

---

# GmGM: a Fast Multi-Axis Gaussian Graphical Model

---

**Bailey Andrew**  
University of Leeds

**David R. Westhead**  
University of Leeds

**Luisa Cuttillo**  
University of Leeds

## Abstract

This paper introduces the Gaussian multi-Graphical Model, a model to construct sparse graph representations of matrix- and tensor-variate data. We generalize prior work in this area by simultaneously learning this representation across several tensors that share axes, which is necessary to allow the analysis of multimodal datasets such as those encountered in multi-omics. Our algorithm uses only a single eigendecomposition per axis, achieving an order of magnitude speedup over prior work in the ungeneralized case. This allows the use of our methodology on large multi-modal datasets such as single-cell multi-omics data, which was challenging with previous approaches. We validate our model on synthetic data and five real-world datasets.

## 1 INTRODUCTION

A number of modern applications require the estimation of networks (graphs) exploring the dependency structures underlying the data. In this paper, we propose a new approach for estimating conditional dependency graphs. Two datapoints  $x, y$  are *conditionally independent* (with respect to a dataset  $\mathcal{D}$ ) if knowing one provides no information about the other that is not already contained in the rest of the dataset:  $\mathbb{P}(x|y, \mathcal{D}_{\setminus xy}) = \mathbb{P}(x|\mathcal{D}_{\setminus xy})$ . For normally distributed data, conditional dependencies are encoded in the inverse of the covariance matrix (the ‘precision’ matrix). Two datapoints are conditionally dependent on each other if and only if their corresponding element in the precision matrix is not zero. If our dataset was in the form of a vector  $\mathbf{d}$ , we could then model it

as  $\mathbf{d} \sim \mathcal{N}(\mathbf{0}, \Psi^{-1})$  for precision matrix  $\Psi$ . This is a Gaussian Graphical Model (GGM);  $\Psi$  encodes the adjacency matrix of the graph.

However, datasets are often more structured than vectors. For example, single-cell RNA sequencing datasets (scRNA-seq) come in the form of a matrix of gene expression counts whose rows are cells and columns are genes. Video data naturally requires a third-order tensor of pixels to represent it - rows, columns, and frames. Furthermore, multi-omics datasets, such as those including both scRNA-seq and scATAC-seq, may require two or more matrices to be properly represented; one for each modality.

We could assume that each row of our matrix is an i.i.d. sample drawn from our model. However, independence is a strong and often incorrect assumption. If we wanted to make no independence assumptions, we could vectorize the dataset  $\mathcal{D}$  and estimate  $\Psi$  in  $\text{vec}[\mathcal{D}] \sim \mathcal{N}(\mathbf{0}, \Psi^{-1})$ . Unfortunately, this produces intractably large  $\Psi$ , whose number of elements is quadratic in the product of the lengths of our dataset’s axes: the size of  $\Psi$  is  $O(\prod_{\ell} d_{\ell}^2)$  (throughout this paper,  $d_{\ell}$  will denote the size of the  $\ell$ th axis of a tensor).

Thankfully, tensors are highly structured, and we are often interested in the dependency structure of each axis individually - i.e. the dependencies between cells and the dependencies between genes, in the case of scRNA-seq - rather than the dependencies between the elements of the tensor themselves. To model this, we can represent  $\Psi$  as some deterministic combination of the axis-wise dependencies:  $\text{vec}[\mathbf{D}] \sim \mathcal{N}(\mathbf{0}, \zeta(\Psi_{\text{row}}, \Psi_{\text{col}})^{-1})$ , for some function  $\zeta$ . The strategy is to estimate  $\Psi_{\text{row}}$  and  $\Psi_{\text{col}}$  directly, without computing the intractable  $\zeta(\Psi_{\text{row}}, \Psi_{\text{col}})^{-1}$ . While there are multiple choices for  $\zeta$ , this paper considers only the Kronecker sum. Choices of  $\zeta$ , and their definitions, will be given in Section 2.

### 1.1 Notation

When possible, this paper follows the notation of prior work in multi-axis methods, such as Kalaitzis et al., 2013 and Greenewald et al., 2019. For tensor-specific

notation, we typically adhere to Kolda and Bader, 2009. Some changes have been made due to inconsistencies or conflicts.

The subscript  $\ell$  will always be used to denote an arbitrary axis.  $d_\ell$  and  $\Psi_\ell$  represent the size and precision matrix for the axis  $\ell$ , respectively - as in prior work. It will be helpful to define the symbols  $d_{<\ell}$ ,  $d_{>\ell}$ ,  $d_{\setminus\ell}$ , and  $d_{\forall}$  to represent the product of the lengths of all axes before  $\ell$ , after  $\ell$ , excluding  $\ell$ , and overall, respectively.

Lowercase bold represents vectors ( $\mathbf{d}$ ), uppercase bold represents matrices ( $\mathbf{D}$ ), and uppercase calligraphic represents tensors ( $\mathcal{D}$ ). We will also be dealing with sets of tensors,  $\{\mathcal{D}^\gamma\}_\gamma$ ; each individual tensor is called a modality, and the superscript  $\gamma$  will always represent indexing a modality. The number of axes of a tensor  $\mathcal{D}^\gamma$  is denoted  $K^\gamma$ , and to describe that an axis  $\ell$  is one of the axes of a tensor  $\mathcal{D}^\gamma$ , we will write  $\ell \in \gamma$ . Axes can be part of several modalities. An important tensor-variate operation is matricization, which takes a  $d_1 \times \dots \times d_K$  tensor  $\mathcal{D}$  and outputs a  $d_\ell \times d_{\setminus\ell}$  matrix. It is denoted  $\text{mat}_\ell[\mathcal{D}]$ , and can be viewed as vectorizing each slice of  $\mathcal{D}$  along the  $\ell$ th axis.

We will make frequent use of the Kronecker sum, defined in terms of the Kronecker product. The Kronecker product of two matrices  $\Psi_{\ell_1}, \Psi_{\ell_2}$  is denoted  $\Psi_{\ell_1} \otimes \Psi_{\ell_2}$ , and is best defined by example;

$$\begin{aligned} \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \otimes \begin{bmatrix} 5 & 6 \end{bmatrix} &= \begin{bmatrix} 1 \times \begin{bmatrix} 5 & 6 \end{bmatrix} & 2 \times \begin{bmatrix} 5 & 6 \end{bmatrix} \\ 3 \times \begin{bmatrix} 5 & 6 \end{bmatrix} & 4 \times \begin{bmatrix} 5 & 6 \end{bmatrix} \end{bmatrix} \\ &= \begin{bmatrix} 5 & 6 & 10 & 12 \\ 15 & 18 & 20 & 24 \end{bmatrix} \end{aligned}$$

The Kronecker sum is  $\Psi_{\ell_1} \oplus \Psi_{\ell_2} = \Psi_{\ell_1} \otimes \mathbf{I}_{d_{\ell_2}} + \mathbf{I}_{d_{\ell_1}} \otimes \Psi_{\ell_2}$ . When the matrices  $\mathbf{A}, \mathbf{B}$  are adjacency matrices of graphs, the Kronecker sum has the interpretation as the Cartesian product of those graphs.

The ‘‘blockwise trace’’ operation defined by Kalaitzis et al. (2013) appears when calculating the gradient of the log-likelihood of KS-structured normal distributions (and is strongly related to the  $\text{proj}_{KS}$  operator in Greenewald et al., 2019). Let  $\mathbf{I}_a$  be the  $a \times a$  identity matrix, and let  $\mathbf{J}^{ij}$  be a matrix of zeros except at  $(i, j)$ , where it equals 1. Then, we denote the blockwise trace of a matrix  $\text{tr}_b^a$  and define it according to Line 1

$$\text{tr}_b^a[\mathbf{M}] = \left[ \text{tr} \left[ \mathbf{M} \left( \mathbf{I}_a \otimes \mathbf{J}^{ij} \otimes \mathbf{I}_b \right) \right] \right]_{ij} \quad (1)$$

The final needed concept is that of the Gram matrix  $\mathbf{S}_\ell^\gamma$ , defined as  $\mathbf{S}_\ell^\gamma = \text{mat}_\ell[\mathcal{D}^\gamma] \text{mat}_\ell[\mathcal{D}^\gamma]^T$ . In

the single-tensor (multi-axis) case, this is a sufficient statistic; all prior work first computes these matrices as the initial step in their algorithm, as do we. When there are multiple modalities, we consider the ‘effective Gram matrices’  $\mathbf{S}_\ell = \sum_{\gamma|\ell \in \gamma} \mathbf{S}_\ell^\gamma$ , as these fulfill the role of sufficient statistic in the multi-tensor case.

## 2 PRIOR WORK

The Graphical Lasso (Friedman et al., 2008) is the standard single-axis GGM. The model assumes one has  $n$  independent samples of  $d_1$ -length vectors, and seeks to estimate the  $d_1 \times d_1$  precision matrix  $\Psi$ .  $\Psi$  is estimated via Equation 2

$$\Psi = \underset{\Psi \succ 0}{\text{argmax}} \log \det \Psi - \text{tr}[\mathbf{S}\Psi] + \rho \|\Psi\|_1 \quad (2)$$

Without the regularizing  $\rho \|\Psi\|_1$  term, this represents the maximum likelihood estimate for  $\Psi$ ;  $\Psi \succ 0$  ensures positive definiteness. The regularization penalty is an L1 penalty over the elements of  $\Psi$  - typically, the diagonal elements are not included in this penalty, as they do not represent edges on the conditional dependency graph.

Our work considers the Kronecker-sum (multi-axis) Graphical Lasso, which was introduced by Kalaitzis et al. (2013). This sum is one choice of  $\zeta$  to combine the per-axis precision matrices into the precision matrix of the vectorized dataset,  $\text{vec}[\mathbf{D}] \sim \mathcal{N}(\mathbf{0}, (\Psi_{\text{row}} \oplus \Psi_{\text{col}})^{-1})$ . Other choices for  $\zeta$  have been considered, such as using the Kronecker product (Dahl et al., 2013; Tsiligkaridis et al., 2013), or the square of the Kronecker sum (Wang & Hero, 2021; Wang et al., 2020). Each method has its strengths; the benefits of a Kronecker sum structure are its interpretability as a graph product, stronger sparsity, convexity of the maximum likelihood, and its allowance of inter-task transfer.

Kalaitzis et al. (2013)’s BiGLasso model only worked for matrix-variate data, and was very slow to converge to a solution, in large part due to its non-optimal space complexity of  $O(d_1^2 d_2^2)$ . This prohibited its use on moderately sized datasets. Numerous modifications have been made to the algorithm to improve its speed and to achieve an optimal space complexity of  $O(d_1^2 + d_2^2)$ , such as scBiGLasso (Li et al., 2022), TeraLasso (Greenewald et al., 2019), and EiGLasso (Yoon & Kim, 2020). Of these TeraLasso is notable in that it allows an arbitrary number of axes, i.e.  $\zeta(\Psi_1, \dots, \Psi_k) = \Psi_1 \oplus \dots \oplus \Psi_k$ , like our proposed method. TeraLasso and EiGLasso, the fastest prior algorithms, both rely on computing an eigendecomposition every iteration.

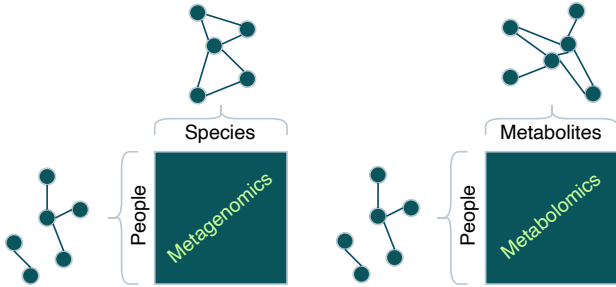


Figure 1: The two matrices of the LifeLines-DEEP dataset. As both matrices include data for the same people, the learned graph between people should be the same.

All of these models, including our own, rely on a normality assumption. We are most interested in the case of omics data, which is typically non-Gaussian; an overview of the use of GGMs in omics data is given by Altenbuchinger et al. (2020). For single-axis methods, the nonparanormal skeptic (Liu et al., 2012) can replace the normality assumption with a weaker nonparanormality assumption; Li et al. (2022) proposed a generalization of this approach to multi-axis methods, which is compatible with our approach. The nonparanormal skeptic often leads to better results on real datasets, for example see Figure 5a. For a thorough comparison of the nonparanormal skeptic, see the supplementary material.

In the single-axis case, several techniques exist for combining datasets with shared axes (Cai et al., 2016; Danaher et al., 2014; McCarter & Kim, 2014). However, none of these have been applied to the multi-axis case.

### 2.1 Unmet need

Many datasets, especially those in multi-omics, are representable as a collection of matrices or tensors. As a case study, we consider (a subset of) the Lifelines-DEEP dataset from Tigchelaar et al. (2015), which is summarized graphically in Figure 1.

In this dataset, two different modalities of data were gathered from the same people: counts of microbial species found in their stools (metagenomics) and counts of metabolites found in their blood plasma (metabolomics). While the metagenomics and metabolomics are different matrices, each modality shares an axis (‘people’). If we were to estimate a graph of people on each modality independently, they would likely yield different graphs. This is not ideal; if our aim is to estimate the true graph of conditional dependencies, there should be only one resultant graph. To estimate it, we should be considering both modal-

ities simultaneously to take advantage of the shared axis.

One way to do this would be to concatenate the modalities, producing a matrix of people by ‘species+metabolites’. This could yield interesting results, if one is interested in connections between individual species and a metabolite. However, it would vastly increase the size of the output graph, which grows quadratically in the length of the axis. Furthermore, it is not always possible; some datasets may not be concatenatable.

## 3 OUR CONTRIBUTIONS

We make three contributions to the field of multi-axis graphical models. Firstly, we produce an algorithm to estimate multi-axis graphical models an order of magnitude faster than existing work. This allows us to apply our algorithm on large datasets, such as single-cell RNA-sequencing datasets which include roughly 20,000 genes and tens of thousands of cells. Prior work could not handle these datasets in reasonable time - ours is the only method able to produce multi-axis estimates for data of this size, and runs comparably quickly to single-axis methods.

Secondly, we generalize the multi-axis model from prior work, which was only able to consider unimodal data, into a model that can handle multiple tensors with shared axes. This generalization is essential to the use of multi-axis methods on multi-modal data such as multi-omics; no multi-axis method had been made for this type of data before our work.

Finally, we produce a few helpful theorems to improve accuracy and further improve speed. A covariance-thresholding technique by Mazumder and Hastie (2012) allowed the partitioning of data into guaranteed-independent chunks of data in the single-axis case; we show how to lift this result to the multi-axis case, and demonstrate its efficacy (Figure 7). We also show how to efficiently include some natural prior distributions into our model, which allows us to demonstrate an extraordinary increase in performance on real-world data (Figure 6c).

### 3.1 The model

To properly handle sets of tensors, we propose modelling each tensor as being drawn independently from a Kronecker-sum normal distribution. If the tensors share an axis  $\ell$ , then they will still be drawn independently - but their distributions will be parameterized by the same  $\Psi_\ell$ . For an arbitrary set of tensors, the model is:

$$\mathcal{D}^\gamma \sim \mathcal{N}_{KS} \left( \{\Psi_\ell\}_{\ell \in \gamma} \right)$$

for  $\mathcal{D}^\gamma \in \{\mathcal{D}^\gamma\}_\gamma$

We call this model the ‘‘Gaussian multi-Graphical Model’’ (GmGM) as it extends Gaussian Graphical Models to estimate multiple graphs from a set of tensors. In this paper, we will make the assumption that no tensor in our set contains the same axis twice. Any tensor with a repeated axis would naturally be interpretable as a graph - such datasets are rare, and if one already has a graph then the need for an algorithm such as this is diminished.

As an example, we model the LifeLines-DEEP dataset  $\mathbf{D}^{\text{metagenomics}}$  and  $\mathbf{D}^{\text{metabolomics}}$  independently as:

$$\mathbf{D}^{\text{metagenomics}} \sim \mathcal{N}_{KS} \left( \Psi^{\text{people}}, \Psi^{\text{species}} \right)$$

$$\mathbf{D}^{\text{metabolomics}} \sim \mathcal{N}_{KS} \left( \Psi^{\text{people}}, \Psi^{\text{metabolites}} \right)$$

### 3.2 The algorithm

Here, we present an algorithm to compute the maximum likelihood estimate (MLE) jointly for all parameters  $\Psi_\ell$  of the GmGM. The general idea is to produce an analytic estimate for the eigenvectors of  $\Psi_\ell$ , and then iterate to solve for the eigenvalues; this is summed up graphically in Figure 2

**Theorem 1.** *Let  $\mathbf{V}_\ell \text{diag}[\mathbf{e}_\ell] \mathbf{V}_\ell^T$  be the eigendecomposition of  $\mathbf{S}_\ell$  (where  $\mathbf{V}_\ell \in \mathbb{R}^{d_\ell \times d_\ell}$  and  $\text{diag}[\mathbf{e}_\ell] \in \mathbb{R}^{d_\ell \times d_\ell}$  is a diagonal matrix with diagonal  $\mathbf{e}_\ell$ ). Then  $\mathbf{V}_\ell$  are the eigenvectors of the maximum likelihood estimate of  $\Psi_\ell$ .*

Theorem 1 is critical to allowing efficient estimation of  $\Psi_\ell$ , as it not only allows us to extract the computationally intensive eigendecomposition operation from the iterative portion of the algorithm, but also reduces the remaining number of parameters to be linear in the length of an axis.

It is now necessary to find estimate the eigenvalues  $\Lambda_\ell = \text{diag}[\lambda_\ell]$ . We will do this iteratively, which requires the gradient of the negative log-likelihood with respect to the eigenvalues. A sketch of the derivation of this is given.

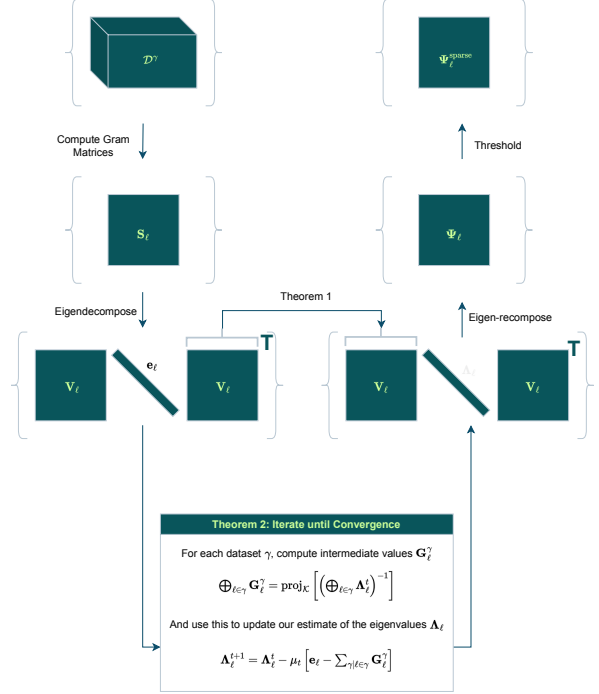


Figure 2: A graphical overview of how the GmGM algorithm works. We use  $\gamma$  to represent an arbitrary modality, and  $\ell$  to represent an arbitrary axis. Proofs are given in the supplementary material.

$$\frac{\partial \text{NLL}}{\partial \lambda_\ell^{(i)}} = \frac{1}{2} \frac{\partial}{\partial \lambda_\ell^{(i)}} \left( \text{tr} \left[ \mathbf{S}_\ell \mathbf{V}_\ell \Lambda_\ell \mathbf{V}_\ell^T \right] - \sum_\gamma \log \left| \bigoplus_{\ell' \in \gamma} \Lambda_{\ell'} \right| \right) \quad (3)$$

$$= \frac{1}{2} \frac{\partial}{\partial \lambda_\ell^{(i)}} \mathbf{e}_\ell \lambda_\ell - \frac{1}{2} \frac{\partial}{\partial \lambda_{\ell, (i)}} \sum_{\gamma, j} \log \left[ \bigoplus_{\ell' \in \gamma} \Lambda_{\ell'} \right]_{jj} \quad (4)$$

$$= \frac{1}{2} \mathbf{e}_{\ell, (i)} - \frac{1}{2} \sum_{\gamma, j} \frac{[\mathbf{I}_{d_{<\ell}} \otimes \mathbf{J}_{ii} \otimes \mathbf{I}_{d_{>\ell}}]_{jj}}{\left[ \bigoplus_{\ell' \in \gamma} \Lambda_{\ell'} \right]_{jj}} \quad (5)$$

$$\frac{\partial \text{NLL}}{\partial \lambda_\ell} = \frac{1}{2} \mathbf{e}_\ell - \frac{1}{2} \sum_\gamma \text{tr}_{d_{>\ell}}^{d_{<\ell}} \left[ \left( \bigoplus_{\ell' \in \gamma} \Lambda_{\ell'} \right)^{-1} \right] \quad (6)$$

Line 6 follows from the definition of the blockwise trace; the rest use Theorem 1 and standard algebra.

An L1 penalty is often included in graphical models to enforce sparsity. To enforce sparsity in our model, there are a few options; one can choose to either keep the top  $p\%$  of edges, or keep the top  $k$  edges per vertex. In Section 4 we see that this technique performs well. Including regularization in our algorithm is non-

Table 1: A comparison of the asymptotic complexity of TeraLasso and GmGM. EiGLasso is omitted because it has the same complexity as TeraLasso.  $N$  represents the number of iterations before convergence. ‘‘Simplified Time Complexity’’ represents the case when all axes have the same size  $d$ , there is only one modality, and the dataset only has two axes.

Algorithm	Simplified Time Complexity	Full Time Complexity	Memory Complexity
TeraLasso	$O(Nd^3)$	$O(\sum_{\ell} d_{\ell}^2 d_{\setminus \ell} + Nd_{\ell}^3 + N \prod_{\ell} d_{\ell})$	$O(\sum_{\ell} d_{\ell}^2)$
GmGM	$O(d^3 + Nd^2)$	$O(\sum_{\ell} d_{\ell}^2 d_{\setminus \ell} + d_{\ell}^3 + \sum_{\gamma} N \prod_{\ell \in \gamma} d_{\ell})$	$O(\sum_{\ell} d_{\ell}^2)$

trivial, as we rely on analytic estimates of the eigenvectors. However, by utilizing a restricted form of the L1 penalty, we can bypass this limitation. For details on how to do so, see Section 5 in the supplementary material - we will refer to GmGM equipped with this penalty as ‘GmGM L1’. In short, the restricted L1 penalty is an L1 penalty on the off-diagonals of a matrix (i.e. the Graphical Lasso) that has given (fixed) eigenvectors (since the iterative portion of our algorithm only iterates over the eigenvalues). We incorporate it with subgradient descent.

### 3.3 Additional Theorems

We can incorporate priors into our algorithm as well. This is nontrivial, as we want it to be in a manner compatible with Theorem 1.

**Theorem 2** (GmGM Estimator with Priors). *Suppose we have the same setup as in Theorem 1, and that we have a prior of the form:*

$$\prod_{\ell} g_{\ell}(\Theta_{\ell}) e^{\text{tr}[\eta_{\ell}(\Theta_{\ell})^T \Psi_{\ell}] + h_{\ell}(\Psi_{\ell})} \quad (7)$$

*In other words, our prior is an exponential distribution for  $\Psi_{\ell}$ , in which  $\Psi_{\ell}$  is the sufficient statistic.*

*Then, if  $h_{\ell}$  depends only on the eigenvalues (i.e. it is ‘unitarily invariant’), the eigenvectors of  $\frac{1}{2}\mathbf{S}_{\ell} - \eta_{\ell}(\Theta_{\ell})$  are the eigenvectors  $\mathbf{V}_{\ell}$  of the MAP estimate for  $\Psi_{\ell}$ . Furthermore, if  $h_{\ell}$  is convex, then a result by Lewis (1996) allows us perform gradient descent for the eigenvalues in an almost-unaltered way. (see supplementary material).*

$\frac{1}{2}\mathbf{S}_{\ell} - \eta_{\ell}(\Theta_{\ell})$  can be thought of as our ‘‘priorized’’ effective Gram matrix; its eigenvalues and eigenvectors play the same role as those of  $\mathbf{S}_{\ell}$  in the original algorithm. The required property of unitary invariance is very common - both the Wishart (arguably the most natural prior to use in this context) and matrix gamma distributions satisfy Theorem 2. The benefits of including priors are demonstrated in Figure 6c.

We also show that the covariance thresholding trick

(Mazumder & Hastie, 2012) works in the multi-axis case. This allows us to partition the dataset into guaranteed-to-be-disconnected subsets before using our algorithm (although we do not make use of this theorem when comparing performance to other work, as it equally benefits all methods). The benefits of using this trick are shown in Figure 7.

**Theorem 3.** *Set all elements of  $\mathbf{S}_{\ell}$  whose absolute value is less than  $\rho$  to 0. This encodes a potentially disconnected graph. Likewise, consider  $\Psi_{\ell}$  to be the estimated precision matrix for our model equipped with an L1 penalty of strength  $\rho$ . This also encodes a potentially disconnected graph. If we label the vertices by which disconnected component they are part of, then this labeling is the same in both procedures (the procedure with  $\mathbf{S}_{\ell}$  and the procedure with  $\Psi_{\ell}$ ).*

### 3.4 Convexity and Uniqueness

As can be observed in Line 4 the negative log-likelihood, as a function of the eigenvalues, can be expressed as the sum of a dot product (an affine, and hence convex, function) and negative logarithms, which are also convex. Thus, the objective is convex.

The Kronecker-sum model, without additional restrictions, has non-identifiable diagonals. Greenewald et al. (2019) show that this can be solved by projecting the gradients at each iteration. Letting  $\mathbf{A}_{\ell} = \text{tr}[\bigoplus_{\ell \in \gamma} \Lambda_{\ell}]$ , their projection is defined according to Line 8. See Sections D and I.3 in their paper for thorough information. We apply this projection to our algorithm as well.

$$\text{proj}_{\mathcal{K}} = \mathbf{A}_{\ell} - \frac{K_{\gamma} - 1}{K_{\gamma}} \frac{\text{tr}[\mathbf{A}_{\ell}]}{d_{\ell}} \mathbf{I}_{d_{\ell}} \quad (8)$$

## 4 RESULTS

We tested our algorithm on synthetic data and five real-world datasets. For synthetic data, we generated Erdos-Renyi graphs as our ground truth precision matrices for each axis (except for Figure 4b which uses an AR(1) process). Datasets were then generated from

the Kronecker-sum normal distribution using these precision matrices. In all cases, we only generated 1 sample from this distribution. For comparison, we set EiGLasso’s  $K$  parameter to 1; this parameter corresponds to the fastest version of EiGLasso. We used the same convergence tolerance and max iterations for all algorithms. Time and memory complexity for GmGM, TeraLasso, and EiGLasso are given in Table 1. When measuring runtimes on synthetic data, we let the regularization parameters be zero for prior work.

#### 4.1 Synthetic Data

We verified that our algorithm was indeed faster on matrix-variate data compared to prior work (Figure 3) on our computer (2020 MacBook Pro with an M1 chip and 8GB RAM). Our results on matrix data are encouraging - datasets of size 4,000 by 4,000 could have their graphs estimated in a minute, and, extrapolating the runtimes, datasets up to size 16,000 by 16,000 could have their graphs estimated in about an hour. This is significantly faster than our baselines; neither EiGLasso nor TeraLasso could compute on 1,500 by 1,500 datasets in a minute. Larger datasets would require more than 6GB of memory for our algorithm to run, pushing the limits of RAM. On higher-order tensor data, our algorithm continues to outperform prior work. For 4-axis and higher data, our algorithm could handle every dataset that could fit in RAM in less than a minute (Fig 3c and supplementary material).

In addition to these speed improvements, we show that we perform similarly to state-of-the-art (Figure 4). We demonstrate that taking into account shared axes does indeed improve performance (Figure 4a), and furthermore that our restricted L1 penalty leads to near-perfect performance on tensor data (Figure 4c).

#### 4.2 Real Data

We tested our method on various real datasets. These include two video datasets (COIL-20 (Nene et al., n.d.) and EchoNet-Dynamic (Ouyang et al., 2020)), a transcriptomics dataset (E-MTAB-2805 (Buettner et al., 2015)), and two multi-omics datasets (LifeLines-DEEP (Tigchelaar et al., 2015) and a 10x Genomics dataset (10x Genomics, 2021)). Due to space concerns, we only briefly describe each experiment and its results here. Full details are available in the supplementary material.

For the COIL-20 dataset, we tested whether our algorithm could be used to reconstruct the frames of a shuffled video. We chose this test as it was the test used in the first Kronecker sum multi-axis model, BiGLasso. BiGLasso had to heavily downsample the image (to a 9x9 image with half the frames), and flatten the rows

and frames into a single axis. Due to the speed improvements of our algorithm, and its ability to handle tensor-variate data, we were able to run our algorithm on the full-sized dataset and achieve a similar result in negligible time. Specifically, the reconstruction of the rows, columns, and frames all had an accuracy of 99%. In Figure 5b we show the shuffled data, with Figure 5c being the reconstruction using our algorithm.

Due to the simplicity of COIL-20, we chose to do a similar test on the more complicated EchoNet-Dynamic dataset. We expected there to be structure in our predicted graphs, which would allow us to detect the pattern of heartbeats. This pattern can be seen in Figure 5d. We were largely successful in this endeavor, with only one of the five videos we tested having poor results. For full details on the experiment, and how the heartbeat was extracted, see the supplementary material.

To test GmGM on our intended use case, ‘omics data, we started with the E-MTAB-2805 transcriptomics dataset, as this dataset was used in the original scBiGLasso paper for exploratory analysis. In this dataset, each cell belongs to one of three classes (G, S, or G2M) depending on its stage in the cell cycle. To evaluate our algorithm’s performance, we use assortativity. Assortativity measures the tendency of cells in the same class to cluster together in the estimated graph; an assortativity of 0 represents no tendency for connection, with 1 being the maximum. Negative assortativites are possible, if there is a tendency to connect to different classes. We report the assortativities in Figure 5a, comparing our results to EiGLasso and with the nonparanormal skeptic applied to the input.

Our fourth dataset, the LifeLines-DEEP dataset, was chosen because it was multi-modal, and because a single-axis graphical model, ZiLN (Prost et al., 2021), had already been tested on one of its modalities. We use assortativity as a metric, as that is what was used in prior work. Our results are comparable to their work (Figure 6). Our algorithm’s runtime of 4.7 seconds was similar to ZiLN’s 3.6 seconds, showing that our multi-axis method was capable of similar runtimes as single-axis methods for the first time. ZiLN was specifically built for metagenomics data - the compositionality assumption it makes would not be correct for most other datasets considered here, and hence we do not evaluate its performance on other datasets. Likewise, TeraLasso and EiGLasso were too slow to evaluate assortativity curves as in Figure 6.

Finally, we tested our approach on a 10x Genomics single-cell (RNA+ATAC) dataset taken from a B cell lymphoma tumour. We chose this dataset because it is multi-modal and quite large - 14,000 cells, 20,000

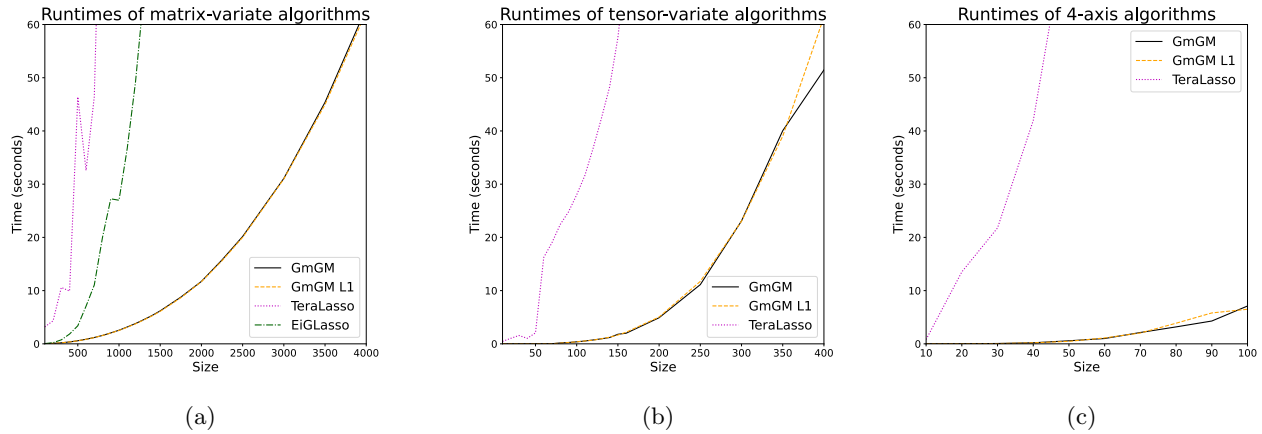


Figure 3: A comparison of the runtimes of our algorithm against (a) bi-graphical, (b) 3-axis, and (c) 4-axis prior work.

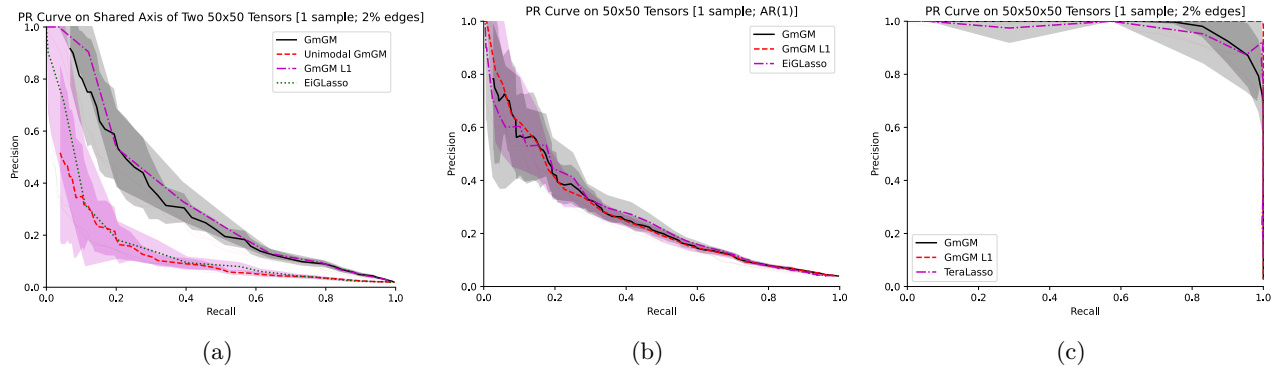


Figure 4: Precision-recall curves comparing various algorithms on synthetic 50x50 data, averaged over multiple runs. For (a) and (c), true graphs have each edge independently having a 2% chance of existing. For (b), true graphs are generated from an AR(1) process. Shaded background represents standard deviation. (a) Two 50x50 matrices with a shared axis. EiGLasso and ‘Unimodal GmGM’ only consider one of the matrices. (b) A single 50x50 matrix. (c) A single 50x50x50 tensor. With the restricted L1 penalty, our algorithm performs nearly perfectly in the tensor-variate case.

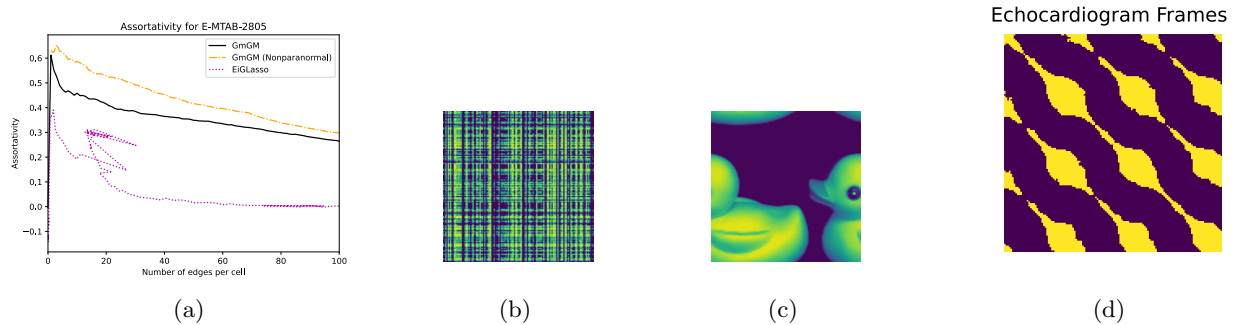


Figure 5: (a) Assortativity as we increase the number of edges in the final graph. EiGLasso data was generated by calculating the assortativity-per-regularization penalty, and calculating the edges-per-cell from this data afterwards; this is the cause for the curve’s strange behavior around 20 edges per cell. (b) The COIL-20 duck, in which the rows, columns and frames are shuffled. (c) GmGM does an almost perfect job at recovering the duck; unfortunately, it gets cut in half. (d) Predicted precision matrix for echocardiogram frames (yellow=connected, blue=disconnected). The periodic structure of the heartbeat is evident in the diagonals.

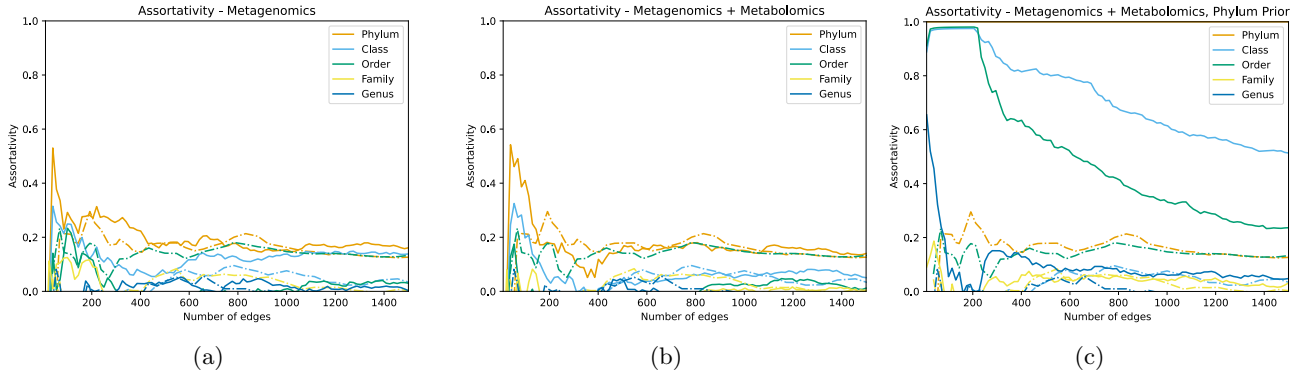


Figure 6: Assortativity in the LifeLines-DEEP dataset, comparing our method with the Zero-inflated Log-Normal (ZiLN) model. Solid lines represent our algorithm, dashed represent ZiLN. In one case we show the performance of our algorithm restricted to the metagenomics dataset (a) and when augmented with the metabolomics dataset (b). In both cases, ZiLN only has access to the metagenomics dataset, as it is a single-axis model. (c) The same test as (b), with the incorporation of phylum information as a prior (Theorem 2). Specifically, we used a Wishart prior whose parameter encoded a graph connecting two species if and only if they are in the same phylum.



Figure 7: (a) The precision-recall curves comparing vanilla GmGM with its performance using Theorem 3's covariance thresholding trick. (b) Runtime comparison of vanilla GmGM and the covariance thresholding trick. Times reported were averaged over 50 runs. Data was drawn from two disjoint versions of the same distribution as from Figure 4a to simulate there being multiple groups.

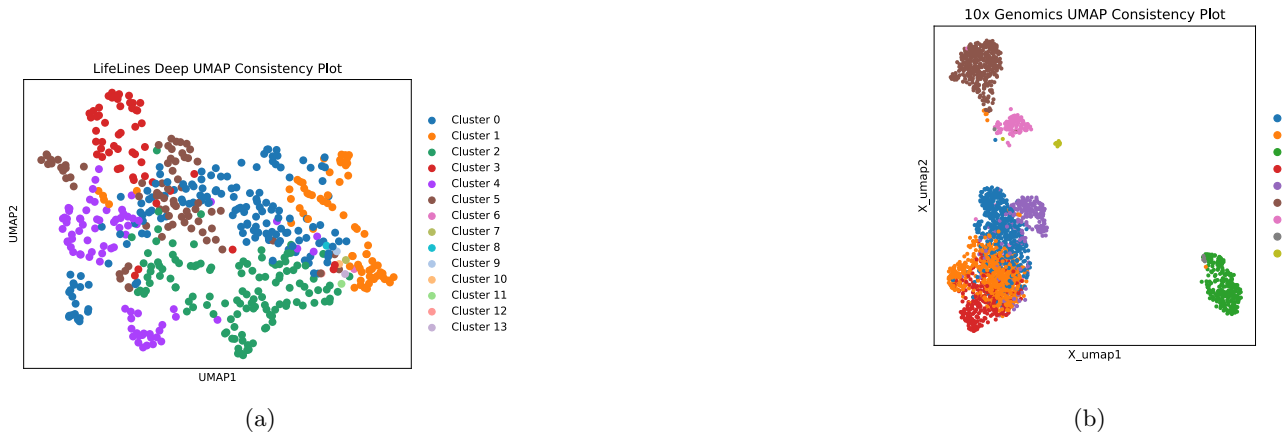


Figure 8: Both (a) and (b) are UMAP plots, colored by Louvain clustering of GmGM's resultant graphs. (a) Species from the LifeLines-DEEP dataset. (b) Cells from the 10x Genomics Dataset. In both cases, we can see that clusters on our graph correspond to contiguous regions of UMAP-space.



genes, and 100,000 peaks (although after preprocessing it became 2359 cells, 5350 genes, and 12485 peaks). This dataset is unlabelled, prohibiting more quantitative analyses; our main aim in this test was to prove that the algorithm could produce results in reasonable time. Our algorithm took 590 seconds to consider both modalities jointly and 52 seconds on just the scRNA-seq data. In contrast, EiGLasso took more than 60,000 seconds on the scRNA-seq data - due to its excessive runtime, we did not let EiGLasso finish.

To ensure our algorithm was producing sensible results on the 10x Genomics dataset, we performed a ‘UMAP consistency analysis’. Specifically, we ran Louvain clustering (Blondel et al., 2008) on our graph, and viewed the clusters in UMAP-space (McInnes et al., 2020). We believe this is a good test in this case, as nonlinear dimensionality reduction is used in almost every scRNA-seq study (Svensson et al., 2020). As we can see in Figure 8b, the clusters on our graph correspond to sensible regions in UMAP-space as well. In the supplementary material, we provide interpretations for some of these clusters. Figure 8a shows another UMAP consistency plot, for the LifeLines-DEEP dataset.

In the supplementary material, we give variants of the UMAP consistency plot for both the LifeLines-DEEP and 10x Genomics datasets. These variants use different clustering methods (Leiden (Traag et al., 2019) and Ensemble Clustering (Poulin & Th  berge, 2019)), or tSNE instead of UMAP (Maaten & Hinton, 2008). In all cases the results are broadly similar, except that Leiden on the LifeLines-Deep dataset tends to put everything in one cluster.

All of the code to run the algorithm and recreate the experiments has been made publicly available on GitHub: <https://github.com/AIStats-GmGM/GmGM> and we have a package ‘GmGM’ on PyPI.

## 5 LIMITATIONS AND FUTURE WORK

One limitation of our algorithm is its lack of compatibility with the standard L1 penalty. We do show that our restricted L1 penalty leads to better performance in synthetic data, even decisively outperforming L1-penalized prior work in some cases (Figure 4c). In the supplementary material, we also show that our restricted penalty leads to much better performance on the E-MTAB-2805 transcriptomics dataset. However, it would be nice to incorporate the standard L1 penalty, as it is a well-studied and well-accepted technique. This would be a rewarding avenue for future

work.

Another limitation of our algorithm is that it does not handle the case in which two axes partially overlap. For example, in the LifeLines-DEEP dataset, not every person is in both the metabolomics and metagenomics datasets. For now, our solution to this is to restrict ourselves to the subset of indices that are available in all modalities sharing that axis (this was about 90% of people in the LifeLines-DEEP dataset). Prior multi-axis methods could only handle a single modality, whereas this partial overlap problem only arises when considering multiple modalities - a case which we are the first to consider. We intend to relax this restriction in future work.

One notable effect of our algorithm’s speed improvements is that we are now limited by RAM rather than runtime; this affected our experiments, wherein we were more aggressive with our quality control filtering of the 100,000 ATAC peaks in the 10x dataset. Before our work, algorithms were limited by how long it would take to run rather than how much space was needed - our algorithm is the first to be fast enough to hit memory limits. Our algorithm has a theoretically optimal space usage of  $O(\sum_{\ell} d_{\ell}^2)$ , since the output is a set of  $d_{\ell} \times d_{\ell}$  precision matrices. However, given that we are often interested in *sparse* matrices, an avenue for future work would be to create variants of the algorithm that are capable of leveraging this sparsity. As our algorithm (along with the fastest prior work, EiGLasso and TeraLasso) depends on eigendecompositions, this would be a nontrivial task.

## 6 CONCLUSION

We have created a novel model, GmGM, which successfully generalizes Gaussian graphical models to the common scenario of multi-modal datasets. Furthermore, we demonstrated that our algorithm is significantly faster than prior work focusing on Gaussian multi-graphical models while still preserving state-of-the-art performance. These improvements allow multi-graphical models to be applied to datasets with axes of length in the tens of thousands. Additionally, we showed how to integrate a couple natural priors into our model, as well as provided a technique to partition datasets into smaller portions before analysis. The latter is particularly useful as runtime scales cubically after the computation of the empirical Gram matrices; partitioning a dataset in half provides an eightfold speedup. Finally, we demonstrated the application of our algorithm on five real-world datasets to prove its efficacy.

## Acknowledgements

We thank the participants and the staff of LifeLines-DEEP for their collaboration. Funding for the project was provided by the Top Institute Food and Nutrition Wageningen grant GH001. The sequencing was carried out in collaboration with the Broad Institute.

We also thank the reviewers for their comments - this paper has been much improved thanks to their feedback.

## References

- 10x Genomics. (2021, May). Flash-Frozen Lymph Node with B Cell Lymphoma (14k sorted nuclei). Retrieved May 10, 2023, from <https://www.10xgenomics.com/resources/datasets/fresh-frozen-lymph-node-with-b-cell-lymphoma-14-k-sorted-nuclei-1-standard-2-0-0>
- Altenbuchinger, M., Weihs, A., Quackenbush, J., Grabe, H. J., & Zacharias, H. U. (2020). Gaussian and Mixed Graphical Models as (multi)omics data analysis tools. *Biochimica et Biophysica Acta (BBA) - Gene Regulatory Mechanisms*, 1863(6), 194418. <https://doi.org/10.1016/j.bbagr.2019.194418>
- Blondel, V. D., Guillaume, J.-L., Lambiotte, R., & Lefebvre, E. (2008). Fast unfolding of communities in large networks [arXiv:0803.0476 [cond-mat, physics:physics]]. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10), P10008. <https://doi.org/10.1088/1742-5468/2008/10/P10008>
- Buettner, F., Natarajan, K. N., Casale, F. P., Prosperio, V., Scialdone, A., Theis, F. J., Teichmann, S. A., Marioni, J. C., & Stegle, O. (2015). Computational analysis of cell-to-cell heterogeneity in single-cell RNA-sequencing data reveals hidden subpopulations of cells [Number: 2 Publisher: Nature Publishing Group]. *Nature Biotechnology*, 33(2), 155–160. <https://doi.org/10.1038/nbt.3102>
- Cai, T. T., Li, H., Liu, W., & Xie, J. (2016). Joint Estimation of Multiple High-dimensional Precision Matrices. *Statistica Sinica*, 26(2), 445–464. <https://doi.org/10.5705/ss.2014.256>
- Dahl, A., Hore, V., Iotchkova, V., & Marchini, J. (2013, December). Network inference in matrix-variate Gaussian models with non-independent noise [arXiv:1312.1622 [stat]]. <https://doi.org/10.48550/arXiv.1312.1622>
- Danaher, P., Wang, P., & Witten, D. M. (2014). The joint graphical lasso for inverse covariance estimation across multiple classes. *Journal of the Royal Statistical Society. Series B, Statistical methodology*, 76(2), 373–397. <https://doi.org/10.1111/rssb.12033>
- Friedman, J., Hastie, T., & Tibshirani, R. (2008). Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3), 432–441. <https://doi.org/10.1093/biostatistics/kxm045>
- Greenewald, K., Zhou, S., & Hero III, A. (2019, September). Tensor Graphical Lasso (TeraLasso) [arXiv:1705.03983 [stat]]. Retrieved February 24, 2023, from <http://arxiv.org/abs/1705.03983>
- Kalaitzis, A., Lafferty, J., Lawrence, N. D., & Zhou, S. (2013). The Bigraphical Lasso [ISSN: 1938-7228]. *Proceedings of the 30th International Conference on Machine Learning*, 1229–1237. Retrieved February 24, 2023, from <https://proceedings.mlr.press/v28/kalaitzis13.html>
- Kolda, T. G., & Bader, B. W. (2009). Tensor Decompositions and Applications. *SIAM Review*, 51(3), 455–500. <https://doi.org/10.1137/07070111X>
- Lewis, A. S. (1996). Derivatives of Spectral Functions [Publisher: INFORMS]. *Mathematics of Operations Research*, 21(3), 576–588. <https://doi.org/10.1287/moor.21.3.576>
- Li, S., López-García, M., Lawrence, N. D., & Cutillo, L. (2022, March). Scalable Bigraphical Lasso: Two-way Sparse Network Inference for Count Data [arXiv:2203.07912 [cs, stat]]. Retrieved February 24, 2023, from <http://arxiv.org/abs/2203.07912>
- Liu, H., Han, F., Yuan, M., Lafferty, J., & Wasserman, L. (2012). The nonparanormal SKEPTIC. *Proceedings of the 29th International Conference on Machine Learning, ICML 2012*, 1415–1422. Retrieved October 9, 2023, from <https://pure.johnshopkins.edu/en/publications/the-nonparanormal-skeptic-4>
- Maaten, L. v. d., & Hinton, G. (2008). Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9(86), 2579–2605. Retrieved February 22, 2024, from <http://jmlr.org/papers/v9/vandermaaten08a.html>
- Mazumder, R., & Hastie, T. (2012). Exact Covariance Thresholding into Connected Components for Large-Scale Graphical Lasso. *Journal of machine learning research : JMLR*, 13, 781–794. Retrieved October 9, 2023, from <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4225650/>
- McCarter, C., & Kim, S. (2014). On Sparse Gaussian Chain Graph Models. *Advances in Neural Information Processing Systems*, 27. Re-

- rieved February 22, 2024, from [https://papers.nips.cc/paper\\_files/paper/2014/hash/81c650caac28cdefce4de5ddc18befa0-Abstract.html](https://papers.nips.cc/paper_files/paper/2014/hash/81c650caac28cdefce4de5ddc18befa0-Abstract.html)
- McInnes, L., Healy, J., & Melville, J. (2020, September). UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction [arXiv:1802.03426 [cs, stat]]. <https://doi.org/10.48550/arXiv.1802.03426>
- Nene, S. A., Nayar, S. K., & Murase, H. (n.d.). Columbia Object Image Library (COIL-20).
- Ouyang, D., He, B., Ghorbani, A., Yuan, N., Ebinger, J., Langlotz, C. P., Heidenreich, P. A., Harrington, R. A., Liang, D. H., Ashley, E. A., & Zou, J. Y. (2020). Video-based AI for beat-to-beat assessment of cardiac function [Number: 7802 Publisher: Nature Publishing Group]. *Nature*, 580(7802), 252–256. <https://doi.org/10.1038/s41586-020-2145-8>
- Poulin, V., & Théberge, F. (2019). Ensemble clustering for graphs: Comparisons and applications [Number: 1 Publisher: SpringerOpen]. *Applied Network Science*, 4(1), 1–13. <https://doi.org/10.1007/s41109-019-0162-z>
- Prost, V., Gazut, S., & Brüls, T. (2021). A zero inflated log-normal model for inference of sparse microbial association networks [Publisher: Public Library of Science]. *PLOS Computational Biology*, 17(6), e1009089. <https://doi.org/10.1371/journal.pcbi.1009089>
- Svensson, V., da Veiga Beltrame, E., & Pachter, L. (2020). A curated database reveals trends in single-cell transcriptomics. *Database*, 2020, baaa073. <https://doi.org/10.1093/database/baaa073>
- Tigchelaar, E. F., Zhernakova, A., Dekens, J. A. M., Hermes, G., Baranska, A., Mujagic, Z., Swertz, M. A., Muñoz, A. M., Deelen, P., Cénit, M. C., Franke, L., Scholtens, S., Stolk, R. P., Wijmenga, C., & Feskens, E. J. M. (2015). Cohort profile: LifeLines DEEP, a prospective, general population cohort study in the northern Netherlands: Study design and baseline characteristics [Publisher: British Medical Journal Publishing Group Section: Epidemiology]. *BMJ Open*, 5(8), e006772. <https://doi.org/10.1136/bmjopen-2014-006772>
- Traag, V. A., Waltman, L., & van Eck, N. J. (2019). From Louvain to Leiden: Guaranteeing well-connected communities [Number: 1 Publisher: Nature Publishing Group]. *Scientific Reports*, 9(1), 5233. <https://doi.org/10.1038/s41598-019-41695-z>
- Tsiligkaridis, T., Hero III, A. O., & Zhou, S. (2013). Convergence Properties of Kronecker Graphical Lasso Algorithms [arXiv:1204.0585 [stat]]. *IEEE Transactions on Signal Processing*, 61(7), 1743–1755. <https://doi.org/10.1109/TSP.2013.2240157>
- Wang, Y., & Hero, A. (2021). SG-PALM: A Fast Physically Interpretable Tensor Graphical Model [ISSN: 2640-3498]. *Proceedings of the 38th International Conference on Machine Learning*, 10783–10793. Retrieved February 27, 2023, from <https://proceedings.mlr.press/v139/wang21k.html>
- Wang, Y., Jang, B., & Hero, A. (2020, February). The Sylvester Graphical Lasso (SyGlasso) [arXiv:2002.00288 [cs, stat]]. <https://doi.org/10.48550/arXiv.2002.00288>
- Yoon, J. H., & Kim, S. (2020). EiGLasso: Scalable Estimation of Cartesian Product of Sparse Inverse Covariance Matrices [ISSN: 2640-3498]. *Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI)*, 1248–1257. Retrieved February 24, 2023, from <https://proceedings.mlr.press/v124/ho-yoon20a.html>

## Checklist

- For all models and algorithms presented, check if you include:
  - A clear description of the mathematical setting, assumptions, algorithm, and/or model. **Yes**; all proofs, and the pseudocode, are available in the supplementary material.
  - An analysis of the properties and complexity (time, space, sample size) of any algorithm. **Yes**; the time and space complexity are given in the supplementary material. All experiments, synthetic and real, were done with sample size=1 as that is the case in the real world when using this type of algorithm.
  - (Optional) Anonymized source code, with specification of all dependencies, including external libraries. **Yes**; the source code has been put onto GitHub under a custom-made anonymized account for this submission. All libraries, including their version numbers, are given in the README of that repository.
- For any theoretical claim, check if you include:
  - Statements of the full set of assumptions of all theoretical results. **Yes**
  - Complete proofs of all theoretical results. **Yes**; all proofs, and the pseudocode, are available in the supplementary material.

- (c) Clear explanations of any assumptions. **Yes**; all theorems give their specific assumptions within them, and our model is stated clearly in Section [3.1](#)
  - (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. **Not applicable**
3. For all figures and tables that present empirical results, check if you include:
- (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). **Yes**; in the GitHub repo linked in supplementary material
  - (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). **Yes**; in the supplementary material, we give details on thresholding techniques.
  - (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). **Yes**; for each real dataset and the synthetic data, details of the experiment not contained in the main paper are contained in the supplementary material.
  - (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). **Yes**; we stated our computer’s specs in Section [4.1](#)
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
- (a) Citations of the creator If your work uses existing assets. **Yes**; we cite the datasets used
  - (b) The license information of the assets, if applicable. **Yes**; All details about the datasets are available in the supplementary material
  - (c) New assets either in the supplemental material or as a URL, if applicable. **Not applicable**
  - (d) Information about consent from data providers/curators. **Yes**; All details about datasets are available in the supplementary material
  - (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. **Not applicable**
5. If you used crowdsourcing or conducted research with human subjects, check if you include:
- (a) The full text of instructions given to participants and screenshots. **Not applicable**
  - (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. **Not applicable**

---

# Supplementary Materials for: GmGM: a Fast Multi-Axis Gaussian Graphical Model

---

## OUTLINE OF THE SUPPLEMENTARY MATERIAL

1. [Notation](#)
2. [Proofs](#)
  - 2.1. [Permutations](#)
  - 2.2. [Derivation of the probability density function](#)
  - 2.3. [Gradient](#)
  - 2.4. [Maximum likelihood estimate for the eigenvectors](#)
  - 2.5. [Maximum likelihood estimate for the eigenvalues](#)
  - 2.6. [Incorporation of priors](#)
  - 2.7. [Covariance thresholding](#)
3. [Dependencies](#)
4. [Experiments](#)
  - 4.1. [Synthetic data](#)
  - 4.2. [COIL video](#)
  - 4.3. [EchoNet-Dynamic ECGs](#)
  - 4.4. [Mouse embryo stem cell transcriptomics](#)
  - 4.5. [LifeLines-DEEP metagenomics + metabolomics](#)
  - 4.6. [10x Genomics flash frozen lymph nodes](#)
5. [Regularization](#)
6. [Asymptotic complexity](#)
7. [Centering the data](#)

In the supplementary material, we give proofs of all theorems (Section 2), provide code, pseudocode, and computer specs (Section 3), give more informations on our experiments (Section 4), describe how we incorporated regularization (Section 5), and finally give an overview of the asymptotic complexity of ours and other algorithms (Section 6).

Lemmas 1 and 2 are used to derive the Kronecker sum normal distributions’s pdf, which has been known in prior work; we have included it for completeness. All other lemmas and theorems are novel.

## 1 NOTATION

In addition to the notation used in the main paper, we also introduce further notation to aid in the proofs. For working with tensors, Kolda and Bader (2009) proved to be an invaluable resource; we have borrowed their notation in most cases. The only exception is that we have chosen to denote the  $\ell$ -mode matricization of a tensor  $\mathcal{T}$  as  $\text{mat}_\ell[\mathcal{T}]$  rather than  $\mathcal{T}_{(\ell)}$ , to highlight its similarity to  $\text{vec}[\mathcal{T}]$  and free up the subscript for other purposes.

To keep track of lengths of axes, we define the following notation:

- $d_\ell^\gamma$  is the length of axis  $\ell$
- $d_{>\ell}^\gamma$  is the product of lengths of all axes after  $\ell$
- $d_{<\ell}^\gamma$  is the product of lengths of all axes before  $\ell$
- $d_{\setminus\ell}^\gamma$  is the product of lengths of all axes except for  $\ell$
- $d_{\forall}^\gamma$  is the product of lengths of all axes (i.e. the number of elements in  $\mathcal{D}^\gamma$ )
- $d_{\forall} = \sum_{\gamma} d_{\forall}^\gamma$  is the total number of elements across all datasets

In prior work,  $d_\ell$  has been used to represent the lengths of axes but  $m_\ell$  was used where we write  $d_{\setminus\ell}$  (such as in Greenewald et al. (2019)). As prior work also used  $\setminus\ell$  to represent leaving out the  $\ell$ th axis in other contexts (such as in Kalaitzis et al. (2013)), and the analogous definitions of  $d_{>\ell}$  and  $d_{<\ell}$  were convenient for use in proofs, we chose to introduce  $d_{\setminus\ell}$  as the variable to represent leave-one-out length products. By representing all of these related concepts with similar symbols, we hope the maths will be easier to parse.

We will let  $\mathbf{I}_a$  be the  $a \times a$  identity matrix, which allows a concise definition of the Kronecker sum:  $\bigoplus_{\ell} \Psi_{\ell} = \sum_{\ell} \mathbf{I}_{d_{<\ell}} \otimes \Psi_{\ell} \otimes \mathbf{I}_{d_{>\ell}}$ .

We make frequent use of the vectorization  $\text{vec}[\mathbf{M}]$  of a matrix  $\mathbf{M}$ , and more generally of a tensor  $\text{vec}[\mathcal{T}]$ . We adopt the rows-first convention of vectorization, such that:

$$\text{vec} \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = [1 \quad 2 \quad 3 \quad 4] \tag{1}$$

While columns-first is more common, rows-first is more natural when we adopt the convention that rows are the first axis of tensor; this is the convention that matricization uses, and matricization is much more important for our work due to its role in defining the Gram matrices. Note that, for matrices, a rows-first vectorization of  $\mathbf{M}$  is equivalent to a columns-first vectorization of  $\mathbf{M}^T$ , so there is no fundamental difference between the two. For vectorizing a tensor, we proceed by stacking the rest of the axes in order, such that an element  $(i_1, \dots, i_K)$  in  $\mathcal{T}$  gets mapped to the element  $\sum_{\ell} i_{\ell} d_{<\ell}$  in  $\text{vec}[\mathcal{T}]$ .

We define the Gram matrices as  $\mathbf{S}_{\ell}^{\gamma} = \text{mat}_{\ell}[\mathcal{D}^{\gamma}] \text{mat}_{\ell}[\mathcal{D}^{\gamma}]^T$ . Typically we consider only the one-sample case. If you have multiple samples, indexed by a subscript  $i$ , then the Gram matrix becomes an average:  $\mathbf{S}_{\ell}^{\gamma} = \frac{1}{n} \sum_i^n \text{mat}_{\ell}[\mathcal{D}_i^{\gamma}] \text{mat}_{\ell}[\mathcal{D}_i^{\gamma}]^T$ .

An essential concept is that of the ‘stridewise-blockwise trace’, defined as:

$$\text{tr}_b^a [\mathbf{M}] = \left[ \text{tr} \left[ \mathbf{M} \left( \mathbf{I}_a \otimes \mathbf{J}^{ij} \otimes \mathbf{I}_b \right) \right] \right]_{ij} \quad (2)$$

Where  $\mathbf{J}^{ij}$  is the matrix of zeros except at  $(i, j)$  where it has a 1. It is a generalization of the blockwise trace considered by Kalaitzis et al. (2013), and is related to the  $\text{proj}_{\mathcal{K}}$  operation defined by Greenewald et al. (2019). Specifically,  $\text{proj}_{\mathcal{K}}[\mathbf{M}]$  is equivalent to  $\bigoplus_{\ell} \text{tr}_{d_{>\ell}}^{d_{<\ell}}[\mathbf{M}]$  up to an additive diagonal factor (Lemma 33 from Greenewald et al. (2019)).  $\text{proj}_{\mathcal{K}}[\mathbf{M}]$  was defined to be the matrix that best approximates  $\mathbf{M}$  (in terms of the Frobenius norm) while being Kronecker-sum-decomposable. This matrix is not unique; the choice by Greenewald et al. (2019) to include an additive factor was to enforce  $\text{tr}[\text{proj}_{\mathcal{K}}[\mathbf{M}]] = 0$ . We do not wish to enforce this constraint as it would be difficult to preserve in the multi-tensor case.

The parameter  $b$  of the stridewise-blockwise trace partitions the  $m \times m$  matrix  $\mathbf{M}$  into a block matrix with  $b \times b$  blocks of size  $(\frac{m}{b} \times \frac{m}{b})$ . The parameter  $a$  then partitions these blocks into a ‘strided’ matrix with  $a \times a$  strides containing  $\frac{m}{ab} \times \frac{m}{ab}$  blocks. We take the trace of each stride, and the final matrix is the matrix of these traces. As this is conceptually complicated, we provide an example.

$$\text{tr}_2^2 \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{bmatrix} \quad (3)$$

$$= \text{tr}^2 \begin{bmatrix} \text{tr} \begin{bmatrix} 1 & 2 \\ 1 & 2 \end{bmatrix} & \text{tr} \begin{bmatrix} 3 & 4 \\ 3 & 4 \end{bmatrix} & \text{tr} \begin{bmatrix} 5 & 6 \\ 5 & 6 \end{bmatrix} & \text{tr} \begin{bmatrix} 7 & 8 \\ 7 & 8 \end{bmatrix} \\ \text{tr} \begin{bmatrix} 1 & 2 \\ 1 & 2 \end{bmatrix} & \text{tr} \begin{bmatrix} 3 & 4 \\ 3 & 4 \end{bmatrix} & \text{tr} \begin{bmatrix} 5 & 6 \\ 5 & 6 \end{bmatrix} & \text{tr} \begin{bmatrix} 7 & 8 \\ 7 & 8 \end{bmatrix} \\ \text{tr} \begin{bmatrix} 1 & 2 \\ 1 & 2 \end{bmatrix} & \text{tr} \begin{bmatrix} 3 & 4 \\ 3 & 4 \end{bmatrix} & \text{tr} \begin{bmatrix} 5 & 6 \\ 5 & 6 \end{bmatrix} & \text{tr} \begin{bmatrix} 7 & 8 \\ 7 & 8 \end{bmatrix} \\ \text{tr} \begin{bmatrix} 1 & 2 \\ 1 & 2 \end{bmatrix} & \text{tr} \begin{bmatrix} 3 & 4 \\ 3 & 4 \end{bmatrix} & \text{tr} \begin{bmatrix} 5 & 6 \\ 5 & 6 \end{bmatrix} & \text{tr} \begin{bmatrix} 7 & 8 \\ 7 & 8 \end{bmatrix} \\ \text{tr} \begin{bmatrix} 1 & 2 \\ 1 & 2 \end{bmatrix} & \text{tr} \begin{bmatrix} 3 & 4 \\ 3 & 4 \end{bmatrix} & \text{tr} \begin{bmatrix} 5 & 6 \\ 5 & 6 \end{bmatrix} & \text{tr} \begin{bmatrix} 7 & 8 \\ 7 & 8 \end{bmatrix} \\ \text{tr} \begin{bmatrix} 1 & 2 \\ 1 & 2 \end{bmatrix} & \text{tr} \begin{bmatrix} 3 & 4 \\ 3 & 4 \end{bmatrix} & \text{tr} \begin{bmatrix} 5 & 6 \\ 5 & 6 \end{bmatrix} & \text{tr} \begin{bmatrix} 7 & 8 \\ 7 & 8 \end{bmatrix} \end{bmatrix} \quad (4)$$

$$= \text{tr}^2 \begin{bmatrix} 3 & 7 & 11 & 15 \\ 3 & 7 & 11 & 15 \\ 3 & 7 & 11 & 15 \\ 3 & 7 & 11 & 15 \end{bmatrix} \quad (5)$$

$$= \begin{bmatrix} \text{tr} \begin{bmatrix} 3 & 11 \\ 3 & 11 \end{bmatrix} & \text{tr} \begin{bmatrix} 7 & 15 \\ 7 & 15 \end{bmatrix} \\ \text{tr} \begin{bmatrix} 3 & 11 \\ 3 & 11 \end{bmatrix} & \text{tr} \begin{bmatrix} 7 & 15 \\ 7 & 15 \end{bmatrix} \end{bmatrix} \quad (6)$$

$$= \begin{bmatrix} 14 & 22 \\ 14 & 22 \end{bmatrix} \quad (7)$$

Notice the construction of the ‘strides’ in Line 6 - the parameter of 2 told us to take the trace of the most ‘spread out’ evenly spaced  $2 \times 2$  strided submatrix.

## 2 PROOFS

We will assume that no dataset contains repeated axes (i.e. no single tensor has two axes represented by the same graph), as this greatly affects the derived gradients. Shared axes - two tensors having one or more axes in

common - are allowed. The case of shared axes is, after all, the whole point of developing this extension to prior work.

## 2.1 Permutations

Note that both  $\text{vec} [\text{mat}_1 [\mathcal{D}^\gamma]]$  and  $\text{vec} [\text{mat}_\ell [\mathcal{D}^\gamma]]$  are row vectors containing the same elements, just in a different order. This means that there is a permutation matrix  $\mathbf{P}_{\ell \rightarrow 1}$  such that  $\text{vec} [\text{mat}_1 [\mathcal{D}^\gamma]]^T \mathbf{P}_{\ell \rightarrow 1} = \text{vec} [\text{mat}_\ell [\mathcal{D}^\gamma]]^T$ .

**Lemma 1** (Rearrangement lemma).  $\mathbf{P}_{\ell \rightarrow 1} (\mathbf{I}_{d_{<\ell}} \otimes \Psi_\ell \otimes \mathbf{I}_{d_{>\ell}}) \mathbf{P}_{\ell \rightarrow 1}^T = \Psi_\ell \otimes \mathbf{I}_{d_{\setminus \ell}}$

*Proof.* While  $\text{vec}$ ,  $\text{mat}_\ell$  and  $\otimes$  are defined as operations on matrices, for the purposes of permutations we can consider them as operations on indices. We can express them as follows:

$$\text{vec} : (i_1, \dots, i_K) \rightarrow \left( \sum_\ell i_\ell d_{<\ell} \right) \quad (8)$$

$$\text{mat}_\ell : (i_1, \dots, i_K) \rightarrow \left( i_\ell, \sum_{\ell' < \ell} i_{\ell'} d_{<\ell'} + \sum_{\ell' > \ell} i_{\ell'} \frac{d_{<\ell'}}{d_\ell} \right) \quad (9)$$

$$\otimes : ((i_1^1, i_1^2), \dots, (i_K^1, i_K^2)) \rightarrow \left( \sum_\ell i_\ell^1 d_{<\ell}, \sum_\ell i_\ell^2 d_{<\ell} \right) \quad (10)$$

We'll consider just the rows of  $\otimes$ ,  $\otimes_{rows}$  - although the same argument applies with columns:

$$\otimes_{rows} : (i_1^1, \dots, i_K^1) \rightarrow \left( \sum_\ell i_\ell^1 d_{<\ell} \right) \quad (11)$$

Finally, we'll introduce the permutation operation  $\sigma_{\ell \rightarrow 1}$  that will change the order of our Kronecker product:

$$\sigma_{\ell \rightarrow 1} : ((i_1^1, i_1^2), \dots, (i_K^1, i_K^2)) \rightarrow (((i_\ell^1, i_\ell^2), (i_1^1, i_1^2), \dots, (i_{\ell-1}^1, i_{\ell-1}^2), (i_{\ell+1}^1, i_{\ell+1}^2), \dots, (i_K^1, i_K^2)) \quad (12)$$

And again without loss of generality we restrict ourself to  $\sigma_{\ell \rightarrow 1}^{rows}$ :

$$\sigma_{\ell \rightarrow 1}^{rows} : (i_1^1, \dots, i_K^1) \rightarrow (i_\ell^1, i_1^1, \dots, i_{\ell-1}^1, i_{\ell+1}^1, \dots, i_K^1) \quad (13)$$

After a Kronecker product our indices are in the form  $\sum_\ell i_\ell d_{<\ell}$ , and if we were to reorder it with  $\sigma_{\ell \rightarrow 1}$  they would be in the form  $i_\ell + \sum_{\ell' < \ell} i_{\ell'} d_{<\ell'} d_\ell + \sum_{\ell' > \ell} i_{\ell'} d_{<\ell'}$ . Likewise, if we had matricized it we would have  $(i_\ell, \sum_{\ell' < \ell} i_{\ell'} d_{<\ell'} + \sum_{\ell' > \ell} i_{\ell'} \frac{d_{<\ell'}}{d_\ell})$ , which is vectorized to  $i_\ell + \sum_{\ell' < \ell} i_{\ell'} d_{<\ell'} d_\ell + \sum_{\ell' > \ell} i_{\ell'} d_{<\ell'}$ . These reorderings are the same, and hence the matrix that represents it is  $\mathbf{P}_{\ell \rightarrow 1}$ .

□



## 2.2 Derivation of the probability density function

Recall that the Kronecker-sum-structured normal distribution for a single tensor is defined as follows:

$$\text{vec}[\mathcal{D}^\gamma] \sim \mathcal{N}\left(\mathbf{0}, \left(\bigoplus_{\ell \in \gamma} \Psi_\ell\right)^{-1}\right) \iff \mathcal{D}^\gamma \sim \mathcal{N}_{KS}\left(\{\Psi_\ell\}_{\ell \in \gamma}\right) \quad (14)$$

The log-likelihood for this distribution is given in Kalaitzis et al. (2013) for the matrix case and Greenewald et al. (2019) for the general tensor case. However, neither of these papers provide a derivation. As the full derivation will motivate the construction of lemmas useful for the proofs of Theorems I we will give it here. First, observe that the density function is that of a normal distribution.

$$p(\mathcal{D}^\gamma) = \frac{\sqrt{\left|\bigoplus_{\ell \in \gamma} \Psi_\ell\right|}}{(2\pi)^{\frac{d_\gamma}{2}}} e^{-\frac{1}{2} \text{vec}[\mathcal{D}^\gamma]^T \left(\bigoplus_{\ell \in \gamma} \Psi_\ell\right) \text{vec}[\mathcal{D}^\gamma]} \quad (15)$$

**Lemma 2** ( $\oplus$ -vec lemma).  $\text{vec}[\mathcal{D}^\gamma]^T \left(\bigoplus_{\ell \in \gamma} \Psi_\ell\right) \text{vec}[\mathcal{D}^\gamma] = \sum_{\ell \in \gamma} \text{tr}[\mathbf{S}_\ell^\gamma \Psi_\ell]$

*Proof.* This proof relies on the following two properties of  $\text{vec}$ :  $(\mathbf{A} \otimes \mathbf{B}) \text{vec}[\mathbf{C}] = \text{vec}[\mathbf{B}\mathbf{C}^T\mathbf{A}^T]$  and  $\text{tr}[\mathbf{A}^T\mathbf{B}] = \text{vec}[\mathbf{A}]^T \text{vec}[\mathbf{B}]$ . The  $\mathbf{C}$  term picks up a transpose due to our use of the rows-first vectorization; when using columns-first notation the right hand side becomes  $\text{vec}[\mathbf{B}\mathbf{C}\mathbf{A}^T]$ .

$$\text{vec}[\mathcal{D}^\gamma] \left(\bigoplus_{\ell \in \gamma} \Psi_\ell\right) \text{vec}[\mathcal{D}^\gamma] = \sum_{\ell \in \gamma} \text{vec}[\mathcal{D}^\gamma]^T (\mathbf{I}_{d_{<\ell}} \otimes \Psi_\ell \otimes \mathbf{I}_{d_{>\ell}}) \text{vec}[\mathcal{D}^\gamma] \quad (\text{Definition of } \oplus)$$

$$= \sum_{\ell \in \gamma} \text{vec}[\text{mat}_1[\mathcal{D}^\gamma]]^T (\mathbf{I}_{d_{<\ell}} \otimes \Psi_\ell \otimes \mathbf{I}_{d_{>\ell}}) \text{vec}[\text{mat}_1[\mathcal{D}^\gamma]] \quad (16)$$

$$= \sum_{\ell \in \gamma} \text{vec}[\text{mat}_\ell[\mathcal{D}^\gamma]]^T \mathbf{P}_{\ell \rightarrow 1}^T (\mathbf{I}_{d_{<\ell}} \otimes \Psi_\ell \otimes \mathbf{I}_{d_{>\ell}}) \mathbf{P}_{\ell \rightarrow 1} \text{vec}[\text{mat}_\ell[\mathcal{D}^\gamma]] \quad (17)$$

$$= \sum_{\ell \in \gamma} \text{vec}[\text{mat}_\ell[\mathcal{D}^\gamma]]^T (\Psi_\ell \otimes \mathbf{I}_{d_{\setminus \ell}}) \text{vec}[\text{mat}_\ell[\mathcal{D}^\gamma]] \quad (\text{Rearrangement Lemma})$$

$$= \sum_{\ell \in \gamma} \text{vec}[\text{mat}_\ell[\mathcal{D}^\gamma]]^T \text{vec}[\text{mat}_\ell[\mathcal{D}^\gamma] \Psi_\ell^T] \quad (18)$$

$$= \sum_{\ell \in \gamma} \text{tr}[\mathbf{S}_\ell^\gamma \Psi_\ell] \quad (19)$$

□

With this lemma, the probability density function in the single-tensor case can be expressed in the form:

$$p(\mathcal{D}^\gamma) = \frac{\sqrt{\left|\bigoplus_{\ell \in \gamma} \Psi_\ell\right|}}{(2\pi)^{\frac{d_\gamma}{2}}} e^{-\frac{1}{2} \sum_{\ell \in \gamma} \text{tr}[\mathbf{S}_\ell^\gamma \Psi_\ell]} \quad (20)$$

Leading to the probability density function for the multi-tensor case as:

$$p(\{\mathcal{D}^\gamma\}) = \prod_{\gamma} \frac{\sqrt{|\bigoplus_{\ell \in \gamma} \Psi_{\ell}|}}{(2\pi)^{\frac{d_{\Psi}^{\gamma}}{2}}} e^{-\frac{1}{2} \sum_{\ell} \text{tr}[\mathbf{S}_{\ell}^{\gamma} \Psi_{\ell}]} \quad (21)$$

$$= \frac{\prod_{\gamma} \sqrt{|\bigoplus_{\ell \in \gamma} \Psi_{\ell}|}}{(2\pi)^{\frac{d_{\Psi}^{\gamma}}{2}}} e^{-\frac{1}{2} \sum_{\gamma} \sum_{\ell} \text{tr}[\mathbf{S}_{\ell}^{\gamma} \Psi_{\ell}]} \quad (22)$$

$$= \frac{\prod_{\gamma} \sqrt{|\bigoplus_{\ell \in \gamma} \Psi_{\ell}|}}{(2\pi)^{\frac{d_{\Psi}^{\gamma}}{2}}} e^{-\frac{1}{2} \sum_{\ell} \text{tr}[\mathbf{S}_{\ell} \Psi_{\ell}]} \quad (23)$$

The negative log-likelihood is thus:

$$\text{NLL}(\{\mathcal{D}^\gamma\}) = \frac{d_{\Psi}}{2} \log(2\pi) + \frac{1}{2} \sum_{\ell} \text{tr}[\mathbf{S}_{\ell} \Psi_{\ell}] - \frac{1}{2} \sum_{\gamma} \log \left| \bigoplus_{\ell \in \gamma} \Psi_{\ell} \right| \quad (24)$$

### 2.3 Gradient

The derivation of the gradient of the negative log-likelihood is essentially the same as the derivation given by Kalaitzis et al. (2013) for the original Bi-Graphical Lasso. Our derivation is complicated by the fact that we are considering general tensors rather than matrices. We'll let  $\text{sym}$  be the symmetricizing operator that must be applied as we are taking the derivative with respect to a symmetric matrix:  $\text{sym}[\mathbf{M}] = \mathbf{K} \circ \mathbf{M}$ , where  $\mathbf{K}$  is a matrix with 1s on the diagonal and 2s everywhere else. We'll also define  $\mathbf{J}^{ij}$  to be the matrix of zeros except for a 1 at position  $(i, j)$ .

$$\frac{d}{d\Psi_\ell} \text{NLL}(\{\mathcal{D}^\gamma\}) = \frac{1}{2} \text{sym}[\mathbf{S}_\ell] - \frac{1}{2} \sum_\gamma \frac{d}{d\Psi_\ell} \log \left| \bigoplus_{\ell' \in \gamma} \Psi_{\ell'} \right| \quad (25)$$

$$= \frac{1}{2} \text{sym}[\mathbf{S}_\ell] - \frac{1}{2} \sum_\gamma \text{tr} \left[ \left( \bigoplus_{\ell' \in \gamma} \Psi_{\ell'} \right)^{-1} \frac{d}{d\psi_\ell^{ij}} \bigoplus_{\ell' \in \gamma} \Psi_{\ell'} \right]_{ij} \quad (26)$$

$$= \frac{1}{2} \text{sym}[\mathbf{S}_\ell] - \frac{1}{2} \sum_\gamma \text{tr} \left[ \left( \bigoplus_{\ell' \in \gamma} \Psi_{\ell'} \right)^{-1} \left( \mathbf{I}_{d_{<\ell}} \otimes \frac{d}{d\psi_\ell^{ij}} \Psi_\ell \otimes \mathbf{I}_{d_{>\ell}} \right) \right]_{ij} \quad (27)$$

$$= \frac{1}{2} \text{sym}[\mathbf{S}_\ell] - \frac{1}{2} \sum_{\gamma|\ell \in \gamma} \text{tr} \left[ \left( \bigoplus_{\ell' \in \gamma} \Psi_{\ell'} \right)^{-1} \left( \mathbf{I}_{d_{<\ell}} \otimes (\mathbf{J}^{ij} + \mathbf{J}^{ji} - \delta_{ij} \mathbf{J}^{ij}) \otimes \mathbf{I}_{d_{>\ell}} \right) \right]_{ij} \quad (28)$$

$$= \frac{1}{2} \text{sym}[\mathbf{S}_\ell] - \frac{1}{2} \sum_{\gamma|\ell \in \gamma} \left[ (2 - \delta_{ij}) \text{tr} \left[ \left( \bigoplus_{\ell' \in \gamma} \Psi_{\ell'} \right)^{-1} \left( \mathbf{I}_{d_{<\ell}} \otimes \mathbf{J}^{ij} \otimes \mathbf{I}_{d_{>\ell}} \right) \right] \right]_{ij} \quad (29)$$

$$= \frac{1}{2} \text{sym}[\mathbf{S}_\ell] - \frac{1}{2} \sum_{\gamma|\ell \in \gamma} (2\mathbf{J} - \mathbf{I}) \circ \text{tr} \left[ \left( \bigoplus_{\ell' \in \gamma} \Psi_{\ell'} \right)^{-1} \left( \mathbf{I}_{d_{<\ell}} \otimes \mathbf{J}^{ij} \otimes \mathbf{I}_{d_{>\ell}} \right) \right]_{ij} \quad (30)$$

$$= \frac{1}{2} \text{sym}[\mathbf{S}_\ell] - \frac{1}{2} \sum_{\gamma|\ell \in \gamma} \text{sym} \left[ \text{tr}_{d_{>\ell}}^{d_{<\ell}} \left[ \left( \bigoplus_{\ell' \in \gamma} \Psi_{\ell'} \right)^{-1} \right] \right] \quad (31)$$

The MLE occurs when this gradient is zero, i.e. when the following equation is satisfied:

$$\mathbf{S}_\ell = \sum_{\gamma|\ell \in \gamma} \text{tr}_{d_{>\ell}}^{d_{<\ell}} \left[ \left( \bigoplus_{\ell' \in \gamma} \Psi_{\ell'} \right)^{-1} \right] \quad (32)$$

In other words, our effective Gram matrices are the best Kronecker-sum decomposition of the covariance matrix of the maximum likelihood estimate. Unfortunately, the Kronecker-sum decomposition does not interact well with matrix inverses, so this does not directly yield an analytic solution. It does, however, yield a solution for the eigenvectors.

## 2.4 Maximum Likelihood Estimate for the Eigenvectors

We first produce two lemmas to aid in the derivation.

**Lemma 3** (Cyclic property of the stridewise-blockwise trace). *For any matrices  $\mathbf{M}$ ,  $\mathbf{A}_{a \times a}$ ,  $\mathbf{B}_{b \times b}$ , we have that  $\text{tr}_b^a [(\mathbf{A} \otimes \mathbf{I} \otimes \mathbf{B}) \mathbf{M}] = \text{tr}_b^a [\mathbf{M} (\mathbf{A} \otimes \mathbf{I} \otimes \mathbf{B})]$*

*Proof.* This follows directly from the cyclic property of the (normal) trace operator and the definition of the stridewise-blockwise trace.  $\square$

**Lemma 4** (Conjugacy extraction of the stridewise-blockwise trace). *For any matrices  $\mathbf{M}$  and  $\mathbf{V}$ , we have that  $\text{tr}_b^a [(\mathbf{I}_a \otimes \mathbf{V} \otimes \mathbf{I}_b) \mathbf{M} (\mathbf{I}_a \otimes \mathbf{V} \otimes \mathbf{I}_b)^T] = \mathbf{V} \text{tr}_b^a [\mathbf{M}] \mathbf{V}^T$ .*

*Proof.*

$$\text{tr}_b^a \left[ (\mathbf{I}_a \otimes \mathbf{V} \otimes \mathbf{I}_b) \mathbf{M} (\mathbf{I}_a \otimes \mathbf{V} \otimes \mathbf{I}_b)^T \right] = \left[ \text{tr} \left[ (\mathbf{I}_a \otimes \mathbf{V} \otimes \mathbf{I}_b) \mathbf{M} (\mathbf{I}_a \otimes \mathbf{V} \otimes \mathbf{I}_b)^T (\mathbf{I}_a \otimes \mathbf{J}^{ij} \otimes \mathbf{I}_b) \right] \right]_{ij} \quad (\text{Definition of } \text{tr}_b^a)$$

Thanks to the Rearrangement Lemma, we can get this just in terms of the standard blockwise trace, for which there exists a convenient lemma from Dahl et al. (2013) that does the heavy lifting for us. Unfortunately, this requires inserting permutation matrices into every nook and cranny.

$$= \left[ \text{tr} \left[ \mathbf{P} (\mathbf{I}_a \otimes \mathbf{V} \otimes \mathbf{I}_b) \mathbf{P}^T \mathbf{P} \mathbf{M} \mathbf{P}^T \mathbf{P} (\mathbf{I}_a \otimes \mathbf{V} \otimes \mathbf{I}_b)^T \mathbf{P}^T \mathbf{P} (\mathbf{I}_a \otimes \mathbf{J}^{ij} \otimes \mathbf{I}_b) \mathbf{P}^T \right] \right]_{ij} \quad (33)$$

$$= \left[ \text{tr} \left[ (\mathbf{V} \otimes \mathbf{I}_{ab})^T \mathbf{P} \mathbf{M} \mathbf{P}^T (\mathbf{V} \otimes \mathbf{I}_{ab}) (\mathbf{J}^{ij} \otimes \mathbf{I}_{ab}) \right] \right]_{ij} \quad (34)$$

$$= \text{tr}_{ab} \left[ (\mathbf{V} \otimes \mathbf{I}_{ab}) \mathbf{P} \mathbf{M} \mathbf{P}^T (\mathbf{V} \otimes \mathbf{I}_{ab})^T \right] \quad (\text{Definition of } \text{tr}_{ab})$$

$$= \mathbf{V} \text{tr}_{ab} \left[ \mathbf{P} \mathbf{M} \mathbf{P}^T \right] \mathbf{V}^T \quad (\text{Lemma 2 of Dahl et al. (2013)})$$

We then can see analogously that  $\text{tr}_{ab} [\mathbf{P} \mathbf{M} \mathbf{P}^T] = \text{tr}_b^a [\mathbf{M}]$ , completing the proof.  $\square$

**Theorem 1.** Let  $\mathbf{V}_\ell \mathbf{e}_\ell \mathbf{V}_\ell^T$  be the eigendecomposition of  $\mathbf{S}_\ell$ . Then  $\mathbf{V}_\ell$  are the eigenvectors of the maximum likelihood estimate of  $\Psi_\ell$ .

*Proof.*

$$\mathbf{S}_\ell = \sum_{\gamma | \ell \in \gamma} \text{tr}_{d_{>\ell}^\gamma}^{d_{<\ell}^\gamma} \left[ \left( \bigoplus_{\ell' \in \gamma} \Psi_{\ell'} \right)^{-1} \right] \quad (35)$$

$$= \sum_{\gamma | \ell \in \gamma} \text{tr}_{d_{>\ell}^\gamma}^{d_{<\ell}^\gamma} \left[ \left( \bigoplus_{\ell' \in \gamma} \mathbf{V}_{\ell'} \Lambda_{\ell'} \mathbf{V}_{\ell'}^T \right)^{-1} \right] \quad (36)$$

$$= \sum_{\gamma | \ell \in \gamma} \text{tr}_{d_{>\ell}^\gamma}^{d_{<\ell}^\gamma} \left[ \left( \bigotimes_{\ell'} \mathbf{V}_{\ell'} \right) \left( \bigoplus_{\ell' \in \gamma} \Lambda_{\ell'} \right)^{-1} \left( \bigotimes_{\ell'} \mathbf{V}_{\ell'}^T \right) \right] \quad (37)$$

$$= \sum_{\gamma | \ell \in \gamma} \text{tr}_{d_{>\ell}^\gamma}^{d_{<\ell}^\gamma} \left[ (\mathbf{I}_{d_{<\ell}} \otimes \mathbf{V}_\ell \otimes \mathbf{I}_{d_{>\ell}}) \left( \bigoplus_{\ell' \in \gamma} \Lambda_{\ell'} \right)^{-1} (\mathbf{I}_{d_{<\ell}} \otimes \mathbf{V}_\ell \otimes \mathbf{I}_{d_{>\ell}})^T \right] \quad (\text{Cyclic Property})$$

$$= \sum_{\gamma | \ell \in \gamma} \mathbf{V}_\ell \text{tr}_{d_{>\ell}^\gamma}^{d_{<\ell}^\gamma} \left[ \left( \bigoplus_{\ell' \in \gamma} \Lambda_{\ell'} \right)^{-1} \right] \mathbf{V}_\ell^T \quad (\text{Conjugacy Extraction})$$

$$= \mathbf{V}_\ell \left( \sum_{\gamma | \ell \in \gamma} \text{tr}_{d_{>\ell}^\gamma}^{d_{<\ell}^\gamma} \left[ \left( \bigoplus_{\ell' \in \gamma} \Lambda_{\ell'} \right)^{-1} \right] \right) \mathbf{V}_\ell^T \quad (38)$$

We conclude the proof by observing that the central matrix is diagonal, and thus the right hand side constitutes an eigendecomposition of  $\mathbf{S}_\ell$ . Thus  $\mathbf{S}_\ell$  and  $\Psi_\ell$  share eigenvectors.  $\square$

## 2.5 Maximum Likelihood Estimate for the Eigenvalues

In the previous section, we derived the eigenvectors of the maximum likelihood estimate. While interesting (they correspond to the principal components of our data), we need the eigenvalues to reconstruct  $\Psi_\ell$ . In the main paper, we derived the gradient with respect to the Negative Log-Likelihood. Then, for a learning rate  $\mu_t$ , gradient descent can be performed with the update equation  $\Lambda_\ell^{t+1} = \Lambda_\ell^t - \mu_t \left[ \mathbf{e}_\ell - \sum_{\gamma|\ell \in \gamma} \text{tr}_{d_{>\ell}^{\gamma}} \left[ \left( \bigoplus_{\ell' \in \gamma} \Lambda_{\ell'} \right)^{-1} \right] \right]$ . A reasonable restriction is to make  $\Psi_\ell$  is positive definite, in which case  $\mu_t$  must be chosen to prevent  $\Lambda_\ell^t$  from becoming negative. Iterating over the eigenvalues reduces our optimization task from one with  $\sum_\ell d_\ell^2$  parameters to one with  $\sum_\ell d_\ell$  parameters.

## 2.6 Incorporation of Priors

**Theorem 2** (GmGM Eigenvector Estimator with Priors). *Suppose we have the same setup as in Theorem 1, and that we have a prior of the form:*

$$\prod_{\ell} g_{\ell}(\Theta_{\ell}) e^{\text{tr}[\eta_{\ell}(\Theta_{\ell})^T \Psi_{\ell}] + h_{\ell}(\Psi_{\ell})} \quad (39)$$

*In other words, our prior is an exponential distribution for  $\Psi_\ell$ , in which  $\Psi_\ell$  is the sufficient statistic.*

*Then, if  $h_\ell$  depends only on the eigenvalues (i.e. it is ‘unitarily invariant’), the eigenvectors of  $\frac{1}{2}\mathbf{S}_\ell - \eta_\ell(\Theta_\ell)$  are the eigenvectors  $\mathbf{V}_\ell$  of the MAP estimate for  $\Psi_\ell$ .*

*Proof.* Observe that our log-prior is:

$$\sum_{\ell} \log g_{\ell}(\Theta_{\ell}) + \text{tr} \left[ \eta_{\ell}(\Theta_{\ell})^T \Psi_{\ell} \right] + h_{\ell}(\Psi_{\ell}) \quad (40)$$

With gradient:

$$\eta_{\ell}(\Theta_{\ell}) + \frac{\partial}{\partial \Psi_{\ell}} h_{\ell}(\Psi_{\ell}) \quad (41)$$

Theorem 1.1 from Lewis (1996) states that the eigenvectors of the derivative of any spectral function are the same as the eigenvectors of its input. In our case, this means that the eigenvectors of  $\frac{\partial}{\partial \Psi_{\ell}} h_{\ell}(\Psi_{\ell})$  are  $\mathbf{V}_\ell$ , the eigenvectors of  $\Psi_\ell$ .

Recall the gradient of the log-likelihood of the Kronecker-sum-structured normal distribution was:

$$\frac{1}{2} \sum_{\gamma|\ell \in \gamma} \text{tr}_{d_{>\ell}^{\gamma}} \left[ \left( \bigoplus_{\ell' \in \gamma} \Psi_{\ell'} \right)^{-1} \right] - \frac{1}{2} \mathbf{S}_{\ell} \quad (42)$$

The gradient of the log-likelihood after including the prior is the sum of these two gradients:

$$\left( \frac{1}{2} \sum_{\gamma|\ell \in \gamma} \text{tr}_{d_{>\ell}^{\gamma}} \left[ \left( \bigoplus_{\ell' \in \gamma} \Psi_{\ell'} \right)^{-1} \right] + \frac{\partial}{\partial \Psi_{\ell}} h_{\ell}(\Psi_{\ell}) \right) - \frac{1}{2} \mathbf{S}_{\ell} + \eta_{\ell}(\Theta_{\ell}) \quad (43)$$

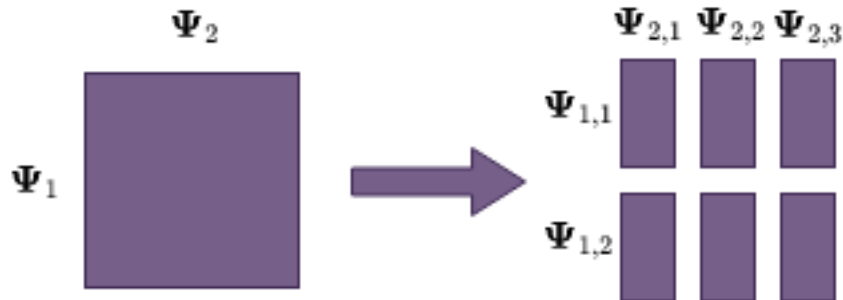


Figure 1: Graphical representation of the benefits of Theorem 3. Provided the thresholded covariance matrix can be partitioned, so can the precision matrix. This allows us to split the dataset into distinct parts, and estimate the covariance matrices separately. Note that, in the diagram, we have split the dataset into 6 chunks but there are only 5 precision matrices to estimate - this is because there are shared axes, necessitating the use of our shared axis framework to take advantage of this partitioning.

And the optimum is achieved at:

$$\left( \frac{1}{2} \sum_{\gamma|\ell \in \gamma} \text{tr}_{d_{>\ell}^{\gamma}} \left[ \left( \bigoplus_{\ell' \in \gamma} \Psi_{\ell'} \right)^{-1} \right] + \frac{\partial}{\partial \Psi_{\ell}} h_{\ell}(\Psi_{\ell}) \right) = \frac{1}{2} \mathbf{S}_{\ell} - \eta_{\ell}(\Theta_{\ell}) \quad (44)$$

The right hand side is known, and hence has known eigenvectors. The left hand side is unknown, but has eigenvectors  $\mathbf{V}_{\ell}$ . □

The change to the iterative portion of the GmGM algorithm is minimal; by virtue of being unitarily invariant and convex, Lewis (1996) guarantees that  $\frac{\partial}{\partial \Psi_{\ell}} h_{\ell}(\Psi_{\ell})$  is some function of just the eigenvalues  $\Lambda_{\ell}$  of  $\Psi_{\ell}$ . This function is added to our gradient each step (Line 41). This fact is significant, as it means the algorithm does not need major changes to incorporate priors - we only need to consider  $\frac{\partial}{\partial \Lambda_{\ell}} h_{\ell}(\Lambda_{\ell})$ .  $\frac{1}{2} \mathbf{S}_{\ell} - \eta_{\ell}(\Theta_{\ell})$  can be thought of as our ‘priorized’ effective Gram matrix, its eigenvalues fill the role exactly of  $\mathbf{e}_{\ell}$  in the original algorithm. Thus, the only change needed to incorporate priors in the algorithm, other than use of the prioritized Gram matrices, is inclusion of the  $\frac{\partial}{\partial \Lambda_{\ell}} h_{\ell}(\Lambda_{\ell})$  term.

**Corollary 1.** *The Wishart and matrix gamma distributions all satisfy Theorem 2.*

## 2.7 Covariance Thresholding

In this section, we restate and prove Theorem 4 from the main paper in more general terms; see Figure 1. When we say that two matrices  $\mathbf{A}$  and  $\mathbf{B}$  have the same block diagonal structure, we mean that both have the following form:

$$\mathbf{B} = \begin{bmatrix} \mathbf{B}_1 & 0 & \dots \\ 0 & \mathbf{B}_2 & \\ \vdots & & \ddots \end{bmatrix} \quad (45)$$

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 & 0 & \dots \\ 0 & \mathbf{A}_2 & \\ \vdots & & \ddots \end{bmatrix} \quad (46)$$

Where there is no restrictions placed on the submatrices except that they are square and that the shape of  $\mathbf{A}_i$  is the same as the shape of  $\mathbf{B}_i$  for all  $i$ . The structure of our proof of Lemma 6 follows the same structure as that of Theorem 1 of Mazumder and Hastie (2012); we give it in a more general form in such that both their result and our result arise as corollaries in conjunction with Lemma 6.

**Lemma 5.** Let  $\mathbf{F}[\Psi_\ell; \{\Psi_{\ell'}\}_{\ell' \neq \ell}]$  be a function that preserves the block diagonal structure of  $\Psi_\ell$ , which as a shorthand we will denote  $\mathbf{F}_\ell$ . Suppose we had an objective whose gradient is  $\mathbf{S}_\ell - \mathbf{F}_\ell$ . If we incorporate an L1 penalty on  $\Psi_\ell$  with strength  $\rho$  into the objective, then the subgradient of the function contains zero for an estimate  $\hat{\Psi}_\ell$  with the following property:

If we replace every element of  $\mathbf{S}_\ell$  whose magnitude is less than  $\rho$  with 0, we can partition the encoded graph into disconnected components. This partition is the same as the partition into disconnected components for  $\hat{\Psi}_\ell$ .

*Proof.* The incorporation of an L1 penalty leads to a subgradient of:

$$\mathbf{S}_\ell - \mathbf{F}_\ell + \rho \text{sign} \Psi_\ell \quad (47)$$

Where sign is the element-wise sign function:

$$\text{sign} \triangleq [\mathbf{M}]_{ij} \begin{cases} \in [-\rho, \rho] & \text{if } \mathbf{M}_{ij} = 0 \\ = -\rho & \text{if } \mathbf{M}_{ij} > 0 \\ = \rho & \text{if } \mathbf{M}_{ij} < 0 \end{cases} \quad (48)$$

We have critical points when the subgradient contains zero; this leads to the KKT conditions, which must be satisfied for all  $ij$ .

$$\left| \mathbf{S}_\ell^{ij} - \mathbf{F}_\ell^{ij} \right| \leq \rho \quad \text{if } \Psi_\ell^{ij} = 0 \quad (49)$$

$$\mathbf{F}_\ell^{ij} = \mathbf{S}_\ell^{ij} + \rho \quad \text{if } \Psi_\ell^{ij} > 0 \quad (50)$$

$$\mathbf{F}_\ell^{ij} = \mathbf{S}_\ell^{ij} - \rho \quad \text{if } \Psi_\ell^{ij} < 0 \quad (51)$$

Define  $\text{thresh}_\rho[\mathbf{S}_\ell]$  to be a matrix with 0s everywhere that  $|\mathbf{S}_\ell^{ij}| < \rho$  and otherwise is  $\mathbf{S}_\ell^{ij}$ . Without loss of generality, we can assume our data has been ordered such that  $\text{thresh}_\rho[\mathbf{S}_\ell]$  is block diagonal. We will construct a  $\hat{\Psi}_\ell$  that satisfies the KKT conditions while also having a block-diagonal structure. The construction of such a  $\hat{\Psi}_\ell$  would show that the graph partition in our L1-penalized solution is at least as fine as the graph partition arising from thresholding  $\mathbf{S}_\ell$ .

Let us construct  $\hat{\Psi}_\ell$  to be block-diagonal with the same block structure as  $\text{thresh}_\rho[\mathbf{S}_\ell]$ . Denote the blocks to be indexed by  $a$  ( $\hat{\Psi}_{\ell,a}$  for  $\Psi$ ,  $\mathbf{S}_{\ell,a}$  for  $\mathbf{S}$ ), and define them to be solutions to the following problem:

$$0 \in \mathbf{S}_{\ell,a} - \mathbf{F}[\Psi_{\ell,a}; \{\Psi_{\ell'}\}_{\ell' \neq \ell}] + \rho \text{sign} \Psi_{\ell,a} \quad (52)$$

To see why  $\hat{\Psi}_\ell$  satisfies the KKT conditions, note that for any index  $ij$  lying off the block diagonal,  $\mathbf{S}_\ell^{ij} < \rho$  by construction. As  $\mathbf{F}_{ij}$  preserves the sparsity structure of  $\hat{\Psi}_\ell$ , and  $\hat{\Psi}_\ell^{ij} = 0$ ,  $|\mathbf{S}_\ell^{ij} - \mathbf{F}_{ij}| = 0 < \rho$  is satisfied. For the indices  $ij$  lying on the block diagonal, note that the KKT conditions are satisfied via each block's constricton in Equation 52.

To conclude the proof, it is necessary to show that if our solution  $\hat{\Psi}_\ell$  has a block diagonal structure, then  $\mathbf{S}_\ell$  has the same block diagonal structure. In other words, the L1-penalized solution is not a more fine-grained partition than that which can be attained through covariance thresholding. Suppose that we have some  $\Psi_\ell$  with

block-diagonal structure, and thus that  $\mathbf{F}_\ell$  has that same structure. Then,  $\mathbf{F}_\ell^{ij} = 0$  implies  $|\mathbf{S}_\ell^{ij}| \leq \rho$ , as by the KKT we know that  $|\mathbf{S}_\ell^{ij} - \mathbf{F}_\ell^{ij}| \leq \rho$ . Thus,  $\text{thresh}_\rho[\mathbf{S}_\ell]$  has the same block diagonal structure as  $\Psi_\ell$ .

We have now shown that both graph partitions are equally fine-grained, and are thus equivalent. This completes the proof.  $\square$

**Corollary 2.** *As the matrix inverse preserves block diagonal structure, Theorem 1 of Mazumder and Hastie (2012) follows immediately.*

**Lemma 6.**  $\sum_{\gamma|\ell \in \gamma} \text{tr}_{d_{>\ell}^{>\ell}} \left[ \left( \bigoplus_{\ell' \in \gamma} \Psi_{\ell'} \right)^{-1} \right]$  preserves the block diagonal structure of  $\Psi_\ell$ .

*Proof.* For notational convenience, and without loss of generality, assume that  $\ell$  is the first axis in each of the modalities it appears in. Furthermore, observe that if for all  $\gamma$  we have that  $\text{tr}_{d_{>\ell}^{>\ell}} \left[ \left( \bigoplus_{\ell' \in \gamma} \Psi_{\ell'} \right)^{-1} \right]$  preserves the block diagonal structure, then the sum  $\sum_{\gamma|\ell \in \gamma} \text{tr}_{d_{>\ell}^{>\ell}} \left[ \left( \bigoplus_{\ell' \in \gamma} \Psi_{\ell'} \right)^{-1} \right]$  also preserves this structure. Thus, we consider only one term in the sum and omit reference to  $\gamma$ . Doing this, our problem reduces to showing that  $\text{tr}_{d_{\setminus \ell}} \left[ \left( \Psi_\ell \otimes \mathbf{I}_{d_{\setminus \ell}} + \mathbf{I}_{d_\ell} \otimes \bigoplus_{\ell' \neq \ell} \Psi_{\ell'} \right)^{-1} \right]$  preserves the block diagonal structure.

While  $\Psi_\ell \otimes \mathbf{I}_{d_{\setminus \ell}}$  does not preserve the block diagonal structure in the strict sense (as its blocks have different sizes than those of  $\Psi_\ell$ ), it is still block diagonal and there is an obvious mapping from the blocks of one to the blocks of the other:

$$\begin{bmatrix} \Psi_\ell^1 & 0 & \dots \\ 0 & \Psi_\ell^2 & \\ \vdots & & \ddots \end{bmatrix} \otimes \mathbf{I}_{d_{\setminus \ell}} = \begin{bmatrix} \Psi_\ell^1 \otimes \mathbf{I}_{d_{\setminus \ell}} & 0 & \dots \\ 0 & \Psi_\ell^2 \otimes \mathbf{I}_{d_{\setminus \ell}} & \\ \vdots & & \ddots \end{bmatrix} \quad (53)$$

Suppose block  $\Psi_i$  has size  $s_i$ , then the size of the blocks in  $\Psi_\ell \otimes \mathbf{I}_{d_{\setminus \ell}}$  is  $s_i \times d_{\setminus \ell}$ . The second term in the sum is also block diagonal:

$$\begin{bmatrix} 1 & 0 & \dots \\ 0 & 1 & \\ \vdots & & \ddots \end{bmatrix} \otimes \bigoplus_{\ell' \neq \ell} \Psi_{\ell'} = \begin{bmatrix} \bigoplus_{\ell' \neq \ell} \Psi_{\ell'} & 0 & \dots \\ 0 & \bigoplus_{\ell' \neq \ell} \Psi_{\ell'} & \\ \vdots & & \ddots \end{bmatrix} \quad (54)$$

But these blocks are of size  $1 \times d_{\setminus \ell}$ ; no larger than the blocks of  $\Psi_\ell \otimes \mathbf{I}_{d_{\setminus \ell}}$ . Thus, overall,  $\Psi_\ell \otimes \mathbf{I}_{d_{\setminus \ell}} + \mathbf{I}_{d_\ell} \otimes \bigoplus_{\ell' \neq \ell} \Psi_{\ell'}$  has block diagonal structure with block sizes  $s_i \times d_{\setminus \ell}$ . It follows that its inverse does as well. The blockwise trace maps blocks of size  $s_i \times d_{\setminus \ell}$  back to size  $s_i$ , completing the proof.  $\square$

**Theorem 3.** *Set all elements of  $\mathbf{S}_\ell$  whose absolute value is less than  $\rho$  to 0. This encodes a potentially disconnected graph. Likewise, consider  $\Psi_\ell$  to be the estimated precision matrix for our model equipped with an L1 penalty of strength  $\rho$ . This also encodes a potentially disconnected graph. If we label the vertices by which disconnected component they are part of, then this labeling is the same in both procedures (the procedure with  $\mathbf{S}_\ell$  and the procedure with  $\Psi_\ell$ ).*

*Proof.* This follows directly from the convexity of our objective as well as Lemmas 5 and 6.  $\square$

### 3 DEPENDENCIES

All tests and figures were generated on a Mac with an M1 chip and 8GB of RAM. Along with our code, we provide an environment file (environment.yml) that contains full details of all the dependencies used. In our



---

### The GmGM algorithm

---

**Input:**  $\{\mathcal{D}_i^\gamma\}$ , tolerance

**Output:**  $\{\Psi_\ell\}$

```
1: for  $1 \leq \ell \leq K$ 
2:    $\mathbf{S}_\ell \leftarrow \sum_{\gamma|\ell \in \gamma} \frac{1}{n^\gamma} \sum_i^{n^\gamma} \text{mat}_\ell [\mathcal{D}_i^\gamma] \text{mat}_\ell [\mathcal{D}_i^\gamma]^T$ 
3:    $\mathbf{V}_\ell \leftarrow \text{eigenvectors}[\mathbf{S}_\ell]$ 
4:    $\mathbf{e}_\ell \leftarrow \text{eigenvalues}[\mathbf{S}_\ell]$ 
5: end for
6:  $\mathbf{\Lambda} \leftarrow [1 \ \dots \ 1]^T$ 
7:  $\mu \leftarrow 1$ 
8: while not converged
9:   for  $1 \leq \ell \leq K$ 
10:     $\mathbf{G}_\ell^\gamma \leftarrow \text{proj}_{KS} \left[ \left( \bigoplus_{\ell' \in \gamma} \mathbf{\Lambda}_{\ell'} \right)^{-1} \right]$ 
11:     $\mathbf{\Lambda}'_\ell \leftarrow \mathbf{\Lambda}_\ell - \mu \left[ \mathbf{e}_\ell - \sum_{\gamma|\ell \in \gamma} \mathbf{G}_\ell^\gamma \right]$ 
12:   end for
13:   for  $1 \leq \ell \leq K$ 
14:     $\mathbf{\Lambda}_\ell \leftarrow \mathbf{\Lambda}'_\ell$ 
15:   end for
16:   for  $\gamma \in \text{modalities}$ 
17:    if  $\sum_{\ell \in \gamma} \min \mathbf{\Lambda}_\ell < \text{tolerance}$  then
18:      decrease  $\mu$  to make sum far from zero
19:    end if
20:   end for
21: end while
22: for  $1 \leq \ell \leq K$ 
23:    $\Psi_\ell \leftarrow \mathbf{V}_\ell \mathbf{\Lambda}_\ell \mathbf{V}_\ell^T$ 
24: end for
```

---

GitHub repository (<https://github.com/AIStats-GmGM/GmGM/>), we give precise and simple instructions on how to create a conda environment with the same setup as ours. Most of the packages used were specific to the experiments we ran. The dependencies necessary for our algorithm were Python 3.9 and NumPy 1.23.5.

## 4 EXPERIMENTS

### 4.1 Synthetic data

For various precision-recall performances, see Figures 2, 3, and 4. For runtimes, see Figure 5. We can see that our algorithm performs comparably to prior work; it does slightly worse without the restricted L1 penalty. We focused on experimenting with the 1-sample case, as that is typically the case with real-world data (all of the five real-world experiments in Section 4 are an example of this).

For the runtimes, we cut off our tests once GmGM reached the one-minute runtime mark, or if the size of the synthetic data exceeded a gigabyte; see Table 1 for an overview, and Section 6 for the computational complexity. Note that the runtime is almost linear in the total size of the dataset  $d_V$  ( $O(Kd_V^{1+\frac{1}{K}})$ ). As  $K$  increases,  $\frac{1}{K}$  vanishes. The cost of Gram matrix computation and eigendecomposition compared to the iterations also levels out; this is likely what explains the higher-order tensors all taking roughly the same amount of time to compute 1GB-sized datasets (10 seconds). These results clearly show that for high-axis-data ( $K > 3$ ), our algorithm can handle all smaller-than-RAM datasets in reasonable time on a personal computer. For low-axis data ( $K < 4$ ), it may take more than a minute but, as we can see from our algorithm’s performance on large real world datasets such as the 10x Genomics dataset, its runtime will still be measured in minutes, not hours, before reaching the limits of RAM. When we consider the covariance-thresholding trick (Figure 7b in the main paper), we can push these limits even further.

Amount of Axes ( $K$ )	$d_{GB}$	$d_{min}$	Runtime at $d_{GB}$
2	11,000	4,000	???
3	500	400	???
4	100	???	7.1 seconds
5	40	???	9.5 seconds
6	20	???	7.4 seconds

Table 1: Comparing the point at which a  $K$ -axis tensor of double precision floats surpasses a gigabyte of data ( $\sqrt[K]{\frac{1000000000}{64}} = d_{GB}$ ) to the point at which GmGM’s runtime surpasses a minute ( $d_{min}$ ). Values are rounded.

In all tests, our regularized algorithm took about the same time as the unregularized algorithm. This is not unexpected, as our restricted regularization only affects the iterations, not the eigendecomposition. In the real world, where the data may be ‘harder’ than our synthetic data (i.e. take more iterations to converge), we would expect our regularized algorithm to be slower.

### 4.2 COIL video

#### 4.2.1 The Dataset

We downloaded the processed COIL-20 (Nene et al., n.d.) dataset, and tested our model on it. It is available at <https://cave.cs.columbia.edu/repository/COIL-20/>; we used the ‘processed’ data. They do not formally license it, but state that it is available for academic purposes and provide the download links with no restrictions.

#### 4.2.2 Experiment Justification

In the original paper introducing Kronecker-sum-structured normal distributions for graphical models, Kalaitzis et al. (2013) showed that their BiGraphical Lasso (BiGLasso) could reconstruct the ordering of the frames of the video. Thus, it seemed natural to choose this as an experiment, to show that our lack of L1 regularization was not a significant inhibitor for the model in the real world setting.

BiGLasso was a significant algorithm in that it was the first of this type of model and proved that it could work, but it was also incredibly slow (taking around 15 minutes on 100x100 matrix data) and limited to two axes. Thus,

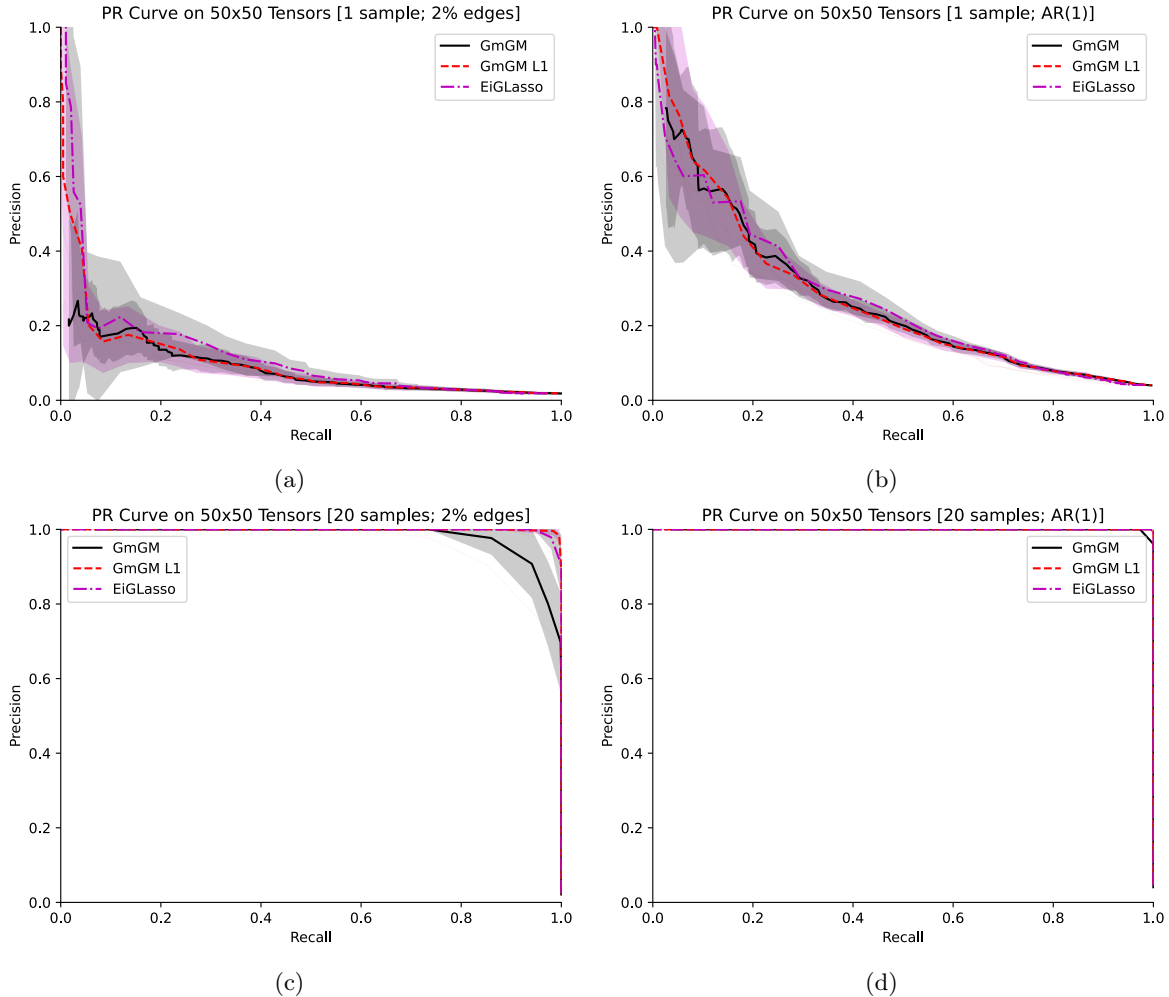


Figure 2: Various precision-recall curves on synthetic matrix-variate data, for both an Erdos-Renyi with a 2% per-edge probability and AR(1) distribution.

they reduced the resolution of the image from 72 frames and 128x128 pixels to 36 frames and 9x9 pixels and flattened the rows and columns axis into one axis. We wanted to show that our algorithm could handle the full resolution in negligible time while still reconstructing the video.

### 4.2.3 Results

We wanted to see if our model could understand the structure of a video, which we expected to consist of two linear graphs (for the rows and columns, i.e. each row is connected only to its neighbor rows) and a circular graph (for the frames, because the video is of a 360° rotation). To generate these graphs, we ran our algorithm on the duck video from the dataset, and then greedily kept the largest edge from each vertex such that vertices in the final graph had at most two edges. If we shuffled our data (shuffle rows, columns, and frames) and try to reconstruct it with these graphs, we get decent without the nonparanormal skeptic (Figure 6a) and great results with it (Figure 6b). While in both cases the algorithm has clearly reconstructed most of the duck, when using the nonparanormal skeptic our algorithm reconstructs it nearly perfectly. The only human-noticeable issue is that it does not know which row/column the duck should start on - if we tessellate the image, the reconstruction would be indistinguishable from the original image.

We can put a numeric value to the reconstruction, by measuring the percentage of the time that our reconstructed edges connect two adjacent rows/columns/frames. Without the nonparanormal skeptic, we get an accuracy of 88% for the rows, 94% for the columns, and 93% for the frames. With it, the accuracy becomes 99% for all

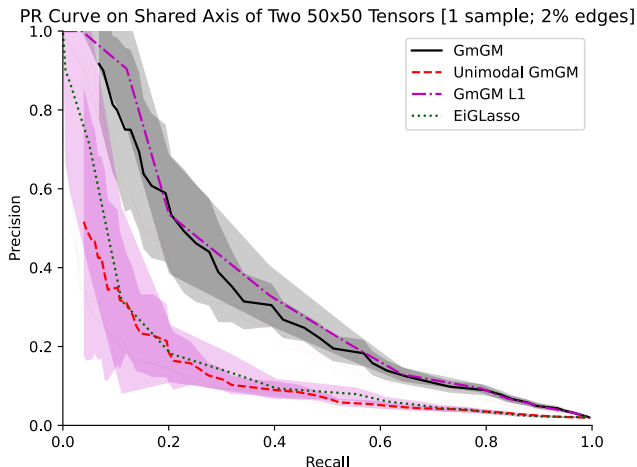


Figure 3: Performance on two 50x50 datasets with one shared axis, whose true graphs were drawn from an Erdos-Renyi distribution (2% probability for each edge to exist, independently). Unimodal GmGM and EiGLasso only consider one of the two datasets.

Algorithm	Preprocessing	Runtime
GmGM	Center Data	0.044 seconds
GmGM	Nonparanormal Skeptic	0.23 seconds
TeraLasso	Center Data	33 seconds
TeraLasso	Nonparanormal Skeptic	5.2 seconds

Table 2: A comparison of the runtimes our algorithm and TeraLasso. Runtimes were averaged over 10 runs, and include the preprocessing.

three. These values are without regularization; we did experiment with regularization, but it did not seem to help. TeraLasso had the same results.

On this dataset, our algorithm runs in 0.044 seconds whereas TeraLasso takes 33 seconds, showing that our algorithm is highly performant on real world data. For data of this size, the nonparanormal skeptic is expensive compared to our algorithm - but our algorithm is still an order of magnitude faster than TeraLasso. If we do not include the nonparanormal skeptic in our runtime calculation, the speed of GmGM is comparable regardless of the input. However, convergence is much quicker for TeraLasso when it is fed the nonparanormal skeptic (Table 2).

### 4.3 EchoNet-Dynamic ECGs

#### 4.3.1 The Dataset

We downloaded all of the EchoNet-Dynamic (Ouyang et al., 2020) data. The dataset is available at <https://echonet.github.io/dynamic/index.html>; one has to sign a Research Use Agreement to access it. It is available for personal, non-commercial research purposes only. It prohibits distribution of portions of the data, which is why we do not show any frames from the echocardiograms in this paper. The dataset has labels (for example, end systolic volume), but these labels are not relevant to the task of predicting the frames the heartbeats occur on. For this, we picked the opening of the mitral valve as an arbitrary but easily visible occurrence, and manually recorded the frames of these openings for 5 videos in the dataset. These videos were picked at random.

#### 4.3.2 Experiment Justification

We chose the COIL duck video because of its use in prior work, but these methods were already known to perform quite well on it. Thus, we wanted to try a similar but harder task. Echocardiograms have a periodic structure (the heartbeat), and hence we would expect the learned graph of an echocardiogram to be similar to that of the duck video, with the addition of extra connections corresponding to the same phase of the heartbeat across

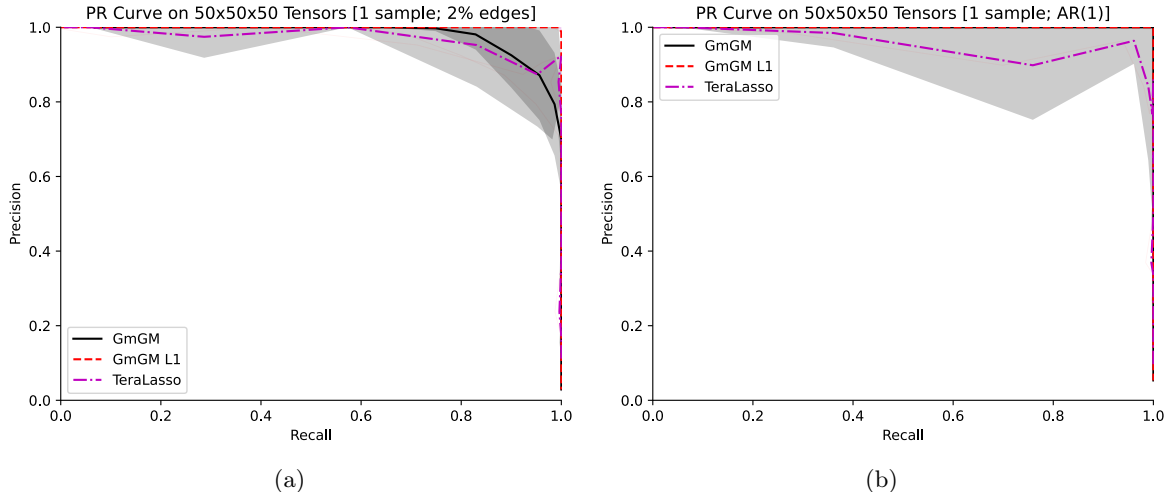


Figure 4: Performance on tensor-variate (50x50x50) data whose true graphs were drawn from an Erdos-Renyi distribution (2% edge probability) and an AR(1) distribution.

Algorithm	Preprocessing	Runtime
GmGM	None	0.067 seconds
GmGM	Nonparanormal Skeptic	0.33 seconds
TeraLasso	None	40 seconds
TeraLasso	Nonparanormal Skeptic	12 seconds

Table 3: Runtimes of our algorithm and TeraLasso, with various preprocessing methods, on the EchoNet-Dynamic dataset. Runtimes given are the average runtime over the five videos considered.

different heartbeats. We saw that our algorithm was able to find this more complicated structure, and prove it by using that structure to detect future heartbeats.

### 4.3.3 Results

We downloaded all of the EchoNet-Dynamic (Ouyang et al., 2020) data. This dataset did not have heartbeats labeled, so we picked a few videos at random and labeled them ourselves as a proof of concept. Specifically, we labeled every frame in which the mitral valve opened. Our goal was to see if the graphs produced by our algorithm could predict this opening. Table 4 contains the videos we picked, the labels we gave, and the labels we predicted. When we used the nonparanormal skeptic as a preprocessing step, we got broadly similar empirical results (Table 5), although the precision matrices arguably more clearly show the periodic structure.

Mitral valve predictions were done by taking GmGM’s output frames graph in precision matrix form, and measuring the mass along the diagonals. We treated this as a time series (since each diagonal corresponds to an increasing time offset from all frames simultaneously). We applied gaussian blur and then a continuous wavelet transform peak detection algorithm (Du et al., 2006) to find which diagonals had the most mass (Figure 7). These represent the offsets corresponding with a heartbeat. Given the first mitral valve opening and these offsets, we predict the remaining openings. From Tables 4 and 5 we can see that our algorithm performs well. Our algorithm is much quicker than TeraLasso; see Table 3.

## 4.4 Mouse embryo stem cell transcriptomics

### 4.4.1 The Dataset

We used the mouse embryo stem cell dataset E-MTAB-2805 (Buettner et al., 2015), available at <https://www.ebi.ac.uk/biostudies/arrayexpress/studies/E-MTAB-2805> under a Creative Commons Zero license. This dataset is by what stage of the cell cycle each cell was in (G1, S, and G2M). We consider a subset of the genes available in this dataset, as that was the case in the scBiGLasso paper (Li et al., 2022); this subset is given





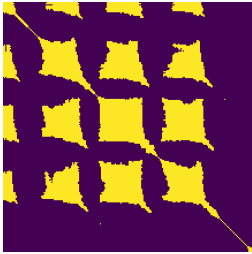
Video ID	Label	Predicted	Precision Matrix
0XFE6E32991136338	[17, 47, 77, 106]	[17, 47, 78, 104]	<p>Echocardiogram Frames</p> 
0XF072F7A9791B060	[24, 56, 100]	[24, 59, 90]	<p>Echocardiogram Frames</p> 
0XF70A3F712E03D87	[22, 66, 110]	[22, 67, 111]	<p>Echocardiogram Frames</p> 
0XF60BBEC9C303C98	[19, 67, 114, 162]	[19, 66, 115, 162]	<p>Echocardiogram Frames</p> 
0XF46CF63A2A1FA90	[25, 79, 134, 188]	[25, 80, 133, 184]	<p>Echocardiogram Frames</p> 

Table 4: Mitral valve labelings and precision matrices for the EchoNet-Dynamic dataset. The precision matrices, for the most part, seem to have clear off-diagonal structures, as expected, and the mitral valve prediction is generally quite good; it is only significantly off for the last opening in 0XF072F7A9791B060. The precision matrices are shown with the top 20% of the edges kept.

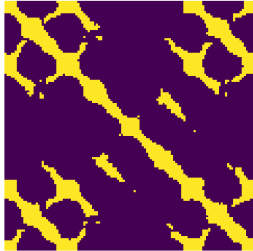
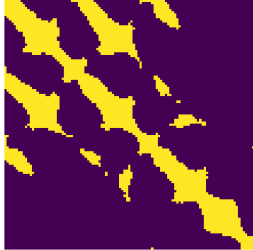
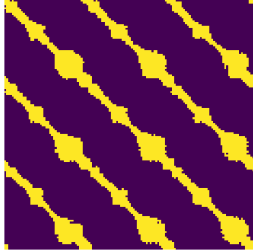
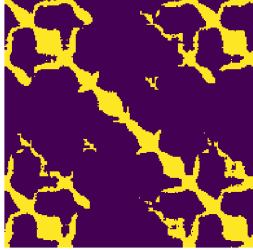
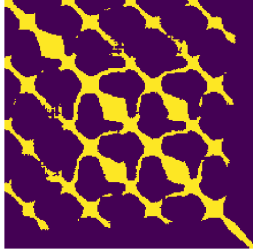
Video ID	Label	Predicted	Precision Matrix
0XFE6E32991136338	[17, 47, 77, 106]	[17, 48, 77, 106]	 <p>Precision Matrices</p>
0XF072F7A9791B060	[24, 56, 100]	[24, 58, 91]	 <p>Precision Matrices</p>
0XF70A3F712E03D87	[22, 66, 110]	[22, 67, 112]	 <p>Precision Matrices</p>
0XF60BBEC9C303C98	[19, 67, 114, 162]	[19, 66, 116, 162]	 <p>Precision Matrices</p>
0XF46CF63A2A1FA90	[25, 79, 134, 188]	[25, 80, 134, 188]	 <p>Precision Matrices</p>

Table 5: Mitral valve labelings and precision matrices for the EchoNet-Dynamic dataset, using the nonparanormal skeptic to preprocess. The precision matrices, for the most part, seem to have clear off-diagonal structures, as expected, and the mitral valve prediction is generally quite good; it is only significantly off for the last opening in 0XF072F7A9791B060. The precision matrices are shown with the top 20% of the edges kept.

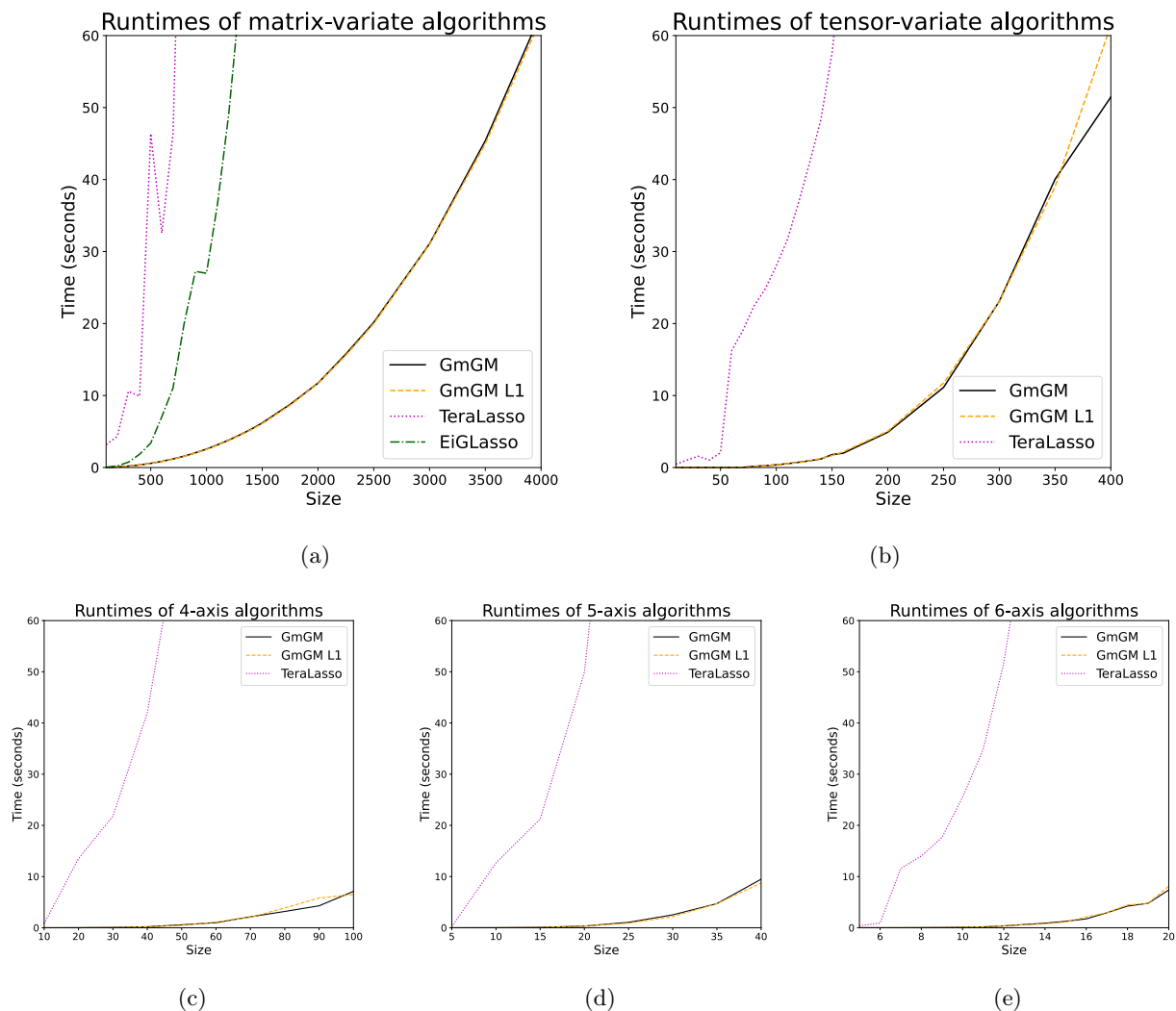


Figure 5: Runtimes for 2-axis (a) 3-axis (b) 4-axis (c) 5-axis (d) and 6-axis (e) data, whose true graphs come from the Erdos-Renyi distribution with 2% edge weights.

in the text file [on their github page for the algorithm](#)

#### 4.4.2 Experiment Justification

We chose this dataset as it had been considered by multi-axis methods before (albeit in a non-quantitative way) and had a relatively clear task associated with it (cell cycle stage clustering).

#### 4.4.3 Results - Thresholding Methods

We experimented with various thresholding methods on this dataset. One type of thresholding is global, in which one knocks out all edges whose magnitude is below a certain value. We found that this meant that clusters that tended to have high edge values were preserved relative to clusters who, while distinct, had a lower average edge value. One attempt to fix this was by thresholding per row; this has the interpretation of keeping the top  $n$  edges per vertex. Per-row thresholding mitigated this problem for some datasets (such as the 10x Genomics one), but not this dataset. The main problem was that some vertices tended to have higher edges than others - this meant that low-value vertices that were connected to a high-value vertex would always keep their connections to it, even if those connections were not very important from the perspective of the high-value vertex. To address this, we used a strategy of normalizing each column to sum to one before thresholding per row.



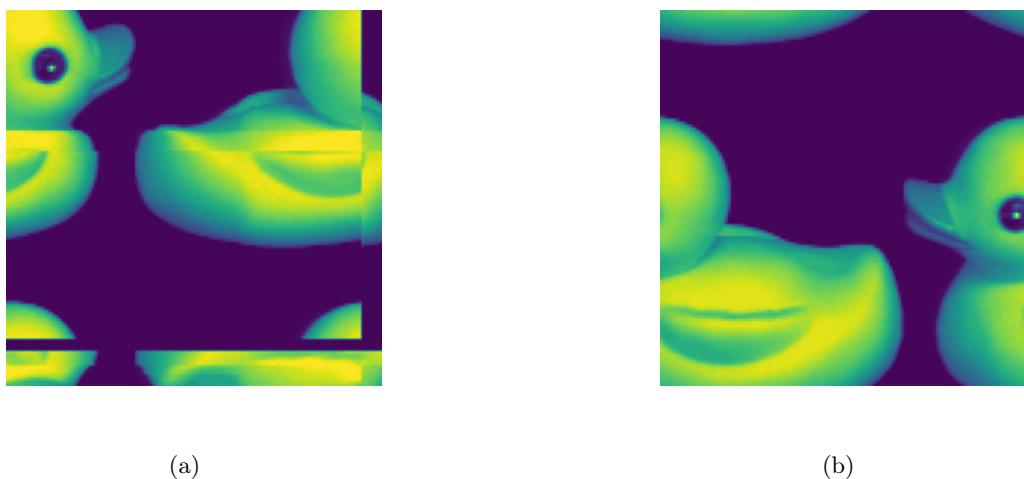


Figure 6: A reconstruction of the COIL-20 duck video after shuffling the rows, columns, and frames, using GmGM. (a) without the nonparanormal skeptic. (b) with the nonparanormal skeptic.

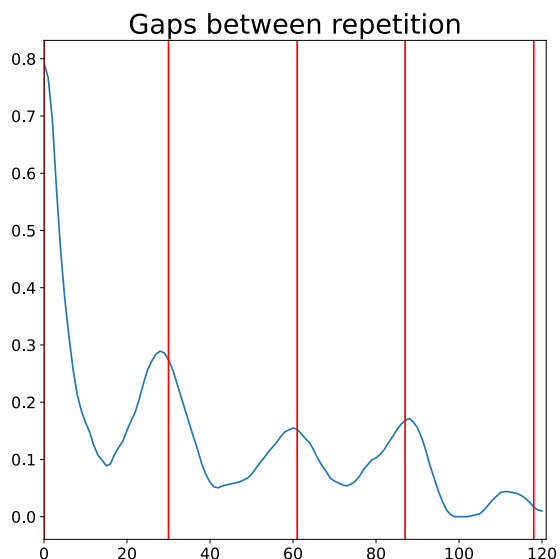


Figure 7: An example heartbeat offset detection, from EchoNet-Dynamic video 0XFE6E32991136338. The blue curve represents our Gaussian-blurred diagonal mass (if  $x=10$ , it represents the blurred mass of the 10th diagonal to the right of the main diagonal). The red lines represent the predicted peaks via a continuous wavelet transform peak detection algorithm. These represent offsets from the first mitral valve opening. For this video, the mitral valve opened on frame 17 and our first offset was on the 30th diagonal. Hence, we would predict the second mitral valve opening to occur at frame 47 (which, in this case, was correct).

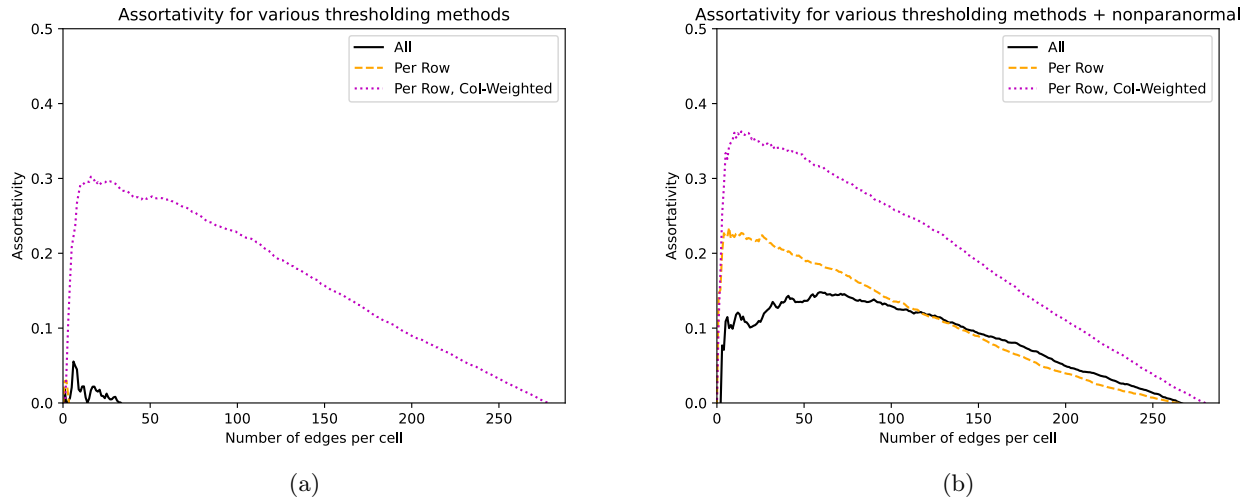


Figure 8: (a) Various thresholding methods with log-transformed data. (b) Various thresholding methods with nonparanormal-skeptic-transformed data. In both cases, we did not center the data.

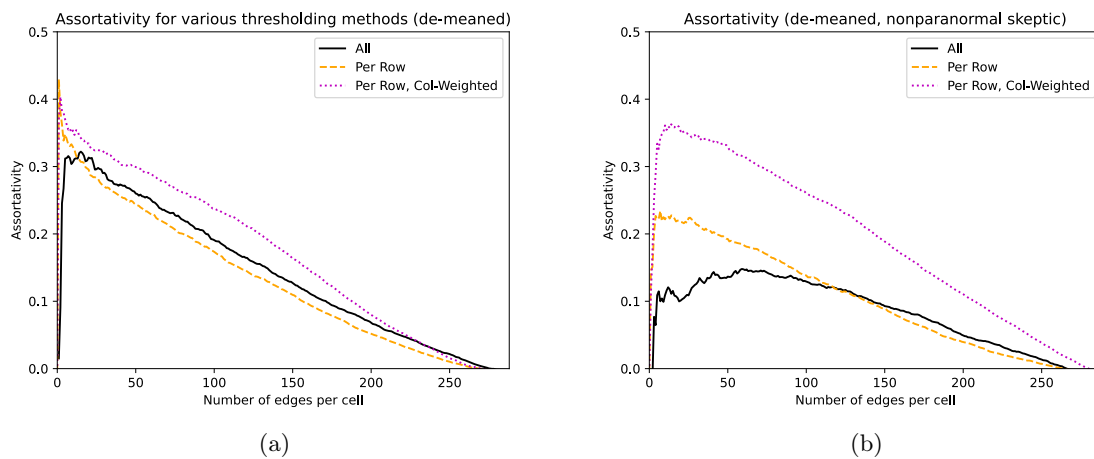


Figure 9: Assortativity of the various methods once we center the data. (a) log-transformed (b) nonparanormal skeptic.

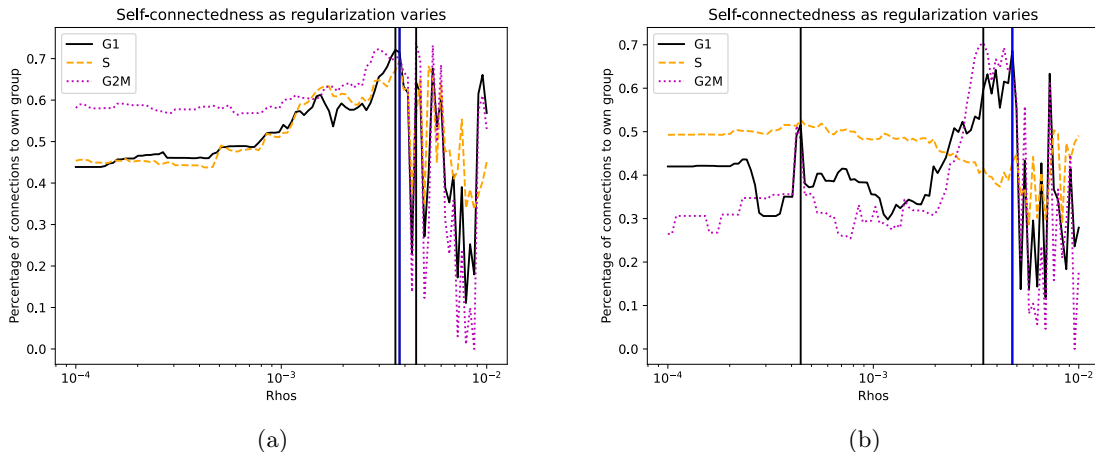


Figure 10: The percentage of within-group connections, varied as the regularization parameter increases. (a) with a log transform (b) with the nonparanormal skeptic. Vertical lines represent the best values for each class; the blue vertical line represents the best value overall.

From Figure 8 we can see that, with uncentered data, the third thresholding method is clearly better. In fact, without the nonparanormal skeptic, the other methods are practically worthless. However, if we centered the data as described in Section 7, all methods perform similarly; the third is still best. This makes sense, as the model assumes zero-mean data. Performance doesn't change under the nonparanormal transformation if we center the data, as centering is a monotonic transformation. Notably, this means that the nonparanormal skeptic does not always yield the best performance. We can see this in Figure 9

#### 4.4.4 Results - Regularization

We wanted to check whether our restricted L1 regularizer would improve performance. To choose the best regularization parameter, we devised the following test:

1. Run GmGM with regularization parameter  $\rho$ .
2. Threshold according to the third method (normalize the columns, then pick the top  $n$  cells from each row). We let  $n = 1$  as we wanted to optimize the parameter for the very sparse case.
3. Repeat for a range of values of  $\rho$ .
4. Measure the percentage of within-group connections. I.e, if 30% of the connections from cells in S stage were to other cells in S stage, we would report 30% for this metric.
5. Pick the  $\rho$  that optimizes the average percentage of within-group connections over all groups. Run GmGM with that  $\rho$ .
6. Calculate the assortativities over all thresholding methods.

We chose this setup as it is a distinct but related task to assortativity. Its results can be seen in Figure 10. The best parameters were around 0.0038 for the log-transformed data, and 0.0048 when using the nonparanormal skeptic. Figure 11 demonstrates the advantage of regularization.

#### 4.4.5 Results - Comparison with EiGLasso

We compare the assortativities of the best GmGM L1 parameters we discovered in the last section with the assortativities of EiGLasso as we varied its regularization parameter. We found a surprisingly clear advantage when using our method. This was unexpected, as our method and EiGLasso are quite similar, differing only in how we enforce sparsity. Judging by the 'chink' in EiGLasso's curve (Figure 12) it could be that EiGLasso simply

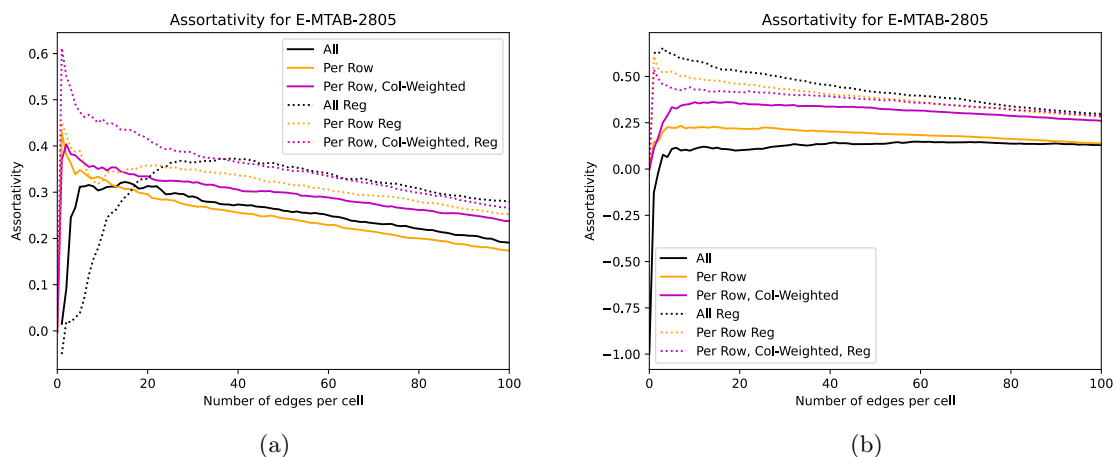


Figure 11: Assortativity for GmGM L1, (a) with a log transform and (b) with the nonparanormal skeptic. Note that the scales between the two plots are different.

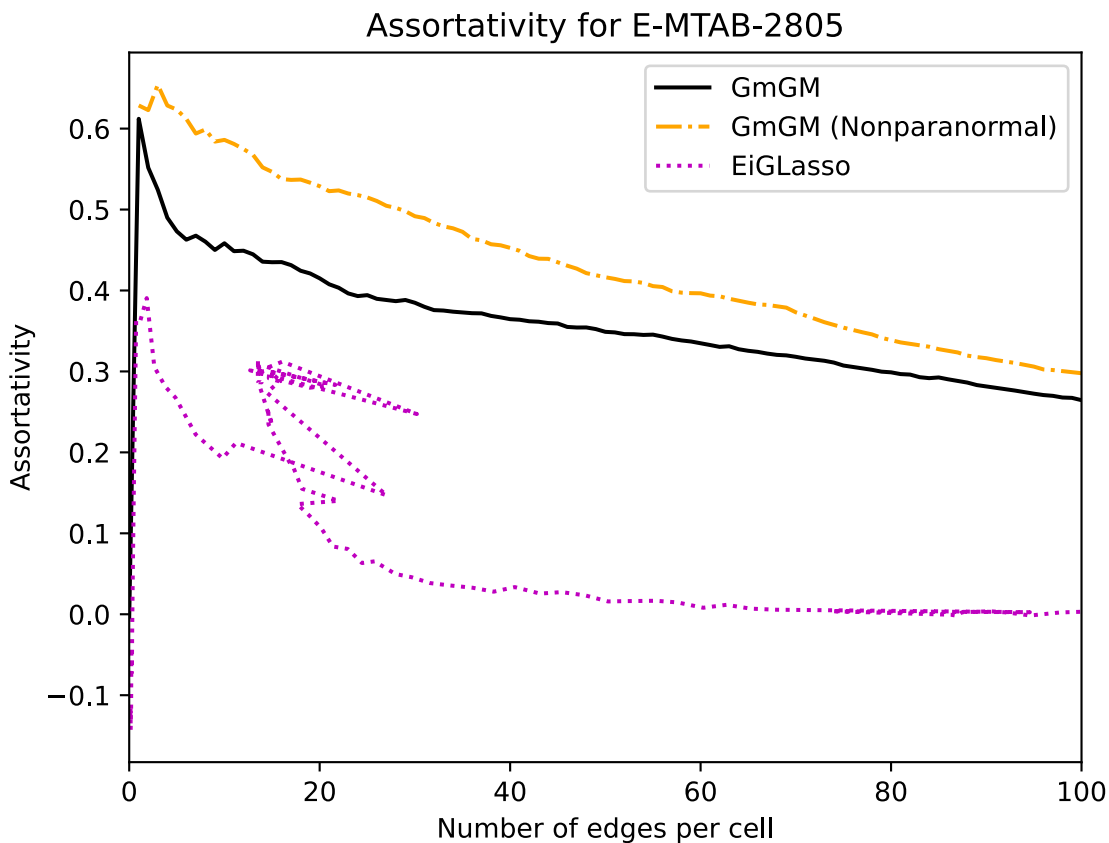


Figure 12: A comparison of GmGM with and without the nonparanormal skeptic, and EiGLasso. As EiGLasso was much slower than GmGM, we did not rerun the calculation for it with the nonparanormal skeptic.

Algorithm	Preprocessing	Runtime
GmGM	Log Transform, Centered	0.0079 seconds
GmGM	Nonparanormal Skeptic	0.043 seconds
EiGLasso	Log Transform, Centered	30 seconds
EiGLasso	Nonparanormal Skeptic	108 seconds

Table 6: Runtimes of our algorithm and EiGLasso, with various preprocessing techniques, on the E-MTAB-2805 dataset. Runtimes given are an average over 10 runs.

needed more time to converge to a good result. However, generating this curve for EiGLasso already took a long time, so we did not investigate further. To compare runtimes, see Table 6. We did not perform a UMAP consistency plot, as cells in UMAP-space formed a homogenous blob.

## 4.5 LifeLines-DEEP metagenomics + metabolomics

### 4.5.1 The Dataset

We used the LifeLines-DEEP metagenomics and metabolomics datasets (Tigchelaar et al., 2015). We did not do any pre-processing to the metabolomics, and we used the already pre-processed version of the metagenomics data from Prost et al. (2021). This dataset is available on the European Genome-Phenome Archive (EGA) under Study ID EGAS00001001704; to access it, one has to agree to an LL-DEEP-specific data access agreement.

### 4.5.2 Experiment Justification

We chose this dataset as it had been considered by prior single-axis work (ZiLN; Prost et al. (2021)), as well as being multi-omic. It had a more complicated class structure (taxonomy) than the E-MTAB-2805 dataset, so it was a natural next step in our investigations of the performance of this algorithm.

### 4.5.3 Results

We kept only patients that appeared in both datasets. We compared our model’s results to the model given by Prost et al. (2021) in the main paper. We report runtimes in Table 7 and give a UMAP consistency plot as in Figure 13. We can see that our algorithm is the fastest on the metagenomics dataset, even outperforming a single-axis method. Furthermore, our results seem quite sensible in UMAP-space.

Finally, we compared our assortativity to that of the Zero-Inflated Log Normal model by Prost et al. (2021). We found that our method performed similarly to theirs; see Figure 14.

Finally, we experimented with incorporating priors in our model. It is not unreasonable to find oneself in a situation in which one knows the highest-level taxonomic categorization of an organism (its phylum), but not know its lower-level categorizations. Thus, we chose a Wishart prior whose covariance matrix encoded the adjacency matrix of the following graph:

1. If two species are in the same phylum, they are connected
2. If two species are in a different phylum, they are not connected.

This *vastly* improves performance (Figure 15a). One might worry that the algorithm is just ‘memorizing’ the prior, and that the lower taxonomic ranks are improving because no genera are cross-phylum, for example. To

Algorithm	Axes	Speed	Per-Axis Speed
ZiLN	Species	3.2	3.2
GmGM	Species, People	2.59	1.30
GmGM	Species, People, Metabolites	22.18	7.39
TeraLasso	Species, People	1299.33	649.67

Table 7: Runtimes of various algorithms on the LifeLines-DEEP dataset. Speed is measured in seconds

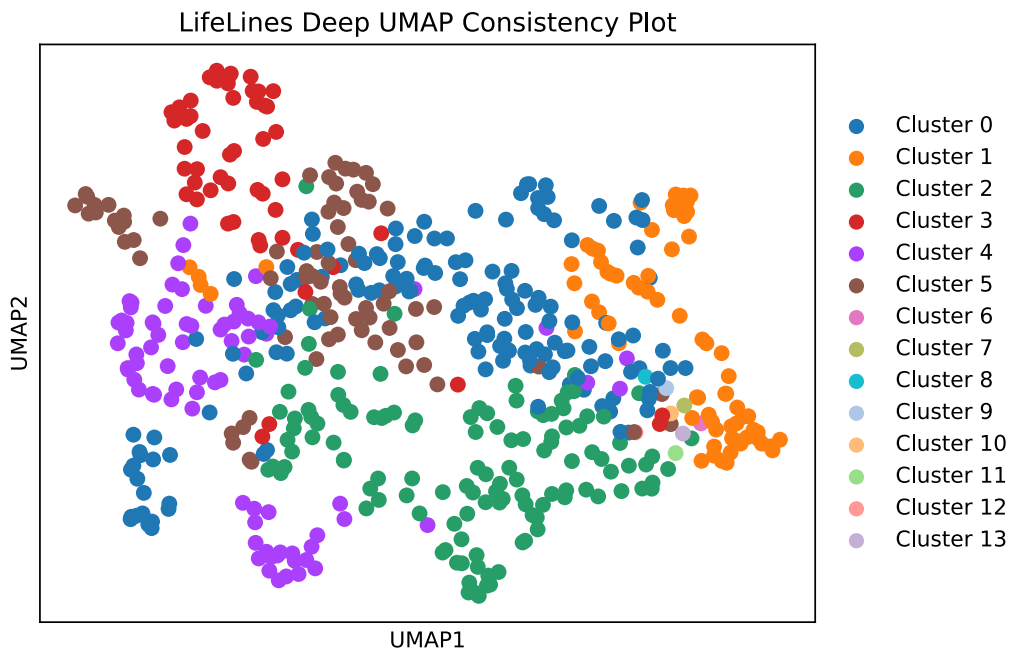


Figure 13: UMAP consistency plot for our algorithm applied to the LifeLines-DEEP dataset. The UMAP was constructed from log-transformed metagenomics data; the coloring represents a Louvain clustering of the output of GmGM performed on the centered log-transformed metagenomics+metabolomics data, thresholded by keeping the top 7% of edges overall.

show that this is not the case, we conduct a quick experiment with synthetic data, in which we feed the true graph in as the prior. If our algorithm were memorizing priors, then we would expect performance with the prior to be perfect. However, what we instead see is a huge but imperfect gain in performance (Figure 15b).

## 4.6 10x Genomics flash frozen lymph nodes

### 4.6.1 The Dataset

For this experiment, we looked at a single-cell RNA-sequencing+ATAC-sequencing dataset from 10x Genomics (10x Genomics, 2021). It is available here. The dataset is publicly available, but one does need to fill in a brief form for data collection purposes. They do not specify a license.

### 4.6.2 Experiment Justification

It is a realistically-sized multi-omics dataset, and by far the largest considered in this paper. It is so large that the use of prior work becomes infeasible, so demonstrating that our algorithm works on it would represent a significant leap forward. The dataset is unlabeled, so we evaluated our algorithm by means of a UMAP consistency plot.

### 4.6.3 Results

Before performing the experiment, we removed cells whose library size was three median absolute deviations from the median, and similarly removed genes and peaks if the the total amount of times they were expressed was three median absolute deviations from the median. In our output graphs, we kept the top 5 edges per vertex. We can see the UMAP consistency plot in Figure 17 and the runtimes in Table 8.

In Figure 17, we can see that clusters 0, 1, 3, and 4 all inhabit the large contiguous region in the lower left. Clusters 2 and 7 inhabit the contiguous region in the lower right. In the top right, there are three regions; 5, 6, and 8 (in order of left to right). Figure 20 implies that clusters 5 and 6 have some similarities, whereas cluster 8 is largely different from all the other clusters. A GO term analysis shows cluster 8 is associated with blood

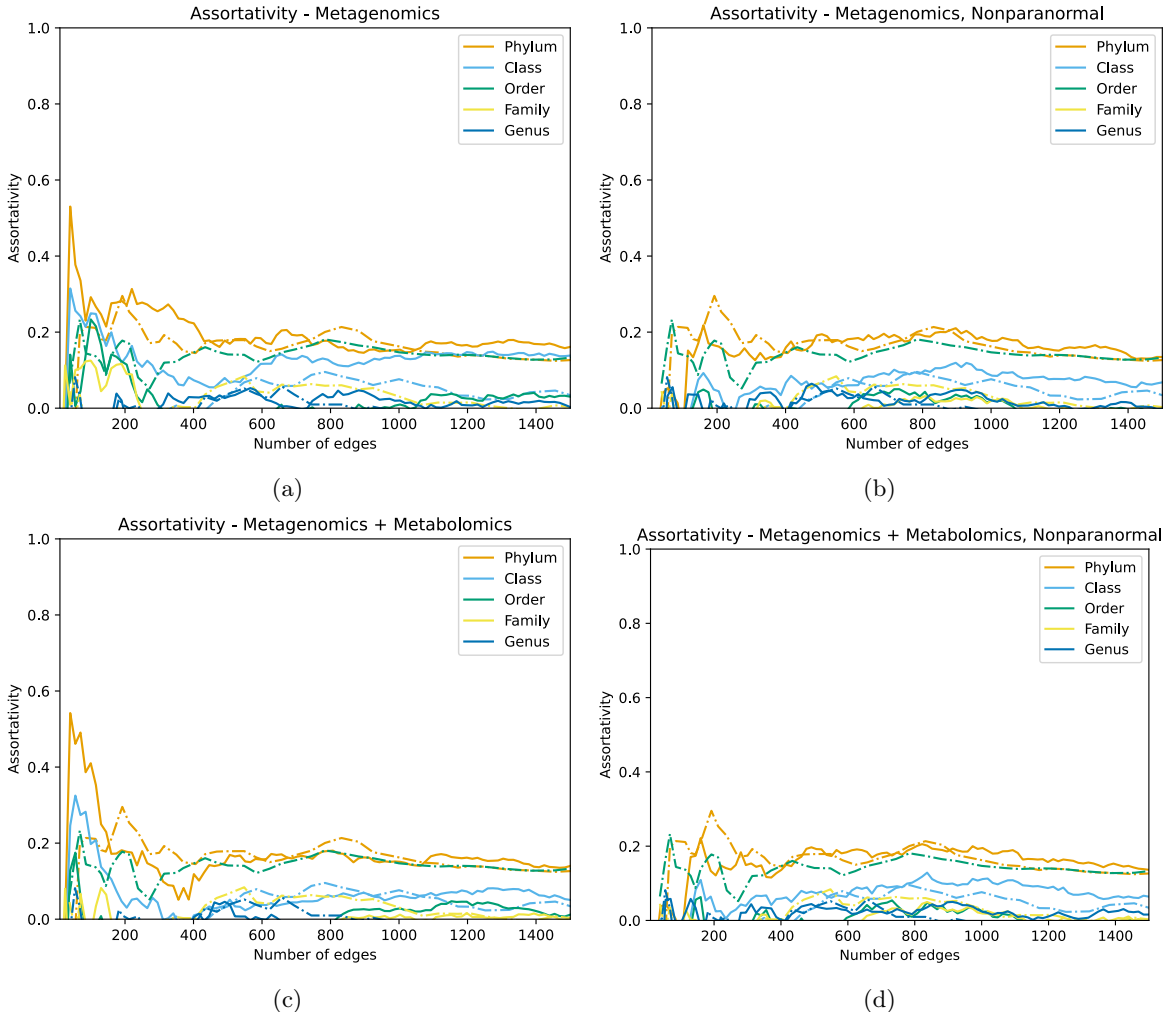


Figure 14: Comparison of our method (solid lines) to that of Prost et al. (2021) (dashed lines). (a) Just metagenomics, log-transformed. (b) Metagenomics using the nonparanormal skeptic. (c) The full multi-omic dataset, log-transformed. (d) The full multi-omic dataset using the nonparanormal skeptic.

coagulation and the integrin signalling pathway, whereas none of the other clusters have such an association. All other clusters, except 2 and 7, are highly associated with B cell activation-related genes. Cluster 2 is distinct in that it is related to the CCKR signalling map and the apoptosis signalling pathways. Cluster 7 was too small to perform the analysis. Full details of the significant go terms are available on the github repository.

## 5 REGULARIZATION

As remarked in the main paper, our algorithm by default includes no regularization. This is because our algorithm leverages the fact that we have a closed-form expression for the eigenvectors of the maximum likelihood estimate to avoid costly eigendecompositions every iteration. We do not have a closed-form expression for the eigenvectors in the regularized case.

Nevertheless, we can add regularization to the eigenvalue estimation by performing an eigenrecomposition and regularizing that. Eigenrecomposition requires a matrix multiplication, which is quite costly compared to the cost of an unregularized iteration - both in practice, and asymptotically in the matrix-variate case (matrix multiplication is  $O(\sum_{\ell} d_{\ell}^3)$  whereas an unregularized iteration is  $O(\prod_{\ell} d_{\ell})$ ). Thus, to regularize we first let our algorithm converge to the MLE before considering the penalty term. This allows us to avoid a major increase in runtime; our regularized algorithm runs in roughly the same time as the unregularized one (Figure 21a).

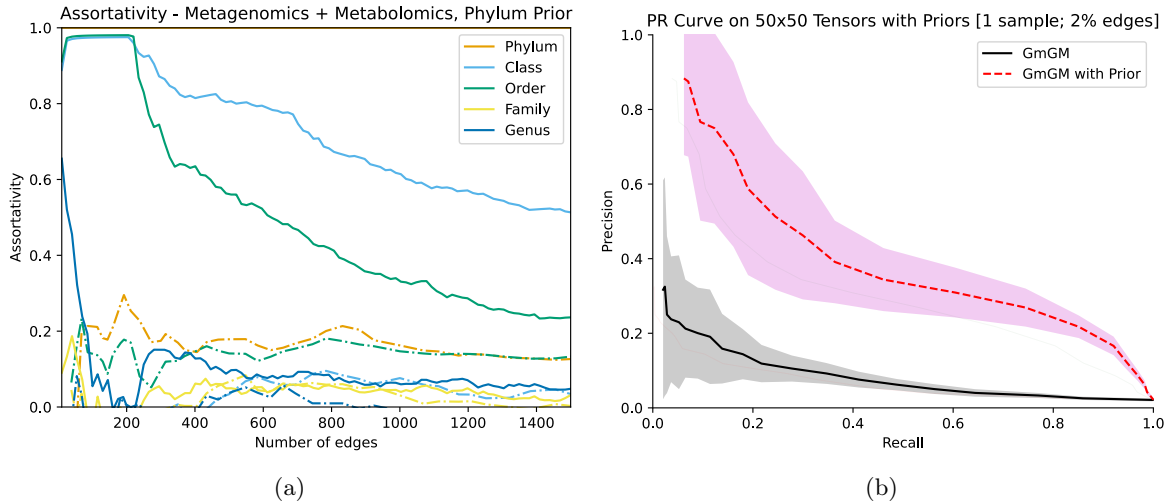


Figure 15: (a) The assortativity when incorporating prior knowledge. (b) A test on 50x50 1-sample synthetic data with 2% edges connected, with the true graphs fed in as the parameter to a Wishart prior to our algorithm.

Algorithm	Preprocessing	Runtime
GmGM	Log transform, just RNA	52 seconds
GmGM	Log transform, just ATAC	607 seconds
GmGM	Log transform, both modalities	590 seconds
EiGLasso	Log transform, just RNA	>60,000 seconds

Table 8: Runtimes of our algorithm and EiGLasso, with various preprocessing techniques, on the 10x Genomics data.

It is important to note that this estimator is slightly different than the standard Lasso estimator, as the standard estimator would minimize  $\|\Psi_\ell\|_1$  and our estimator minimizes  $\|\hat{\mathbf{V}}_\ell \mathbf{\Lambda}_\ell \hat{\mathbf{V}}_\ell^T\|_1$  (where only the eigenvalues  $\mathbf{\Lambda}_\ell$  are free to vary). This restriction prevents it from being able to drive elements exactly to zero, so thresholding is still needed afterwards. It can be derived as follows:

$$\frac{\partial}{\partial \lambda_i} \|\mathbf{V} \mathbf{\Lambda} \mathbf{V}^T\|_1 = \frac{\partial}{\partial \lambda_i} \left\| \sum_j \lambda_j v_{ja} v_{bj} \right\|_1 \quad (55)$$

$$= \left[ \frac{\partial}{\partial \lambda_i} \left| \sum_j \lambda_j v_{ja} v_{bj} \right| \right]_{ab} \quad (56)$$

$$= \left[ \frac{\partial}{\partial \lambda_i} \text{sign} \left[ \sum_j \lambda_j v_{ja} v_{bj} \right] v_{ia} v_{bi} \right]_{ab} \quad (57)$$

$$= \left[ \text{sign} \left[ \mathbf{V} \mathbf{\Lambda} \mathbf{V}^T \right]_{ab} v_{ia} v_{bi} \right]_{ab} \quad (58)$$

$$= \mathbf{v}_i^T \text{sign} \left[ \mathbf{V} \mathbf{\Lambda} \mathbf{V}^T \right] \mathbf{v}_i \quad (59)$$

Despite this difference, it performs comparably to prior work. We show in Figure 21b the precision-recall curves for the 3-axis case, and observe that it performs almost perfectly. This is notable as it was the case that the unregularized algorithm performed worse than prior work.

To test whether regularization improved the algorithm in the real world, we performed another test on the E-MTAB-2805 dataset. We first looked at the metric of ‘percentage of connections to own group’. We varied the



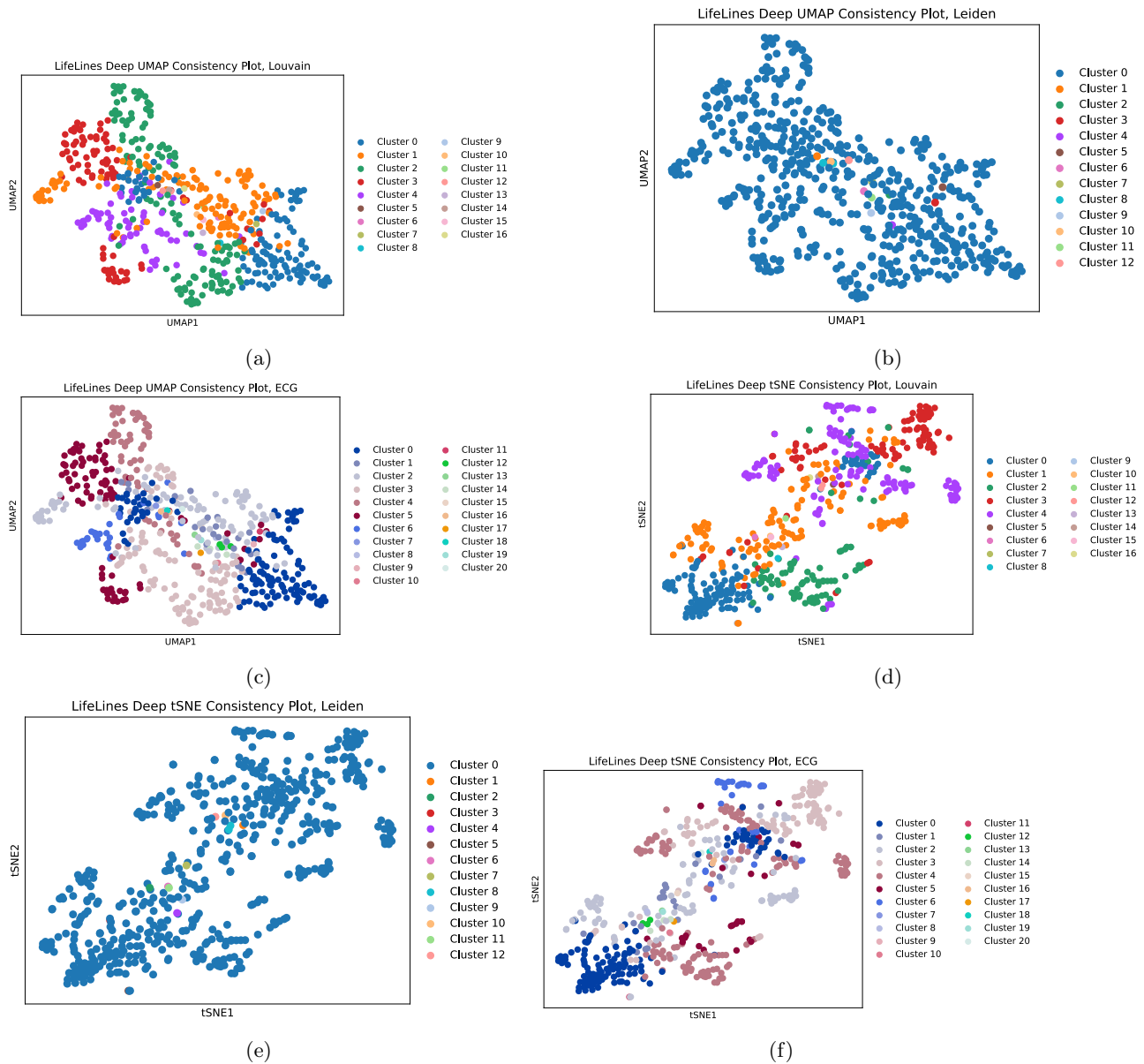


Figure 16: UMAP and tSNE consistency plots with Louvain, Leiden, and ECG clustering algorithms on the Lifelines-Deep data.

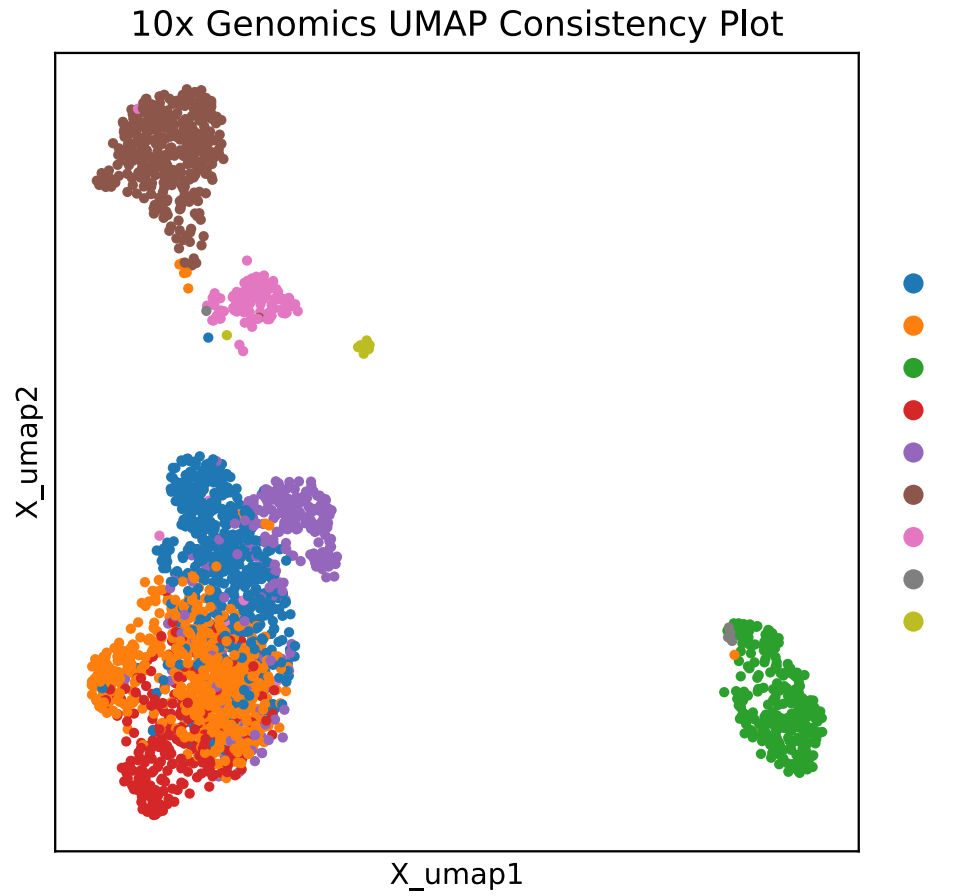


Figure 17: A UMAP consistency plot for the 10x Genomics dataset. Both UMAP and GmGM were used on the full multi-omic log-transformed dataset. The coloring represents a Louvain clustering on the output graph of GmGM, with the top 5 edges per vertex being kept after column normalization (the third thresholding method from Section 4.4).

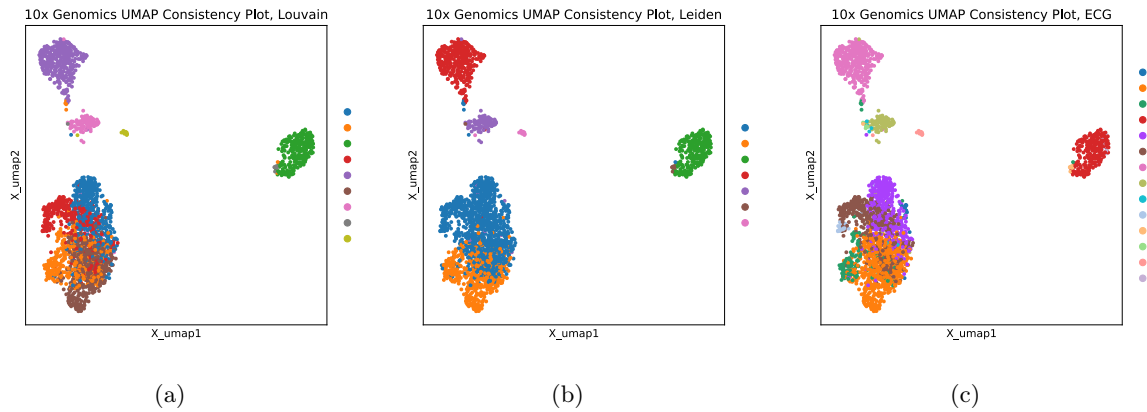


Figure 18: UMAP consistency plots with Louvain, Leiden, and ECG clustering algorithms on the 10x Genomics data.

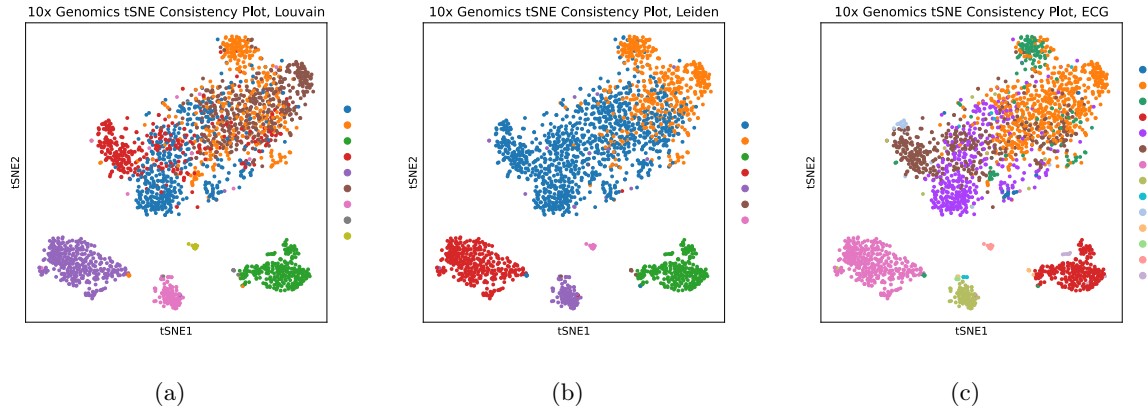


Figure 19: tSNE consistency plots with Louvain, Leiden, and ECG clustering algorithms on the 10x Genomics data.

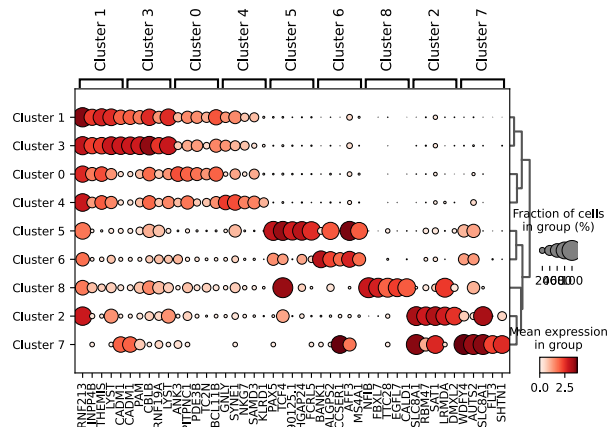


Figure 20: Expression of top differentially-expressed genes by cluster.

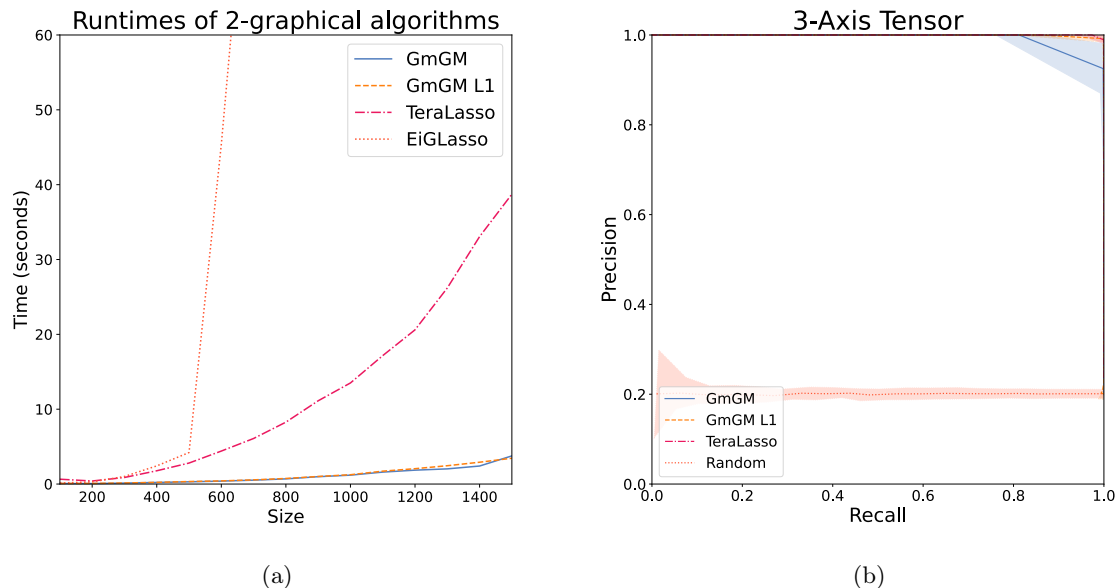


Figure 21: (a) Runtimes of our algorithm and prior work on matrix-variate data. Our regularized algorithm is denoted “GmGM L1”, and takes about the same time as the unregularized “GmGM”. (b) Precision-recall curves for tensor-variate data. TeraLasso and our regularized “GmGM L1” perform almost perfectly.

regularization parameter, and for each estimated graph we thresholded the graph to keep only the largest edge per cell. We plotted the results in Figure 22a

The optimal parameter arises right near the end of the range of considered regularization parameters; when we considered larger parameters, we entered a region of instability in which convergence took far longer and results became sporadic (Figure 22b). We are not sure what causes this, but hypothesize that our restricted regularizer no longer has the degrees of freedom to give useful information to the problem, rather making it harder to find a good solution. The fact that the optimal parameters occur specifically at the end of the stable region implies that the algorithm would likely have benefited from a non-restricted regularizer - although our restricted regularizer is also definitively better than no regularizer at all.

Our second test, shown in Figure 23, was to take the optimal regularizing parameter found in the first test, and perform the same assortativity test as was performed in Section 4.4. We can see that the regularized methods tend to outperform the unregularized methods.

## 6 ASYMPTOTIC COMPLEXITY

Our space complexity is the optimal  $O(\sum_{\ell} d_{\ell}^2)$ , as was all prior work except the original BiGLasso. For computational complexity, refer to Table 9. In the reported values, we treat matrix multiplication as having cubic complexity. Note that Gram matrix computation is an  $O(Kd^{K+1})$  operation, and that our algorithm has the asymptotically fastest iterations in the matrix-variate case ( $O(d^2)$ ). If we ignore the per-axis sizes  $d$  and instead consider the size of the whole dataset  $d_{\vee} = d^K$ , we can note that the runtime of the tensor-variate algorithms is almost linear in the size of the dataset.

## 7 CENTERING THE DATA

Multi-axis models such as ours typically assume a mean of 0 for the data. When one has multiple samples, this is not a problem - one can just center the data. However, with multi-axis methods we typically only have one sample. Because the nonparanormal skeptic is invariant to monotone transformations, this is a viable method to circumvent this problem. When not using that, we would take the average value of the tensor (averaged across all elements in the tensor) and subtract this value from every element in the tensor.

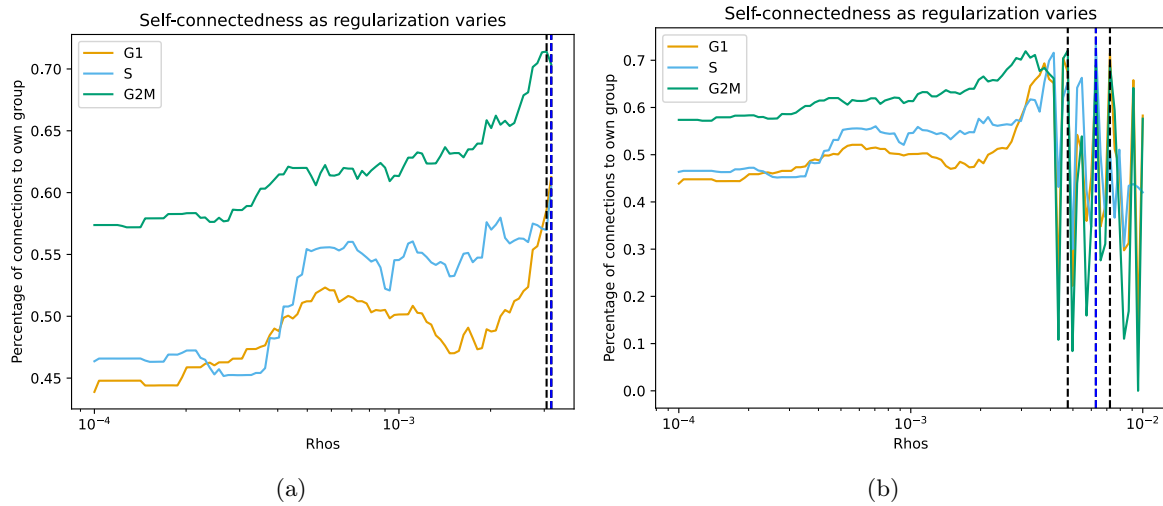


Figure 22: (a) Self-connectedness of each of the three cell cycle stages as regularization parameter varies. Vertical bars represent the optimal parameters; one for each of the parameters (there is overlap). (b) The same plot as on the left, with the range extended to show the region of instability.

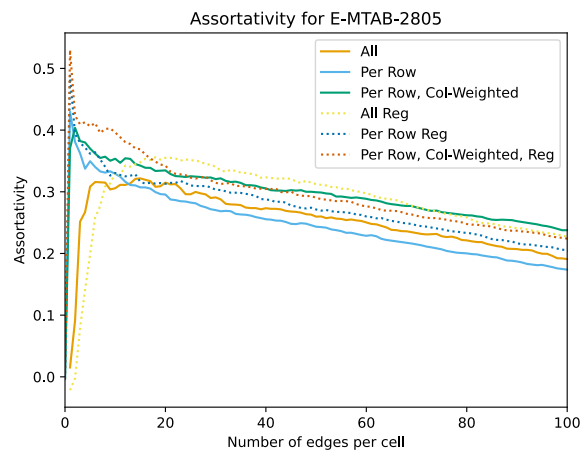


Figure 23: Assortativity as the number of kept edges per cell varies.

Algorithm	Range of $K$	Overall Complexity	Per-Iteration Complexity
BiGLasso	$K = 2$	$O(Kd^4)$	$O(Kd^4)$
EiGLasso	$K = 2$	$O(Kd^3)$	$O(Kd^3)$
TeraLasso	Any	$O(Kd^{K+1})$	$O(Kd^3 + d^K)$
GmGM	Any	$O(Kd^{K+1})$	$O(d^K)$
GmGM L1	Any	$O(Kd^{K+1})$	$O(Kd^3 + d^K)$

Table 9: Computational complexities of selected algorithms. ‘GmGM L1’ refers to GmGM equipped with our restricted L1 penalty; note that GmGM L1 first converges to the unregularized solution (using the cheap iterations of GmGM) before it adds in the regularizer. For simplicity, all dimensions  $d_\ell$  are considered to be the same size  $d$ .

## References

- 10x Genomics. (2021, May). Flash-Frozen Lymph Node with B Cell Lymphoma (14k sorted nuclei). Retrieved May 10, 2023, from <https://www.10xgenomics.com/resources/datasets/fresh-frozen-lymph-node-with-b-cell-lymphoma-14-k-sorted-nuclei-1-standard-2-0-0>
- Buettner, F., Natarajan, K. N., Casale, F. P., Proserpio, V., Scialdone, A., Theis, F. J., Teichmann, S. A., Marioni, J. C., & Stegle, O. (2015). Computational analysis of cell-to-cell heterogeneity in single-cell RNA-sequencing data reveals hidden subpopulations of cells [Number: 2 Publisher: Nature Publishing Group]. *Nature Biotechnology*, *33*(2), 155–160. <https://doi.org/10.1038/nbt.3102>
- Dahl, A., Hore, V., Iotchkova, V., & Marchini, J. (2013, December). Network inference in matrix-variate Gaussian models with non-independent noise [arXiv:1312.1622 [stat]]. <https://doi.org/10.48550/arXiv.1312.1622>
- Du, P., Kibbe, W. A., & Lin, S. M. (2006). Improved peak detection in mass spectrum by incorporating continuous wavelet transform-based pattern matching. *Bioinformatics*, *22*(17), 2059–2065. <https://doi.org/10.1093/bioinformatics/btl355>
- Greenewald, K., Zhou, S., & Hero III, A. (2019, September). Tensor Graphical Lasso (TeraLasso) [arXiv:1705.03983 [stat]]. Retrieved February 24, 2023, from <http://arxiv.org/abs/1705.03983>
- Kalaitzis, A., Lafferty, J., Lawrence, N. D., & Zhou, S. (2013). The Bigraphical Lasso [ISSN: 1938-7228]. *Proceedings of the 30th International Conference on Machine Learning*, 1229–1237. Retrieved February 24, 2023, from <https://proceedings.mlr.press/v28/kalaitzis13.html>
- Kolda, T. G., & Bader, B. W. (2009). Tensor Decompositions and Applications. *SIAM Review*, *51*(3), 455–500. <https://doi.org/10.1137/07070111X>
- Lewis, A. S. (1996). Derivatives of Spectral Functions [Publisher: INFORMS]. *Mathematics of Operations Research*, *21*(3), 576–588. <https://doi.org/10.1287/moor.21.3.576>
- Li, S., López-García, M., Lawrence, N. D., & Cutillo, L. (2022, March). Scalable Bigraphical Lasso: Two-way Sparse Network Inference for Count Data [arXiv:2203.07912 [cs, stat]]. Retrieved February 24, 2023, from <http://arxiv.org/abs/2203.07912>
- Mazumder, R., & Hastie, T. (2012). Exact Covariance Thresholding into Connected Components for Large-Scale Graphical Lasso. *Journal of machine learning research : JMLR*, *13*, 781–794. Retrieved October 9, 2023, from <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4225650/>
- Nene, S. A., Nayar, S. K., & Murase, H. (n.d.). Columbia Object Image Library (COIL-20).
- Ouyang, D., He, B., Ghorbani, A., Yuan, N., Ebinger, J., Langlotz, C. P., Heidenreich, P. A., Harrington, R. A., Liang, D. H., Ashley, E. A., & Zou, J. Y. (2020). Video-based AI for beat-to-beat assessment of cardiac function [Number: 7802 Publisher: Nature Publishing Group]. *Nature*, *580*(7802), 252–256. <https://doi.org/10.1038/s41586-020-2145-8>
- Prost, V., Gazut, S., & Brüls, T. (2021). A zero inflated log-normal model for inference of sparse microbial association networks [Publisher: Public Library of Science]. *PLOS Computational Biology*, *17*(6), e1009089. <https://doi.org/10.1371/journal.pcbi.1009089>
- Tigchelaar, E. F., Zhernakova, A., Dekens, J. A. M., Hermes, G., Baranska, A., Mujagic, Z., Swertz, M. A., Muñoz, A. M., Deelen, P., Cénit, M. C., Franke, L., Scholtens, S., Stolk, R. P., Wijmenga, C., & Feskens, E. J. M. (2015). Cohort profile: LifeLines DEEP, a prospective, general population cohort study in the northern Netherlands: Study design and baseline characteristics [Publisher: British Medical Journal Publishing Group Section: Epidemiology]. *BMJ Open*, *5*(8), e006772. <https://doi.org/10.1136/bmjopen-2014-006772>