulder [2020] to name just a few. However, as with the existing theory, such as Friedman et al. [2008]; Liu et al. [2010]; Majumdar and Michailidis [2022], all make the restrictive assumption that the precision matrix is homogeneous throughout the

Authors' address: Jacob Helwig, jacob.a.helwig@tamu.edu; Sutanoy Dasgupta, sutanoy@stat.tamu.edu; Peng Zhao, pzhao@stat.tamu.edu; Bani K. Mallick, bmallick@stat.tamu.edu; Debdeep Pati, debdeep@stat.tamu.edu, Texas A&M Department of Statistics, 155 Ireland Street, College Station, Texas, USA, 77843-3143.

data, or that the data may be partitioned into homogeneous subgroups. `covdepGE` avoids this strong assumption by modeling the CDS as varying continuously with an extraneous covariate. Intuitively, this implies that observations having similar extraneous covariate values will have similar precision matrices.

To facilitate information sharing through the extraneous covariate while managing complexity, Dasgupta et al. [2022] propose an efficient variational approximation conducted under the novel weighted pseudo-likelihood framework that we provide in the main function `covdepGE::covdepGE` visualized in Figure 1. We improve time complexity by implementing the parallelism proposed by Dasgupta et al. [2022], and further accelerate inference by executing expensive iterative computations in C++. We build on Dasgupta et al. [2022] by developing a principled, data-driven approach for hyperparameter specification that only requires the user to input data and extraneous covariates to perform inference. At the same time, we maintain users' freedom to directly specify all key aspects of modeling through an intuitive API. Finally, we offer several wrappers around `ggplot2` [Wickham 2016] for seamless visualization of resulting estimates, such as `covdepGE::matViz` and the S3 method `plot.covdepGE`.

Our contributions are as follows:

(1) A package for general covariate-dependent Gaussian graphical modeling. Currently available implementations for heterogeneous graphical modeling are limited in that they either treat the data as homogeneous within subgroups, corresponding to a discrete covariate (e.g., `JGL` [Danaher et al. 2014]), or as varying continuously with time, corresponding to a single-dimensional covariate (e.g., `mgm` [Haslbeck and Waldorp 2020]).
(2) An implementation that maintains low time complexity through a variational approximation, information sharing via similarity weights, parallelism, and C++ integration with R.
(3) Three alternatives for automated hyperparameter specification and a study comparing these approaches.
(4) An experimental comparison of our package to two packages representing the most prevalent approaches for heterogeneous graphical modeling: time-varying [Haslbeck and Waldorp 2020], and homogenous subgroups [Danaher et al. 2014]. We consider diverse settings, varying the dimensionality of our data and the extraneous covariate.

In Section 2, we overview the method of Dasgupta et al. [2022], describe how we implement it in `covdepGE`, and detail our newly-proposed hyperparameter specification strategies that we have included in the package. In Section 3, we perform a thorough simulation study, studying differing hyperparameter specification strategies, and comparing the performance of `covdepGE` to the R packages `JGL` [Danaher et al. 2014] and `mgm` [Haslbeck and Waldorp 2020] in 8 diverse settings.

2 COVARIATE-DEPENDENT GRAPH ESTIMATION

Suppose that $\mathbf{X} \in \mathbb{R}^{n \times p}$ is a data matrix and that $\mathbf{Z} \in \mathbb{R}^{n \times q}$ is a q-dimensional extraneous covariate. Further suppose that the l-th row of $\mathbf{X}$ follows a p-dimensional Gaussian distribution with mean $\mathbf{0} \in \mathbb{R}^p$ and precision matrix $\Omega(z_l) \in \mathbb{R}^{p \times p}$, where z_l is the l-th row of $\mathbf{Z} \in \mathbb{R}^q$ and Ω is a continuous function mapping from the space of extraneous covariates to the space of $p \times p$ non-singular matrices. Then, for the l-th observation, the normality assumption implies that the (j,k) entry of $\Omega(z_l)$ is non-zero if, and only if, variable j and variable k are dependent given the remaining variables in $\mathbf{X}$ [Lauritzen 1996]. Given data satisfying these assumptions, we use `covdepGE` to estimate a graphical representation $\mathcal{G}_l$ of the sparsity structure of $\Omega(z_l)$ for each of the observations in $\mathbf{X}$ as a continuous function of $\mathbf{Z}$ by sharing information between observations with similar values of $\mathbf{Z}$. $\mathcal{G}_l$ contains an undirected edge between the node representations of the

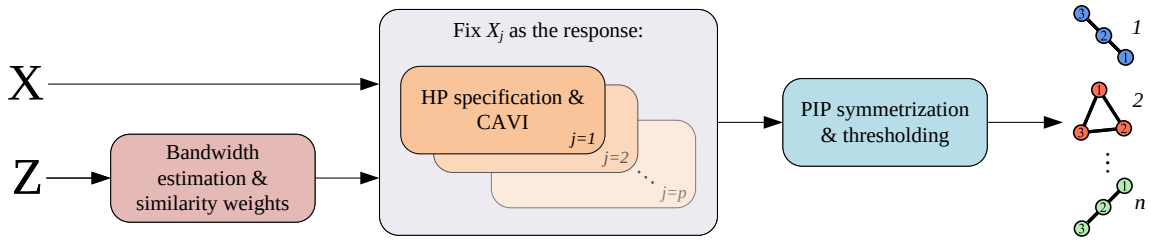

Fig. 1. Computational graph for the `covdepGE::covdepGE` function. We first estimate bandwidths using the extraneous covariate and then use these to calculate pair-wise similarity weights between all observations. Next, we fix each variable in the data as the response and perform variational spike-and-slab regression for each observation using coordinate ascent variational inference, where the regressions are weighted with the observation-specific similarity weights. We select hyperparameters for these regressions using the methods detailed in Section 2.3. In the final step, we post-process the estimated posterior inclusion probabilities from these regressions to estimate the graph describing the conditional dependence structure for each observation.

variables X_j and X_k if, and only if, X_j and X_k are conditionally dependent given the remaining variables for the l-th observation.

We present a pseudo-code representation of the overall algorithm implemented in `covdepGE` in Algorithm 1. Here, we assume that the hyperparameter specification will be fully automated for simplicity, although we note that `covdepGE` permits users to opt out of various parts of the automation as desired. In the sections that follow, we describe the algorithms in the pseudo-code subroutines `get_weights`, `symmetrize`, and `threshold` (Section 2.1), `CAVI` (Section 2.2), `get_bandwidths` (Section 2.3.1), `model_selection` (Section 2.3.2), and `generate_grid` (Section 2.3.3).

2.1 Graph Estimation via the Weighted Pseudo-Likelihood Approach

The weighted pseudo-likelihood approach considered by Dasgupta et al. [2022] for heterogeneous graph estimation introduces similarity weights to the pseudo-likelihood approach [Atchadé 2019; Besag 1975; Meinshausen and Bühlmann 2006] in order to determine the connectivity of $\mathcal{G}_l$ as a continuous function of $\mathbf{Z}$. In the homogenous setting, we can use the pseudo-likelihood approach to estimate the edges of $\mathcal{G}$ by symmetrizing the posterior inclusion probabilities (PIP) of X_k in a regression with X_j fixed as the response ($\text{PIP}_j(X_k)$) and vice versa ($\text{PIP}_k(X_j)$). Here, $\mathcal{G}$ is the CDS for all observations in $\mathbf{X}$, and symmetrization is done with a function mapping from $[0,1] \times [0,1]$ to $[0,1]$ determined by the argument `sym_method` (`"mean"` by default). Then, if the symmetrized PIP is greater than the `edge_threshold` argument (`0.5` by default), we include an undirected edge between X_j and X_k.

We extend this approach to the heterogeneous setting and model Ω as a function of $\mathbf{Z}$ by performing n weighted spike-and-slab regressions for each variable X_j fixed as the response. For the l-th regression, we calculate the similarity weight $w_{i,l}$ for observation i with respect to observation l as

$$w_{i,l} = \frac{n\hat{w}_{i,l}}{\sum_{m=1}^{n} \hat{w}_{m,l}}, \tag{1}$$

where

$$\hat{w}_{i,l} = \mathcal{N}\left(\|z_i - z_l\|; 0, \tau_l\right), \tag{2}$$

such that observations having similar values of $\mathbf{Z}$ to observation l will have larger weights. Here, $\|\cdot\|$ denotes the norm specified by the `norm` argument (*L2* by default) and z_l and z_k are the values of $\mathbf{Z}$ for the l-th and k-th observations. τ_i is

Algorithm 1: Estimate the CDS of $\mathbf{X}$ as a function of $\mathbf{Z}$, selecting hyperparameters according to hp_method

input: $\mathbf{X} \in \mathbb{R}^{n \times p}, \mathbf{Z} \in \mathbb{R}^{n \times q}$, hp_method $\in$ {grid_search, model_average, hybrid}

```
// estimate bandwidths and get similarity weights
τ ← get_bandwidths(Z) ∈ ℝ^n;
W ← get_weights(τ) ∈ ℝ^{n×n};
// fix each of the variables in X as the response (this loop is parallelized)
final_models ← list(p);
for j = 1 to p do
    // generate the hyperparameter grid
    Θ ← generate_grid(data = X, response = j);
    n_Θ ← |Θ|;
    // fit n models for each grid point
    models ← list(n_Θ);
    for t = 1 to n_Θ do
        models_l ← list(n);
        for l = 1 to n do
            // perform variational spike-and-slab weighted with respect to the l-th
            // observation using the t-th hyperparameter grid point
            models_l[l] ← CAVI(data = X, response = j, weights = W_l, hp = Θ_t);
        end
        // save n weighted models corresponding to Θ_t
        models[t] ← models_l;
    end
    // from the resulting n · n_Θ models, select the final model for each observation
    // according to hp_method
    final_models[j] ← model_selection(models, hp_method);
end
// symmetrize the PIP from the final models and threshold the symmetrized PIP into
// adjacency matrices
PIP ← symmetrize(final_models);
graphs ← threshold(PIP);
```

output: n graphs $\mathcal{G}_l \in \mathbb{R}^{p \times p}$

the bandwidth hyperparameter used to determine the similarity weight $w_{i,l}$ of observation i with respect to observation l. Then, the likelihood function for the regression with X_j fixed as the response weighted with respect to the l-th observation is given by

$$L_{l,j}\left(\beta|\mathbf{X}, \mathbf{Z}, \tau, \sigma^2\right) = \prod_{i=1}^{n} \mathcal{N}\left(\mathbf{X}_{i,j}; \sum_{k=1, k\neq j}^{p} \mathbf{X}_{i,k}\beta_k, \frac{\sigma^2}{w_{i,l}}\right). \tag{3}$$

In Equation (1), we normalize the weights for the l-th observation to sum to n so that in our prior (Equation (4)) does not dominate the likelihood (Equation (3)) in our objective function (Equation (9)). For graph estimation, we construct $\mathcal{G}_l$ as in the homogeneous setting using the PIP from the regressions weighted with respect to observation l, $\text{PIP}_{l,j}(\cdot)$.

Through the similarity weights, the weighted pseudo-likelihood approach allows for information sharing conditioned on $\mathbf{Z}$. As opposed to the traditional approach for Bayesian information sharing through a hierarchical modeling scheme, this approach avoids costly MCMC sampling. To add to this efficiency, we observe that since the n regressions for X_j and X_k fixed as the response are independent of each other, we may perform the regressions in parallel by setting `parallel = TRUE`. In factoring parallelism along the variables fixed as the response, we reduce the time complexity of the algorithm by $O(p)$ by registering parallel backend with `num_workers = p`.

2.2 Variational Inference

To limit the number of edges in the estimated graphs, we induce sparsity in $\mathcal{G}$ by placing a spike-and-slab prior on the regression coefficients $\beta \in \mathbb{R}^{p-1}$ [Mitchell and Beauchamp 1988]. That is, we assume *a priori* that the regression coefficients are independently drawn from a $\mathcal{N}(0, \sigma^2\sigma_\beta^2)$ distribution with prior inclusion probability π (the slab) and drawn from a point-mass at 0 with probability $1-\pi$ (the spike). Then, for X_j fixed as the response, the prior is given by

$$p_j^0\left(\beta, \gamma | \sigma^2, \sigma_\beta^2, \pi\right) = \prod_{k=1, k\neq j}^{p} \left(\pi \mathcal{N}\left(\beta_k; 0, \sigma^2\sigma_\beta^2\right)\right)^{\gamma_k} \left((1-\pi)\,\delta_{\{0\}}\left(\beta_k\right)\right)^{1-\gamma_k}, \tag{4}$$

where γ_k is a latent Bernoulli random variable with success probability π that takes on 1 if the k-th regression coefficient is drawn from the slab, and 0 otherwise. Thus, the posterior up to a normalizing constant for (β, γ) with X_j fixed as the response and weighted with respect to observation l is

$$\nu_{l,j}\left(\beta, \gamma | \mathbf{X}, \tau, \sigma^2, \sigma_\beta^2, \pi\right) \propto p_j^0\left(\beta, \gamma | \sigma^2, \sigma_\beta^2, \pi\right) L_{l,j}\left(\beta | \mathbf{X}, \mathbf{Z}, \tau, \sigma^2\right). \tag{5}$$

We avoid expensive posterior sampling by approximating all posterior quantities, including $\text{PIP}_{l,j}(X_k)$, using a block mean-field variational approximation for spike-and-slab regression, as in Carbonetto and Stephens [2012]. Under the variational paradigm, parameters are estimated by first specifying a variational family of densities $\mathcal{Q}$, and then selecting $q^* \in \mathcal{Q}$ that minimizes the Kullback-Leibler divergence between $\mathcal{Q}$ and the posterior ν [Jordan et al. 1999]. Formally, q^* is defined as

$$q^* = \underset{q \in \mathcal{Q}}{\arg\min}\left(D_{\text{KL}}(q \| \nu)\right), \tag{6}$$

where

$$D_{\text{KL}}(q \| \nu) = \mathbb{E}_q \log \frac{q}{\nu}. \tag{7}$$

In practice, the KL divergence is not a tractable quantity, however, minimizing the divergence is equivalent to maximizing the evidence lower bound (ELBO), which is a tractable objective [Blei et al. 2017]. We perform coordinate ascent variational inference (CAVI) to optimize the ELBO over the variational parameters $\phi_{l,j} = (\mu, s^2, \alpha) \in \mathbb{R}^{p-1\times 3}$ with respect to the spike-and-slab family $\mathcal{Q}$ defined as

$$\mathcal{Q} = \left\{ \prod_{k=1, k\neq j}^{p} \left(\alpha_k \mathcal{N}\left(\beta_k; \mu_k, s_k^2\right)\right)^{\gamma_k} \left((1-\alpha_k)\,\delta_{\{0\}}\left(\beta_k\right)\right)^{1-\gamma_k} : \mu_k \in \mathbb{R}, s_k^2 \in \mathbb{R}^+, \alpha_k \in [0,1] \right\}. \tag{8}$$

For each of the regression coefficients β_k, these parameters are the mean μ_k and variance s_k^2 of the slab and the probability α_k that the coefficient is drawn from $\mathcal{N}(\mu_k, s_k^2)$.

We derive the variational updates for each coordinate ascent step by taking the partial derivatives of the ELBO with respect to the variational parameters, where the ELBO is defined as

$$\text{ELBO}(\phi_{l,j}) = \mathbb{E}_q \log p_j^0\left(\beta, \gamma | \sigma^2, \sigma_\beta^2, \pi\right) + \mathbb{E}_q \log L_{l,j}\left(\beta | \mathbf{X}, \mathbf{Z}, \tau, \sigma^2\right) - \mathbb{E}_q \log q(\beta, \gamma; \phi_{l,j}). \tag{9}$$

Here, we denote the final ELBO of the regression with X_j fixed as the response weighted with respect to observation l as $\text{ELBO}\left(\phi_{l,j}\right)$ to emphasize that we view this objective as a function of the variational parameters. Finally, we approximate $\text{PIP}_{l,j}(X_k)$ using α_k, which we symmetrize and threshold to construct the graph estimates. In Appendix A.1, we present the full ELBO and variational updates.

As opposed to fixing variable X_j as the response and performing CAVI for the n regressions sequentially, we perform the optimizations simultaneously, using efficient matrix operations where possible. With each of the n sets of α as the rows of an $n \times (p-1)$ matrix, we end CAVI for all n regressions when the Frobenius norm of the change in the α matrix is less than the argument `alpha_tol` or the number of CAVI updates exceeds the argument `max_iter`. We further improve time complexity by executing the iteration-intensive CAVI entirely with efficient C++ code.

2.3 Automated Hyperparameter Specification

`covdepGE` requires the specification of a bandwidth hyperparameter and spike-and-slab hyperparameters for each regression. Since manual hyperparameter specification can be time consuming and difficult for the user to do effectively, we provide data-driven hyperparameter specification routines within `covdepGE`. We offer the user varying levels of involvement in this procedure, ranging from fully automated to fully manual.

2.3.1 Bandwidth Hyperparameter. The bandwidth hyperparameter $\tau \in \mathbb{R}^n$ regulates the amount of information that is shared through the similarity weights. Smaller values of $\tau_l \in \mathbb{R}$ result in a local estimate of $\mathcal{G}_l$, where the estimate of $\mathcal{G}_l$ is conditioned on information restricted to observations with an extraneous covariate close to z_l. In contrast, larger values of τ_l give rise to a global estimate of $\mathcal{G}_l$. On the extremes, as $\tau_l \to 0$ for all $l \in 1, 2, \ldots, n$, graph estimates are mutually independent for all observations, and as $\tau_l \to \infty$, all observations share a common graph estimate.

We select τ using the 2-step approach for density estimation described by Abramson [1982]. Under this approach, bandwidths are initialized using Silverman's rule of thumb [Silverman 2017], and the density is subsequently refined by updating the bandwidth values. We follow this methodology, which we describe in detail in Appendix A.2, to estimate the density of $\mathbf{Z}$, and use the updated bandwidths from the second step for τ.

2.3.2 Spike-and-slab Hyperparameters. Each spike-and-slab regression requires the specification of 3 hyperparameters: π (the prior probability of inclusion), σ^2 (the prior residual variance), and σ_β^2 (the prior variance of the slab). We offer 3 methods for automated hyperparameter specification via the `hp_method` argument: `grid_search`, `model_average`, and `hybrid`. Under each approach, we generate a hyperparameter candidate grid Θ for each variable X_j fixed as the response by taking the Cartesian product between the arguments `ssq`, `sbsq`, and `pip` (candidate values for σ^2, σ_β^2, and π). We next describe each of these approaches for the regression with X_j fixed as the response weighted with respect to observation l. For this regression, we denote the variational parameter estimates resulting from the choice of hyperparameter $\theta \in \Theta$ as $\phi_{l,j}(\theta) \in \mathbb{R}^{p-1 \times 3}$ and the estimates for the final model as $\phi_{l,j}^* \in \mathbb{R}^{p-1 \times 3}$. We compare the performance of each of these methods in Section 3.3.1.

In `grid_search`, we select the optimal $\theta^* \in \Theta$ by maximizing the total ELBO summed across all n regressions. That is,

$$\theta^* = \arg\max_{\theta \in \Theta} \sum_{l=1}^{n} \text{ELBO}\left(\phi_{l,j}\right), \tag{10}$$

with the final estimates given by $\phi^*_{l,j} = \phi_{l,j}(\theta^*)$.

Alternatively, we average over Θ in `model_average`, where the unnormalized model averaging weights $\hat{\omega}$ are the exponentiated ELBO as proposed in Carbonetto and Stephens [2012]. For all $\tilde{\theta} \in \Theta$, the unnormalized model averaging weights are given by

$$\hat{\omega}(\tilde{\theta}) = \exp \text{ELBO}\left(\phi_{l,j}(\tilde{\theta})\right), \tag{11}$$

which are normalized to

$$\omega(\tilde{\theta}) = \frac{\hat{\omega}(\tilde{\theta})}{\sum_{\theta \in \Theta} \hat{\omega}(\theta)}. \tag{12}$$

Then, the final variational estimates are averaged over $\theta \in \Theta$ as

$$\phi^*_{l,j} = \sum_{\theta \in \Theta} \omega(\theta)\phi_{l,j}(\theta). \tag{13}$$

Finally, similar to the strategy used by the package `varbvs` [Carbonetto et al. 2017], we combine the previous approaches in `hybrid` by averaging over `pip` while grid searching for `ssq` and `sbsq`. For each $\tilde{\pi} \in$ `pip`, we select $\theta_{\tilde{\pi}} = \left(\tilde{\pi}, \tilde{\sigma}^2, \tilde{\sigma}^2_\beta\right)$ as

$$\theta_{\tilde{\pi}} = \arg\max_{\theta \in \{\tilde{\pi}\} \times \texttt{ssq} \times \texttt{sbsq}} \sum_{l=1}^{n} \text{ELBO}\left(\phi_{l,j}(\theta)\right). \tag{14}$$

Next, we define the model averaging weights as

$$\hat{\omega}(\theta_{\tilde{\pi}}) = \exp \text{ELBO}\left(\phi_{l,j}(\theta_{\tilde{\pi}})\right), \tag{15}$$

$$\omega(\theta_{\tilde{\pi}}) = \frac{\hat{\omega}(\theta_{\tilde{\pi}})}{\sum_{\pi \in \texttt{pip}} \hat{\omega}(\theta_\pi)}. \tag{16}$$

Then, the final variational estimates are averaged over $\tilde{\pi} \in$ `pip` as

$$\phi^*_{l,j} = \sum_{\pi \in \texttt{pip}} \omega(\theta_\pi)\phi_{l,j}(\theta_\pi) \tag{17}$$

To improve the efficiency of the search step in `grid_search` and `hybrid` (i.e., the arg max step), we perform a reduced CAVI for each of the hyperparameter candidates for at most `max_iter_grid` iterations. We then perform a subsequent full CAVI for at most `max_iter` iterations using the subset of Θ that maximized the total ELBO in the first step. By setting `max_iter_grid` $= \frac{1}{h}$`max_iter` for $h > 1$, we further reduce complexity by $O(h)$.

2.3.3 Hyperparameter Candidate Grids. We use a data-driven approach to generate the hyperparameter candidate grids `ssq`, `sbsq`, and `pip` spaced uniformly between an upper end point and lower end point, inclusive. We calculate the upper endpoints dependent on the variable X_j fixed as the response. For σ^2, we calculate the upper bound by scaling the sample variance of X_j by the argument `ssq_mult` (1.5 by default). To calculate the upper bound for π, we use the LASSO in the `glmnet` package [Friedman et al. 2010] and regress the remaining variables on X_j to estimate the proportion of non-zero coefficients. Although the LASSO assumes that the CDS is homogeneous, it enforces sparsity while giving a rough estimate to the optimal π.

Finally, we derive a rough upper bound on the signal-to-noise ratio that depends on the upper bound for σ_β^2. For a general spike-and-slab regression of $y \in \mathbb{R}^n$ and $X \in \mathbb{R}^{n \times p}$, we first observe that the prior variance of the regression coefficients is given by $\text{Var}(\beta_k) = \pi\sigma^2\sigma_\beta^2$. Then, under the simplifying assumption that the columns of X are pairwise independent, the prior signal-to-noise ratio is given by:

$$\text{SNR} = \frac{\text{Var}(X\beta)}{\text{Var}(y - X\beta)} = \pi\sigma_\beta^2 \sum_{k=1}^{p} \text{Var}(X_k) \tag{18}$$

Any reasonable upper bound for the signal-to-noise ratio can be used for the left-hand side of Equation (18) which we parameterize with the `snr_upper` argument (25 by default). Replacing π with the upper bound for `pip` and approximating $\text{Var}(X_k)$ with the sample variance of the k-th column of $\mathbf{X}$ on the right-hand side induces a rough upper bound on σ_β^2.

3 SIMULATION STUDY

We perform our simulation study[1] on dual 14-Core (28 threads each) Intel Xeon Gold 6132 processors (@2.6GHz) with 96GB of total RAM. In each setting we consider, we perform 50 trials by first randomly generating an extraneous covariate $\mathbf{Z} \in \mathbb{R}^{n \times q}$ and then generating $\mathbf{X} \in \mathbb{R}^{n \times p}$ conditioned on $\mathbf{Z}$. We define 8 settings through all combinations of $q \in \{1, 2\}$ and $p \in \{10, 25, 50, 100\}$, with $n = 225$ fixed.

We evaluate `covdepGE` on three metrics: sensitivity, specificity, and runtime. We define sensitivity as the proportion of edges correctly detected in the ground truth dependence structures $\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_n$, not including self-loops. We define specificity as the proportion of non-edges correctly detected. Lastly, we study the real time it takes for `covdepGE` to perform inference which, given that experiments are run on 56 threads, is $O(p)$ less than the CPU time.

We begin by comparing the hyperparameter specification strategies discussed in Section 2.3.2 in the $q = 1$ settings before moving on to compare `covdepGE` to 2 baselines for heterogeneous graphical modeling in R: `JGL` [Danaher et al. 2014] and `mgm` [Haslbeck and Waldorp 2020]. These baselines are a representative sample of the currently available packages for modeling heterogeneous graphs, as `mgm` assumes that the graphs vary with time, while `JGL` assumes that the data can be partitioned into subgroups such that the conditional dependence structure is homogeneous within each of the subgroups. We note that since neither of these packages have enabled parallelism, the real time we report for them is equal to their CPU time.

3.1 Competitors

The `mgm::tvmgm` function applies a regularized kernel-smoothing method to model the precision matrices as varying continuously in time. We use the `mgm::bwSelect` function to select the bandwidth hyperparameter for this function with candidate values and settings based on a high-dimensional Gaussian example provided by the authors of the package which we detail in Appendix B.1. We note that although in the setting with $q = 1$, it is natural to frame $\mathbf{Z}$ as indexing time, `mgm` is not directly applicable for $q = 2$. Therefore, we sort $\mathbf{Z}$ to $\tilde{\mathbf{Z}}$ for this setting using a greedy algorithm described in Appendix B.2 such that the observation indices $T = \{1, 2, \ldots, n\}$ mapping $\mathbf{Z}$ to $\tilde{\mathbf{Z}}$ can be interpreted as time. We then apply `mgm` on this data, using the single-dimensional T as the time index. To demonstrate the fairness of this approach, we evaluate `covdepGE` first using $\mathbf{Z}$ as the extraneous covariate, and then using T. We refer to the latter as `covdepGE_time`.

[1] The simulation study scripts are available at
https://github.com/JacobHelwig/covdepGE/tree/master/TOMS_simulation_study

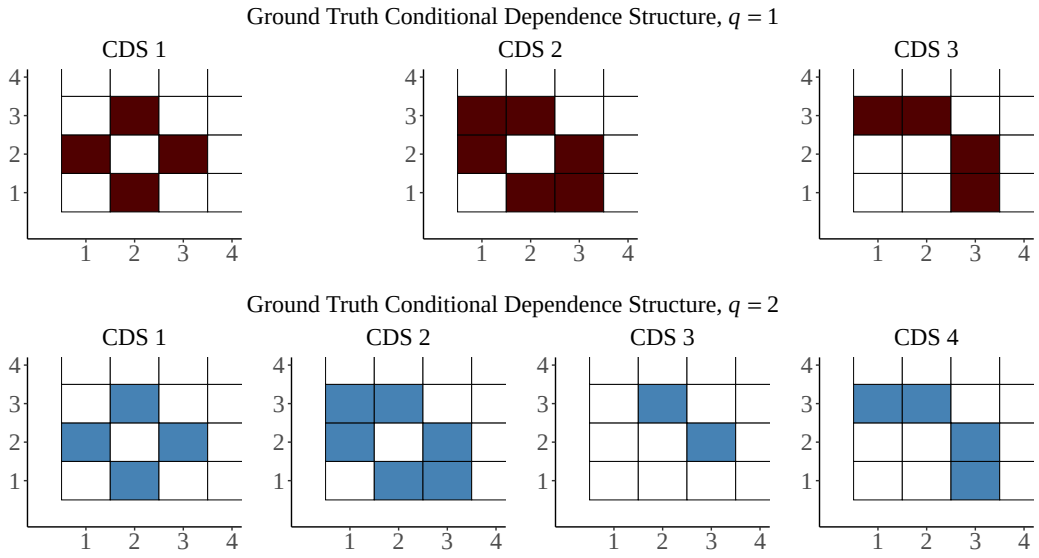

Fig. 2. Upper-left 3×3 block of the ground truth conditional dependence structure visualized using `covdepGE::matViz`. The remaining entries are all 0. *Top*: CDS for $q = 1$ settings. **CDS 1** – $z_l \in [-3, -1]$, **CDS 2** – $z_l \in [-1, 1]$, **CDS 3** – $z_l \in [1, 3]$. *Bottom*: CDS for $q = 2$ settings. **CDS 1** – $z_l \in ([-3, -1] \times [-3, -1]) \cup ([-1, 1] \times [-3, -1])$. **CDS 2** – $z_l \in ([-3, -1] \times [-1, 1]) \cup ([-3, -1] \times [1, 3]) \cup ([-1, 1] \times [-1, 1]) \cup ([-1, 1] \times [1, 3])$. **CDS 3** – $z_l \in ([1, 3] \times [-3, -1])$. **CDS 4** – $z_l \in ([1, 3] \times [-1, 1]) \cup ([1, 3] \times [1, 3])$

The `JGL` package applies the fused graphical LASSO to jointly estimate the precision matrices for each of the subgroups from a given partition of the data. To incorporate information from $\mathbf{Z}$, we apply Gaussian mixture model-based clustering with the package `Mclust` [Scrucca et al. 2016] to partition $\mathbf{X}$ into K subgroups based on $\mathbf{Z}$, selecting K by minimizing the BIC. Since `JGL` does not offer any specification routines for the shrinkage hyperparameters λ_1 and λ_2, we implement the 2-step approach proposed in Section 6 of Danaher et al. [2014] to minimize the approximate AIC over a grid of λ_1 and λ_2.

We additionally considered `HeteroGGM` [Ren et al. 2021] as a competitor. While `HeteroGGM` is also a penalized fusion-based method similar to `JGL`, it does not assume that the subgroups of the data are known *a priori*, and instead estimates subgroup membership jointly with precision matrices. However, `HeteroGGM` requires the subgroup means to be sufficiently separable to estimate membership. Since all observations in our simulation study were drawn from a $\mathbf{0}$-mean Gaussian, we found `JGL` coupled with `Mclust` to be a more appropriate choice.

3.2 Data Generation

For each trial in all settings, we begin by generating $\mathbf{Z} \in \mathbb{R}^{n \times q}$. For $q = 1$, we draw 75 samples each from the uniform distribution on the intervals $(-3, -1)$, $(-1, 1)$, and $(1, 3)$. For $q = 2$, we draw 25 times each from the uniform distribution on the 9 regions defined by $\{Z_1 \times Z_2 : Z_1, Z_2 \in \{[-3, -1], [-1, 1], [1, 3]\}\}$.

We next generate $\mathbf{X} \in \mathbb{R}^{n \times p}$ with a precision matrix $\Omega(z_l) \in \mathbb{R}^{p \times p}$ that varies as a continuous function of $\mathbf{Z}$. For $q = 1$, we let z_l be the extraneous covariate for the l-th observation and define $z_{l,1} = z_{l,2} = z_l \in \mathbb{R}$, while for $q = 2$, we

Table 1. Comparison of hyperparameter specification strategies in the $q = 1$ settings across 50 trials, presented as "*mean* (*sd*)". Sensitivity and specificity are defined as the number of edges and non-edges correctly recovered from the ground-truth CDS excluding self-loops. Time is the elapsed real time for the model to fit in seconds.

p	HP Method	Sensitivity(%)	Specificity(%)	Time (seconds)
10	`grid_search`	**90.65**(5.80)	99.24(0.87)	**4.75**(0.18)
	`hybrid`	89.96(5.59)	**99.39**(0.75)	5.46(0.41)
	`model_average`	88.97(6.19)	99.01(0.78)	20.93(3.71)
25	`grid_search`	**87.32**(6.76)	99.70(0.22)	**12.31**(0.39)
	`hybrid`	86.49(6.99)	**99.77**(0.18)	14.26(1.20)
	`model_average`	85.73(7.00)	99.44(0.23)	62.92(10.96)
50	`grid_search`	**84.35**(6.96)	99.71(0.18)	**42.20**(2.99)
	`hybrid`	83.99(6.74)	**99.82**(0.07)	44.77(2.35)
	`model_average`	84.33(5.94)	99.60(0.09)	166.94(15.87)
100	`grid_search`	80.81(6.82)	99.68(0.15)	**221.77**(17.29)
	`hybrid`	80.91(6.38)	**99.86**(0.04)	236.97(14.32)
	`model_average`	**83.16**(5.85)	99.70(0.04)	961.26(67.73)

define $(z_{l,1}, z_{l,2}) = z_l \in \mathbb{R}^2$. Then, for $q \in \{1, 2\}$, the j, k entry of the precision matrix for the l-th observation is given by

$$\Omega(z_l)_{j,k} = \begin{cases} 2 & j = k \\ \mathbb{1}(z_{l,1} < 1) \cdot \min\left(1, \frac{1}{2} - \frac{1}{2} z_{l,1}\right) & (j,k) \in \{(1,2),(2,1)\} \\ \mathbb{1}(z_{l,2} > -1) \cdot \min\left(1, \frac{1}{2} + \frac{1}{2} z_{l,2}\right) & (j,k) \in \{(1,3),(3,1)\} \\ 0 & \textit{otherwise.} \end{cases} \tag{19}$$

We note that for $q = 1$, all 75 observations with $z_l \in [-3, -1]$ have identical precision matrices, as do observations with $z_l \in [1, 3]$. We further observe that for observations with $z_l \in [-1, 1]$, the entries of $\Omega(z_l)$ vary linearly with z_l, and that

$$\lim_{z_l \downarrow -1} \|\Omega(z_l) - \Omega(-1)\| = 0, \tag{20}$$

and similarly that

$$\lim_{z_l \uparrow 1} \|\Omega(z_l) - \Omega(1)\| = 0, \tag{21}$$

where $\|\cdot\|$ denotes any matrix norm. We visualize the conditional dependence structures corresponding to these precision matrices for $q \in \{1, 2\}$ in Figure 2.

3.3 Single-Dimensional Extraneous Covariate

We begin by comparing the performance of different hyperparameter specification strategies in the $q = 1$ setting before moving on to study the performance of `covdepGE` relative to the baselines.

3.3.1 Hyperparameter Strategy Comparison. We present results for each hyperparameter specification strategy in Table 1. We observe that for larger p, `model_average` detects more edges correctly but is significantly slower than `grid_search` and `hybrid`. As described in Section 2.3.2, this is because `grid_search` and `hybrid` both perform an initial abbreviated CAVI for each $\theta \in \Theta$ for at most `max_iter_grid` iterations (10 by default). Based on this initial

Table 2. Comparison of heterogeneous graphical modeling methods in the $q = 1$ settings. JGL is the method of Danaher et al. [2014], where the subgroups of the data are chosen by applying Gaussian mixture models clustering to **Z**. mgm is the method of Haslbeck and Waldorp [2020].

p	Method	Sensitivity(%)	Specificity(%)	Time (seconds)
10	covdepGE	**89.96**(5.59)	99.39(0.75)	**5.46**(0.41)
	JGL	79.59(8.28)	98.63(1.01)	26.41(15.51)
	mgm	79.69(9.45)	**99.88**(0.33)	760.40(21.33)
25	covdepGE	**86.49**(6.99)	99.77(0.18)	**14.26**(1.20)
	JGL	78.74(11.25)	99.71(0.24)	46.06(72.35)
	mgm	73.51(9.30)	**99.99**(0.03)	2042.37(41.93)
50	covdepGE	**83.99**(6.74)	99.82(0.07)	**44.77**(2.35)
	JGL	72.09(15.35)	99.94(0.05)	107.60(252.87)
	mgm	66.90(9.56)	**100.00**(0.01)	4551.70(127.56)
100	covdepGE	**80.91**(6.38)	99.86(0.04)	236.97(14.32)
	JGL	70.18(15.88)	99.98(0.01)	**190.20**(286.73)
	mgm	59.64(7.26)	**100.00**(0.00)	11112.32(260.85)

search, they select an optimal subset of Θ and perform a full CAVI for at most max_iter iterations (100 by default). By contrast, since all $\theta \in \Theta$ contribute to the final model produced by model_average, CAVI must run for the full max_iter iterations (or until convergence) for all $\theta \in \Theta$, resulting in substantially increased overhead. To assess whether performing CAVI for the full number of steps in the initial search significantly changes results, we repeat this experiment in Appendix C.1, setting max_iter_grid = max_iter for hybrid and grid_search. Unsurprisingly, the runtime for all strategies after this modification is roughly equal. We also observe negligible changes in sensitivity and specificity, suggesting the effectiveness of selecting the optimal subset of Θ based on an abbreviated search.

We note that hybrid offers the best specificity while estimating the second most edges correctly in all settings except $p = 50$. For this reason, in addition to the speed-up relative to model_average, we select hybrid as the default hyperparameter specification strategy.

3.3.2 Comparison to Baselines. We present results for covdepGE and the baselines in Table 2. covdepGE improves the baseline sensitivity by over 7% in all settings. We note that although mgm has the lowest sensitivity, its conservative estimates allow it to excel in specificity. However, the time complexity of mgm is more than 45 times greater than covdepGE in all settings. Compared to JGL, covdepGE is twice as fast in all settings except for $p = 100$, although we observe a large amount of variability in the runtimes for JGL. During the shrinkage hyperparameter search in some trials, JGL would iterate for an abnormally long time on smaller values of λ_1. This is because smaller λ_1 result in less sparse estimates that require a greater amount of computation during optimization for operations such as matrix inversion. We remove the effects of outliers from our analysis in Appendix C.2 by aggregating results using median in place of mean and interquartile range in place of standard deviation.

Although JGL is faster on average than covdepGE in the $p = 100$ setting, JGL requires users to either directly specify hyperparameters *ad hoc* (a task that is often time-consuming and difficult for the user to do effectively) or to manually implement a hyperparameter specification scheme as we did. Additionally, while covdepGE estimates the graphs as a

Table 3. Comparison of heterogeneous graphical modeling methods in the $q = 2$ settings. The time index for `mgm` and `covdepGE_time` is obtained by applying a greedy sorting algorithim to $\mathbf{Z}$.

p	Method	Sensitivity(%)	Specificity(%)	Time (seconds)
10	covdepGE	**91.03**(8.93)	**99.48**(0.59)	5.62(0.37)
	JGL	74.98(8.95)	98.46(1.31)	44.45(15.05)
	mgm	70.83(18.24)	99.40(0.53)	754.48(18.67)
	covdepGE_time	87.95(9.76)	98.78(0.72)	**5.42**(0.40)
25	covdepGE	**84.74**(12.26)	99.89(0.10)	14.41(1.16)
	JGL	62.00(15.07)	99.81(0.21)	62.43(27.48)
	mgm	56.90(15.45)	**99.96**(0.05)	2036.89(51.83)
	covdepGE_time	80.78(13.29)	99.63(0.23)	**14.29**(1.28)
50	covdepGE	**83.66**(11.87)	99.96(0.03)	**43.88**(2.17)
	JGL	55.55(15.81)	99.96(0.05)	127.38(75.39)
	mgm	48.52(10.86)	**100.00**(0.01)	4522.41(96.11)
	covdepGE_time	78.20(12.15)	99.80(0.07)	45.60(2.54)
100	covdepGE	**80.80**(13.01)	99.98(0.01)	**225.97**(13.38)
	JGL	48.63(17.45)	99.99(0.01)	325.51(280.21)
	mgm	45.56(7.50)	**100.00**(0.00)	11008.27(212.55)
	covdepGE_time	73.37(13.63)	99.85(0.04)	235.00(14.83)

continuous function of $\mathbf{Z}$, `JGL` relies on a discretization of the covariate space that is independent of graph estimation via `Mclust` to incorporate $\mathbf{Z}$, and incorrectly models the precision matrix as homogeneous within each of the subgroups.

3.4 Multi-Dimensional Extraneous Covariate

We present results for $q = 2$ in Table 3. We observe that `covdepGE` improves the baseline sensitivity in all settings by over 16%. Further, the reduction in sensitivity of `covdepGE` from $p = 10$ to $p = 100$ is less than 11%, while for the baselines, it is more than 25%, demonstrating the robustness of `covdepGE` to increased dimensionality. Unlike the $q = 1$ settings, `covdepGE` has a faster runtime than `JGL` in all settings including $p = 100$. As we show in Appendix C.3, this is due to the increase in the number of subgroups estimated by `Mclust` relative to the $q = 1$ settings. Because `JGL` jointly estimates the precision matrix in each of the subgroups, a greater number of subgroups increases computation.

`mgm` demonstrates the best specificity in all settings except $p = 10$, however, the generalization to $q > 1$ was not straightforward as it was for `covdepGE`. We note that although `covdepGE_time` outperforms the baselines in sensitivity, its performance is significantly reduced compared to `covdepGE`. This highlights the importance of directly incorporating multidimensional covariates into the estimation procedure as opposed to relying on a reduction to the covariate.

4 CONCLUSION

Our package leverages parallelism and C++ code on top of the efficient algorithm proposed in Dasgupta et al. [2022] to further accelerate inference. We further propose and provide fully-automated and effective hyperparameter specification. As we have demonstrated in a variety of settings through our simulation study, `covdepGE` manages time complexity and consistently recovers the edges of the ground-truth CDS as a function of $\mathbf{Z}$ more reliably than baseline methods for heterogeneous graph estimation. In addition to applications in genetics as illustrated by the case study in Dasgupta et al.

[2022], we expect covdepGE to find applications across diverse disciplines due to its ability to incorporate extraneous covariates in graph estimation for data that are not necessarily identically distributed. Future work should improve the scalability of the algorithm to high-dimensional settings. covdepGE is available on CRAN[2] and GitHub[3].

REFERENCES

Ian S. Abramson. 1982. On Bandwidth Variation in Kernel Estimates-A Square Root Law. *The Annals of Statistics* 10, 4 (1982), 1217–1223. https://www.jstor.org/stable/2240724 Publisher: Institute of Mathematical Statistics.

Yves F. Atchadé. 2019. Quasi-Bayesian estimation of large Gaussian graphical models. *Journal of Multivariate Analysis* 173 (Sept. 2019), 656–671. https://doi.org/10.1016/j.jmva.2019.03.005

Julian Besag. 1975. Statistical Analysis of Non-Lattice Data. *Journal of the Royal Statistical Society. Series D (The Statistician)* 24, 3 (1975), 179–195. https://doi.org/10.2307/2987782 Publisher: [Royal Statistical Society, Wiley].

David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. 2017. Variational Inference: A Review for Statisticians. *J. Amer. Statist. Assoc.* 112, 518 (April 2017), 859–877. https://doi.org/10.1080/01621459.2017.1285773

Peter Carbonetto and Matthew Stephens. 2012. Scalable Variational Inference for Bayesian Variable Selection in Regression, and Its Accuracy in Genetic Association Studies. *Bayesian Analysis* 7, 1 (March 2012), 73–108. https://doi.org/10.1214/12-BA703

Peter Carbonetto, Xiang Zhou, and Matthew Stephens. 2017. varbvs: Fast Variable Selection for Large-scale Regression. https://doi.org/10.48550/arXiv.1709.06597

Patrick Danaher, Pei Wang, and Daniela M. Witten. 2014. The joint graphical lasso for inverse covariance estimation across multiple classes. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 76, 2 (2014), 373–397. https://doi.org/10.1111/rssb.12033

Sutanoy Dasgupta, Peng Zhao, Prasenjit Ghosh, Debdeep Pati, and Bani Mallick. 2022. An approximate Bayesian approach to covariate-dependent graphical modeling. (2022), 1–59.

J. Friedman, T. Hastie, and R. Tibshirani. 2008. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics* 9, 3 (July 2008), 432–441. https://doi.org/10.1093/biostatistics/kxm045

Jerome Friedman, Trevor Hastie, and Rob Tibshirani. 2010. Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of statistical software* 33, 1 (2010), 1–22. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2929880/

Jonas M. B. Haslbeck and Lourens J. Waldorp. 2020. mgm: Estimating Time-Varying Mixed Graphical Models in High-Dimensional Data. *Journal of Statistical Software* 93 (April 2020), 1–46. https://doi.org/10.18637/jss.v093.i08

Michael I. Jordan, Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Saul. 1999. An Introduction to Variational Methods for Graphical Models. *Machine Learning* 37, 2 (Nov. 1999), 183–233. https://doi.org/10.1023/A:1007665907178

Steffen L. Lauritzen. 1996. *Graphical Models.* Clarendon Press.

Han Liu, Xi Chen, Larry Wasserman, and John Lafferty. 2010. Graph-Valued Regression. In *Advances in Neural Information Processing Systems*, Vol. 23. Curran Associates, Inc., 1–9. https://proceedings.neurips.cc/paper/2010/hash/821fa74b50ba3f7cba1e6c53e8fa6845-Abstract.html

Subhabrata Majumdar and George Michailidis. 2022. Joint Estimation and Inference for Data Integration Problems based on Multiple Multi-layered Gaussian Graphical Models. *Journal of Machine Learning Research* 23 (2022), 1–53.

Giovanni M. Marchetti. 2006. Independencies Induced from a Graphical Markov Model After Marginalization and Conditioning: The R Package ggm. *Journal of Statistical Software* 15 (Feb. 2006), 1–15. https://doi.org/10.18637/jss.v015.i06

Nicolai Meinshausen and Peter Bühlmann. 2006. High-dimensional graphs and variable selection with the Lasso. *The Annals of Statistics* 34, 3 (June 2006), 1436–1462. https://doi.org/10.1214/009053606000000281

T. J. Mitchell and J. J. Beauchamp. 1988. Bayesian Variable Selection in Linear Regression. *J. Amer. Statist. Assoc.* 83, 404 (Dec. 1988), 1023–1032. https://doi.org/10.1080/01621459.1988.10478694

Mingyang Ren, Sanguo Zhang, Qingzhao Zhang, and Shuangge Ma. 2021. HeteroGGM: an R package for Gaussian graphical model-based heterogeneity analysis. *Bioinformatics* 37, 18 (Sept. 2021), 3073–3074. https://doi.org/10.1093/bioinformatics/btab134

Luca Scrucca, Michael Fop, T. Brendan Murphy, and Adrian E. Raftery. 2016. mclust 5: Clustering, Classification and Density Estimation Using Gaussian Finite Mixture Models. *The R journal* 8, 1 (Aug. 2016), 289–317. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5096736/

B. W. Silverman. 2017. *Density Estimation for Statistics and Data Analysis.* Routledge, New York. https://doi.org/10.1201/9781315140919

R Core Team. 2021. R: A Language and Environment for Statistical Computing. https://www.R-project.org/

Veronica Vinciotti, Luigi Augugliaro, Antonino Abbruzzo, and Ernst C. Wit. 2016. Model selection for factorial Gaussian graphical models with an application to dynamic regulatory networks. *Statistical Applications in Genetics and Molecular Biology* 15, 3 (June 2016), 193–212. https://doi.org/10.1515/sagmb-2014-0075

Hadley Wickham. 2016. *ggplot2: Elegant Graphics for Data Analysis.* Springer-Verlag New York. https://ggplot2.tidyverse.org

Donald Williams and Joris Mulder. 2020. BGGM: Bayesian Gaussian Graphical Models in R. *Journal of Open Source Software* 5, 51 (July 2020), 2111. https://doi.org/10.21105/joss.02111

[2] https://cran.r-project.org/package=covdepGE

[3] https://github.com/JacobHelwig/covdepGE/

A MODEL DETAILS

A.1 Full ELBO and Variational Updates

Recall from Equation (9) that for $l \in 1, 2, \ldots, n$ and $j \in 1, 2, \ldots, p$, the ELBO for the regression with X_j fixed as the response and weighted with respect to observation l is a function of the variational parameters $\phi_{l,j} = (\mu, s^2, \alpha) \in \mathbb{R}^{p-1\times 3}$ and is given by

$$\text{ELBO}(\phi_{l,j}) = \mathbb{E}_q \log p_j^0 \left(\beta, \gamma | \sigma^2, \sigma_\beta^2, \pi\right) + \mathbb{E}_q \log L_{l,j} \left(\beta | \mathbf{X}, \mathbf{Z}, \tau, \sigma^2\right) - \mathbb{E}_q \log q(\beta, \gamma; \phi_{l,j}). \tag{22}$$

Without loss of generality, assume that $j = p$. Let $w_{i,l}$ denote the similarity weight for observation i with respect to observation l, and $x_{i,k}$ denote the i,k entry of $\mathbf{X} \in \mathbb{R}^{n\times p}$. We denote the prior probability of inclusion, which we have thus far referred to as π, as ξ, and use the usual numerical definition for π. Then, for the hyperparameters $\tau, \xi, \sigma^2, \sigma_\beta^2 \in \mathbb{R}$, each of the terms in the ELBO is given by

$$\mathbb{E}_q \log p_j^0 \left(\beta, \gamma | \sigma^2, \sigma_\beta^2, \pi\right) = \sum_{k=1}^{p-1} -\frac{\alpha_k}{2} \log\left(2\pi\sigma^2\sigma_\beta^2\right) - \frac{\alpha_k\left[\mu_k^2 + s_k^2\right]}{2\sigma^2\sigma_\beta^2} + \alpha_k \log(\xi) + (1-\alpha_k)\log(1-\xi), \tag{23}$$

$$\begin{aligned}\mathbb{E}_q \log L_{l,j} \left(\beta | \mathbf{X}, \mathbf{Z}, \tau, \sigma^2\right) = &-\frac{n}{2}\log\left(2\pi\sigma^2\right) + \frac{1}{2}\sum_{i=1}^{n}\log(w_{i,l}) \\ &- \frac{1}{2\sigma^2}\sum_{i=1}^{n} w_{i,l}\left(\left[x_{i,p} - \sum_{k=1}^{p-1} x_{i,k}\alpha_k\mu_k\right]^2 + \sum_{k=1}^{p-1} x_{i,k}^2\left[\alpha_k(\mu_k^2 + s_k^2) - \alpha_k^2\mu_k^2\right]\right),\end{aligned} \tag{24}$$

$$\mathbb{E}_q \log q(\beta, \gamma; \phi_{l,j}) = \sum_{k=1}^{p-1} -\frac{\alpha_k}{2}\log(2\pi s_k^2) - \frac{\alpha_k}{2} + \alpha_k \log(\alpha_k) + (1-\alpha_k)\log(1-\alpha_k). \tag{25}$$

Taking the derivatives of the ELBO with respect to the k-th coordinate of the variational parameters μ_k, α_k, s_k^2 gives the coordinate ascent variational updates to be

$$s_k^2 = \sigma^2\left(\frac{1}{\sigma_\beta^2} + \sum_{i=1}^{n} x_{i,k}^2 w_{i,l}\right)^{-1}, \tag{26}$$

$$\log\left(\frac{\alpha_k}{1-\alpha_k}\right) = \log\left(\frac{\xi}{1-\xi}\right) + \frac{\mu_k^2}{2s_k^2} + \log\frac{s_k}{\sigma\sigma_\beta}, \tag{27}$$

$$\mu_k = \frac{s_k^2}{\sigma^2}\sum_{i=1}^{n}\left\{w_{i,l}x_{i,k}\left(x_{i,p} - \sum_{m=1}^{p-1} x_{i,m}\mu_m\alpha_m\mathbb{1}(m \neq k)\right)\right\}. \tag{28}$$

A.2 2-step Approach for Density Estimation

Here, we describe the 2-step approach for density estimation used to select the bandwidth hyperparameters τ, as discussed in Section 2.3.1. Let $z_1, z_2, \ldots, z_n \in \mathbb{R}^q$, and consider the k-th coordinate of this data, $z_{j,k}$, with sample variance s_k^2. We choose the initial bandwidth estimate σ_k^2 using Silverman's rule of thumb [Silverman 2017] as

$$\sigma_k^2 = \left(0.9\min\left(s_k, \frac{IQR(z_{1,k}, z_{2,k}, \ldots, z_{n,k})}{1.35}\right)n^{-\frac{1}{5}}\right)^2. \tag{29}$$

Next, we let $\psi^0_{j,k}$ be the density of a Gaussian centered at $z_{j,k}$ with variance σ^2_k. Then, the estimate of the density for the k-th coordinate of the data is given by

$$\phi^0_k(x) = n^{-1} \sum_{j=1}^{n} \psi^0_{j,k}(x). \tag{30}$$

Repeating this for each coordinate $k = 1, 2, \ldots, q$, we will obtain a collection of q Gaussian mixture densities $\phi^0_1, \phi^0_1, \ldots, \phi^0_q$. To update this collection to a single multivariate density, first let let $\mathcal{H}$ be the harmonic mean of the σ^2_k, defined as

$$\mathcal{H} = n \left(\sum_{k=1}^{q} \frac{1}{\sigma^2_k} \right)^{-1}. \tag{31}$$

Then, let Ψ_j be a multivariate Gaussian centered at z_j with variance $\tau_j \mathbb{I}^{q \times q} \in \mathbb{R}^{q \times q}$, where $\tau_j \in \mathbb{R}$ is defined as

$$\tau_j = \frac{\mathcal{H}}{\sqrt{\prod_{k=1}^{q} \phi^0_k(z_{j,k})}}. \tag{32}$$

The final KDE to the multivariate density of the data is given by

$$\Phi(x) = n^{-1} \sum_{j=1}^{n} \Psi_j(x). \tag{33}$$

However, for our purposes, we simply use τ_j as the bandwidth for the j-th observation.

B BASELINE DETAILS

B.1 `mgm` hyperparameters

As discussed in Section 3.1, we select the bandwidth hyperparameter for the `tvmgm` function following this example provided by the authors of the package:
https://github.com/jmbh/mgmDocumentation/blob/22af97a621a1a240b68d5731886d542fb588915a/examples_tvmgm.R#L31
In this example, 5-fold cross validation is applied to time-varying Gaussian data with $n = 67$ and $p = 150$ to select the bandwidth out of the bandwidth candidates given by 0.1, 0.2, 0.3, 0.4.

B.2 Dimensionality Reduction Sorting Algorithm

Here, we describe the sorting algorithm for reducing the dimensionality of the extraneous covariate that we discuss in Section 3.1. Let $z_1, z_2, \ldots, z_n \in \mathbb{R}^q$ which we will sort to $z_{(1)}, z_{(2)}, \ldots, z_{(n)}$. First, we let $z_{(1)} = z_1$. Then, for $t \in \{2, 3, \ldots, n\}$, we define $\mathcal{S}^t = \{z_1, z_2, \ldots, z_n\} \setminus \{z_{(1)}, z_{(2)}, \ldots, z_{(t)}\}$ as the covariates that have not yet been sorted and set

$$z_{(t)} = \arg\min_{z \in \mathcal{S}^t} \|z_{(t)} - z\|. \tag{34}$$

This is a greedy algorithm for identifying a Hamiltonian path through the extraneous covariates. We then define the time index t_k for observation k as the position to which z_k was sorted. That is, $t_k = t$ if, and only if, $z_k = z_{(t)}$.

Table 4. Comparison of hyperparameter specification strategies in the $q = 1$ settings. Here, we set `max_iter_grid` = `max_iter` so that all strategies perform the same number of CAVI iterations.

p	HP Method	Sensitivity(%)	Specificity(%)	Time (seconds)
10	`grid_search`	**90.67**(5.79)	99.23(0.78)	**18.61**(3.70)
	`hybrid`	89.93(5.51)	**99.36**(0.83)	18.78(3.73)
	`model_average`	88.97(6.19)	99.01(0.78)	20.93(3.71)
25	`grid_search`	**87.42**(6.82)	99.69(0.24)	**58.84**(10.81)
	`hybrid`	86.61(6.86)	**99.76**(0.20)	60.40(11.63)
	`model_average`	85.73(7.00)	99.44(0.23)	62.92(10.96)
50	`grid_search`	**84.37**(6.89)	99.75(0.10)	171.72(19.24)
	`hybrid`	83.90(6.81)	**99.81**(0.08)	172.01(17.41)
	`model_average`	84.33(5.94)	99.60(0.09)	**166.94**(15.87)
100	`grid_search`	80.51(6.72)	99.80(0.05)	981.12(99.68)
	`hybrid`	80.66(6.50)	**99.86**(0.04)	976.91(85.36)
	`model_average`	**83.16**(5.85)	99.70(0.04)	**961.26**(67.73)

C EXTENDED EXPERIMENTS AND RESULTS

C.1 Full Grid Search

As discussed in Section 3.3.1, `model_average` incurs additional overhead relative to the alternatives due its inability to perform a cursory initial sweep of Θ as `hybrid` and `grid_search`. To assess whether abbreviating this search has a significant impact on performance, we repeat the experiments in Section 3.3.1, but set `max_iter_grid` = `max_iter` for `hybrid` and `grid_search` so that all methods perform CAVI for the same number of iterations.

We present results for this experiment in Table 4. We observe that aside from the expected increase in runtime to be roughly equivalent with `model_average`, the effect on sensitivity and specificity relative to the results presented in Table 1 is negligible.

C.2 Median/IQR-Aggregated Results

As discussed in Section 3.3.2, we observed the runtimes for `JGL` to be right-skewed due to some values of the hyperparameters causing the algorithm to iterate for an abnormally long time in a subset of the trials. Here, we aggregate the results from the experiments performed in Section 3.3.2 and Section 3.4 using median in place of mean and interquartile range in place of standard deviation.

We present results for the $q = 1$ settings in Table 5 and results for the $q = 2$ settings in Table 6. Notably, the median sensitivity of `JGL` in the $q = 1, p = 25$ setting is better than that of `covdepGE`, as is the median runtime in the $q = 1, p = 50$ setting. However, the ordering of the methods remains the same as for the mean-aggregated results in all other settings and metrics.

C.3 `Mclust` Subgroup Counts

As discussed in Section 3.4, we observed that in the $q = 2, p = 100$ setting, `JGL` no longer offered a better runtime than `covdepGE`. We show here that this is likely a result of increased computation due to a greater number of subgroups

Table 5. Comparison of heterogeneous graphical modeling methods in the $q = 1$ settings, presented as "*median* (*iqr*)".

p	Method	Sensitivity(%)	Specificity(%)	Time (seconds)
10	covdepGE	**91.71**(9.10)	99.56(0.84)	**5.50**(0.54)
	JGL	85.71(13.05)	98.72(1.57)	18.86(24.68)
	mgm	78.19(6.05)	**100.00**(0.02)	759.11(15.14)
25	covdepGE	83.81(11.52)	99.82(0.26)	**14.19**(1.59)
	JGL	**84.38**(13.38)	99.77(0.36)	23.34(39.35)
	mgm	73.71(9.90)	**100.00**(0.00)	2033.92(40.81)
50	covdepGE	**82.48**(9.43)	99.81(0.09)	44.63(2.42)
	JGL	72.19(22.43)	99.95(0.06)	**36.41**(83.18)
	mgm	68.38(14.86)	**100.00**(0.00)	4544.80(187.94)
100	covdepGE	**80.76**(7.00)	99.86(0.05)	235.50(15.63)
	JGL	71.81(22.90)	99.98(0.02)	**62.87**(212.55)
	mgm	60.29(10.14)	**100.00**(0.00)	11038.74(336.10)

Table 6. Comparison of heterogeneous graphical modeling methods in the $q = 2$ settings, presented as "*median* (*iqr*)".

p	Method	Sensitivity(%)	Specificity(%)	Time (seconds)
10	covdepGE	**93.71**(10.48)	**99.70**(0.76)	5.59(0.46)
	JGL	74.95(12.19)	98.63(1.89)	43.81(8.07)
	mgm	70.29(33.10)	99.56(0.71)	753.17(18.46)
	covdepGE_time	90.48(11.00)	98.89(0.96)	**5.47**(0.60)
25	covdepGE	**88.10**(21.76)	99.91(0.14)	**14.38**(1.71)
	JGL	65.05(18.24)	99.86(0.25)	61.67(17.03)
	mgm	54.19(17.95)	**99.98**(0.08)	2027.88(40.53)
	covdepGE_time	82.95(19.57)	99.68(0.29)	14.56(2.00)
50	covdepGE	**84.86**(12.38)	99.96(0.03)	**43.72**(2.33)
	JGL	55.43(21.62)	99.98(0.04)	111.58(17.31)
	mgm	42.86(9.48)	**100.00**(0.01)	4514.89(130.81)
	covdepGE_time	79.43(17.57)	99.80(0.09)	45.69(2.62)
100	covdepGE	**82.38**(17.76)	99.98(0.01)	**225.31**(20.42)
	JGL	51.52(17.57)	99.99(0.01)	297.56(59.60)
	mgm	42.86(0.43)	**100.00**(0.00)	11003.13(288.84)
	covdepGE_time	69.62(23.52)	99.85(0.05)	236.93(22.07)

being estimated by `Mclust`. Because `JGL` jointly estimates the precision structure in each of the subgroups, a greater number of subgroups results in longer runtimes.

In all trials, we choose the number of subgroups K by minimizing the BIC over $K \in \{2, 3, \ldots, 12\}$. In Figure 3, we visualize the empirical distributions of the optimal K conditioned on q. Here, we have marginalized across p as $\mathbf{Z}$, the variable partitioned by `Mclust`. $\mathbf{Z}$, is independent of p.

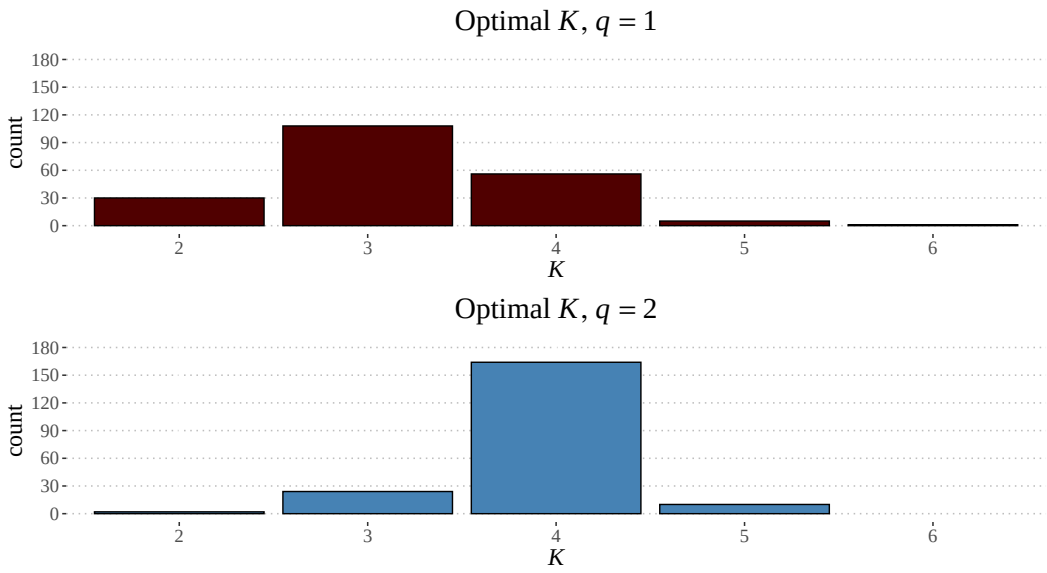

Fig. 3. Empirical distribution of optimal number of subgroups K for partitioning the covariate $\mathbf{Z}$. Because $\mathbf{Z}$ is independent of p, we marginalized the visualized distributions across trials and p. *Top*: $q = 1$ settings *Bottom*: $q = 2$ settings