

Clusterpath Gaussian Graphical Modeling

Daniel J.W. Touw¹, Andreas Alfons¹, Patrick J.F. Groenen¹, and Ines Wilms²

¹Department of Econometrics, Erasmus University Rotterdam

²Department of Quantitative Economics, Maastricht University

Abstract

Graphical models serve as effective tools for visualizing conditional dependencies between variables. However, as the number of variables grows, interpretation becomes increasingly difficult, and estimation uncertainty increases due to the large number of parameters relative to the number of observations. To address these challenges, we introduce the *Clusterpath estimator of the Gaussian Graphical Model* (CGGM) that encourages variable clustering in the graphical model in a data-driven way. Through the use of an aggregation penalty, we group variables together, which in turn results in a block-structured precision matrix whose block structure remains preserved in the covariance matrix. The CGGM estimator is formulated as the solution to a convex optimization problem, making it easy to incorporate other popular penalization schemes which we illustrate through the combination of an aggregation and sparsity penalty. We present a computationally efficient implementation of the CGGM estimator by using a cyclic block coordinate descent algorithm. In simulations, we show that CGGM not only matches, but oftentimes outperforms other state-of-the-art methods for variable clustering in graphical models. We also demonstrate CGGM’s practical advantages and versatility on a diverse collection of empirical applications.

Keywords: hierarchical clustering, precision matrix, covariance matrix, block structure, unsupervised learning

1 Introduction

Gaussian graphical models (GGM) are popular tools for summarizing conditional dependencies among p variables. A GGM is a conditional dependency network where one refers to the variables as the *nodes* and the *edges* represent the conditional dependency relations among each pair of variables. Estimating GGMs is statistically challenging when the number of parameters ($p(p+1)/2$) is large relative to the sample size (n), leading to large estimation variability. Yet, such settings arise across many, diverse fields which has led to a flourishing area on regularized GGM estimation (e.g., Meinshausen & Bühlmann, 2006; Peng et al., 2009; Cai et al., 2011). While much of the existing literature focuses on reducing the estimation variability through edge sparsity, our approach branches into a different direction by using node-clustering. We integrate convex clustering into the GGM framework to reduce estimation variability through estimation pooling for similar variables.

Edge sparsity through, for instance, ℓ_1 -regularization has long formed the predominant choice to reduce dimensionality in GGMs (e.g., Yuan & Lin, 2007; Yuan, 2008; Friedman et al., 2008; Rothman et al., 2008). The idea is to sparsely estimate the precision matrix (i.e., the inverse of the covariance matrix)—the primary object of interest in GGMs—since conditional independencies between variable-pairs can be directly obtained from its sparsity pattern, or equivalently the absence of edges in the estimated GGM, offering an interpretability advantage. Several recent studies, however, point to important drawbacks when solely relying on edge sparsity as simplifying structure; namely weak detection capabilities in large-scale networks (Eisenach et al., 2020), interpretability issues for densely estimated GGMs with many variables (Grechkin et al., 2015), or inability to capture real world network structures such as hub nodes (Tarzanagh & Michailidis, 2018) or block-structured graphs (Colombi et al., 2024).

When GGMs face such challenges, one is often not interested in estimating conditional dependencies among the many observed variables but instead among a smaller number of (unobserved) *clusters* (also

called communities) of original variables that share the same behavior. For instance, biologists estimate large-scale gene regulatory networks and cluster genes into pathways to unravel dependencies among them (e.g., Mestres et al., 2018; Shan et al., 2020), neuroscientists analyzing fMRI data routinely cluster voxels into regions of interest to learn interaction networks of brain activation (e.g., Pircalabelu & Claeskens, 2020), financial analysts cluster company stocks into industry sectors to study how shocks can spread over the market by contagion (e.g., Wilms & Bien, 2022). Cluster analysis is one of the most popular unsupervised learning methods to discover the underlying group structures in data. Variable clustering in GGMs not only offers simple, interpretable dependency networks but may also boost the dependency signals (Eisenach et al., 2020).

To estimate GGMs with clustered variables, a recent yet growing interest arose in *node-based* dimensionality reduction. Initial proposals assume the clusters to be known *a priori* and incorporate this information in a regularization framework to encourage within-cluster over cross-cluster dependencies (e.g., Grechkin et al., 2015; Millington & Niranjana, 2019). Domain-knowledge may, however, not always be available to impose a grouping, thereby still calling for unsupervised clustering procedures. Subsequent works learn the node clustering by decoupling the clustering task from the estimation of the GGM (e.g., Ambroise et al., 2009; Tan et al., 2015; Eisenach et al., 2020; Brownlees et al., 2022; Shi et al., 2024). Such a two-step procedure may however lead to suboptimal results thereby giving rise to a third stream that considers both tasks jointly (e.g., Pircalabelu & Claeskens, 2020; Wilms & Bien, 2022). Yet, to the best of our knowledge, the potential to leverage penalty structures popular in the literature on convex clustering (e.g., Pelckmans et al., 2005; Hocking et al., 2011; Lindsten et al., 2011; Chi et al., 2017 or, Weylandt et al., 2020; Chakraborty & Xu, 2023 for more recent advances) to combine node clustering jointly with the estimation of the GGM is left largely unexplored, a notable exception being Yao & Allen (2019).

We fill this gap by developing a novel regularizer, called the *Clusterpath estimator of the Gaussian Graphical Model* (CGGM), to estimate GGMs that are node-clustered. Specifically, the cluster structure (i.e. the number of clusters and their composition) is identified jointly with the estimation of the parameters. To this end, we propose a novel penalty on the distances between variables in the precision matrix and embed this in a convex optimization framework for which we offer a computationally efficient cyclic block coordinate descent algorithm (Section 2). The resulting estimated precision matrix has a block structure in which all variables belonging to the same cluster share the same within- as well as cross-cluster dependencies. A unique property of our approach for clustering the precision matrix is that its inverse retains the same block structure, a property not shared by other approaches—including those discussed below. Indeed, popular existing paradigms either induce a simplicity structure on the precision matrix or the covariance matrix, and the induced structure is typically not maintained when taking the inverse of the object of interest. CGGM connects these paradigms by inducing a block structure in the precision matrix that is shared in its covariance matrix.

Our proposal is most closely related to Yao & Allen (2019), Pircalabelu & Claeskens (2020), and Wilms & Bien (2022). There are two key distinctions with Yao & Allen (2019): The first lies in the distance metric of the aggregation penalty: their approach does not account for the diagonal elements of the precision matrix, allowing the diagonal elements of clustered variables to differ. It is due to this property that the variable clustering is not retained when taking the inverse of the estimated precision matrix. Second, we show how the aggregation penalty of the CGGM estimator can be easily combined with other popular convex penalties such as a sparsity penalty. Furthermore, the procedure of Pircalabelu & Claeskens (2020) results in a precision matrix with a blockdiagonal structure rather than a full block structure, whereas the regularizer of Wilms & Bien (2022) requires side-information on the similarity of variables to guide node clustering.

Through a comprehensive simulation study, we evaluate the performance of CGGM against its closest benchmark methods for which software implementations are publicly available (Section 3). Our results indicate that CGGM frequently surpasses the benchmarks in both estimation accuracy and clustering performance. While the main focus of our study is on estimating clustered precision matrices to create graphical models, we also demonstrate that CGGM can be easily extended to estimate clustered covariance matrices (Section 4). In fact, when a block structure in the covariance matrix (instead of the precision matrix) is the object of interest, we demonstrate that directly estimating a clustered covariance matrix has practical advantages over inverting a clustered precision matrix estimate, even though the latter also yields a block structure in the covariance matrix. Finally, we illustrate the effectiveness and versatility of CGGM on three

practical applications involving (i) stock market data from the S&P 100, (ii) OECD well-being indicators, and (iii) survey data on participants' humor styles (Section 5).

2 The Clusterpath Estimator for the GGM

We begin in Section 2.1 by discussing GGMs with clustered variables, followed by the introduction of the clusterpath estimator in Section 2.2. Section 2.3 details the cyclic block coordinate descent algorithm used to compute our estimator.

2.1 Clustered GGMs

Let \mathbf{X} be an $n \times p$ matrix of n multivariate normal observations each of dimension p , with sample mean \mathbf{m} and sample covariance matrix \mathbf{S} . Denoting the population covariance matrix by $\mathbf{\Sigma}$, our target of estimation is the precision matrix $\mathbf{\Theta} = \mathbf{\Sigma}^{-1}$. Under the assumption of a multivariate normal distribution, the precision matrix can be equivalently expressed in a graph where the nodes represent the p variables and the edge weights are given by the entries in $\mathbf{\Theta}$ which represent the conditional dependencies among the variables.

Our goal is to estimate the precision matrix $\mathbf{\Theta}$ (and hence the graph structure of the GGM) by encouraging clustering of the p nodes in the graph. The K new cluster variables ξ_1, \dots, ξ_K then represent the average of the variables that belong to that cluster, with K typically much smaller than p to achieve dimensionality reduction in the parameters defining $\mathbf{\Theta}$. The edge weights among the clustered variables represent the conditional dependencies among the K new cluster variables. Clustering of the variables in the graph corresponds to a block structure in the rows and columns of $\mathbf{\Theta}$, see the so-called *G-block* format introduced in Bunea et al. (2020) and discussed by Wilms & Bien (2022) in the context of GGMs. In particular, for a partition $\{G_1, \dots, G_K\}$ of the variables $\{1, \dots, p\}$ and corresponding $p \times K$ cluster membership matrix \mathbf{U} with $u_{jk} = 1$ if variable j belongs to cluster k and zero otherwise, there exists a $K \times K$ symmetric matrix $\mathbf{R} = (r_{k\ell})_{1 \leq k, \ell \leq K}$, and a $p \times p$ diagonal matrix $\mathbf{A} = \text{diag}(a_{11}\mathbf{I}, \dots, a_{KK}\mathbf{I})$ such that the precision matrix can be written in *G-block* format given by

$$\begin{aligned} \mathbf{\Theta} &= \mathbf{U}\mathbf{R}\mathbf{U}^\top + \mathbf{A} \\ &= \begin{bmatrix} r_{11}\mathbf{1}\mathbf{1}^\top & r_{12}\mathbf{1}\mathbf{1}^\top & \dots & r_{1K}\mathbf{1}\mathbf{1}^\top \\ r_{21}\mathbf{1}\mathbf{1}^\top & r_{22}\mathbf{1}\mathbf{1}^\top & \dots & r_{2K}\mathbf{1}\mathbf{1}^\top \\ \vdots & \vdots & \ddots & \vdots \\ r_{K1}\mathbf{1}\mathbf{1}^\top & r_{K2}\mathbf{1}\mathbf{1}^\top & \dots & r_{KK}\mathbf{1}\mathbf{1}^\top \end{bmatrix} + \begin{bmatrix} a_{11}\mathbf{I} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & a_{22}\mathbf{I} & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & a_{KK}\mathbf{I} \end{bmatrix}, \end{aligned} \tag{1}$$

where \mathbf{I} is the identity matrix of appropriate dimension, similarly for $\mathbf{1}$ denoting a column-vector of ones and for $\mathbf{0}$ denoting a matrix of zeros. The within-cluster conditional variances $r_{kk} + a_{kk}$ and covariances r_{kk} are the same for all p_k variables within cluster k . These p_k variables in cluster k also have the same conditional covariance $r_{k\ell}$ with all p_ℓ variables belonging to another cluster ℓ .

A fundamental choice made throughout this paper is that $\mathbf{\Theta}$ should be positive definite. This holds if $\mathbf{R} + (\mathbf{U}^\top\mathbf{U})^{-1}\mathbf{A}$ is positive definite and $a_{kk} > 0$, for more details, see Appendix A. As a consequence of this choice, the range of partial correlations that can be modeled may be limited in certain cases. For example, when $K = 1$ and $\mathbf{\Theta} = r_{11}\mathbf{1}\mathbf{1}^\top + (1 - r_{11})\mathbf{I}$, then r_{11} is limited to the range $-1/(p - 1) < r_{11} < 1$ when requiring $\mathbf{\Theta}$ to be positive definite. Explicit ranges of allowed partial correlations are, however, difficult to formulate more generally for $K > 1$ since this depends on the cluster structure.

In contrast to Bunea et al. (2020); Wilms & Bien (2022), the block structure is not only reflected in the first part of decomposition (1) but also in the diagonal matrix \mathbf{A} . This means that we impose equal conditional variances of cluster members. An important implication of such an assumed block structure is that the clustering structure of $\mathbf{\Theta}$ is thereby retained when taking the inverse (e.g., Gower & Groenen, 1991; Archakov & Hansen, 2024). The approaches proposed by Yao & Allen (2019), Pircalabelu & Claeskens (2020), and Wilms & Bien (2022) do not display this property. Section 4 explores the estimation of block-structured covariance matrices using CGGM through a numerical experiment.

Next, the block-structured precision matrix can be equivalently visualized through a clustered GGM, see the toy example with $p = 8$ variables and $K = 3$ clusters in Figure 1. The block structure in $\mathbf{\Theta}$ can be

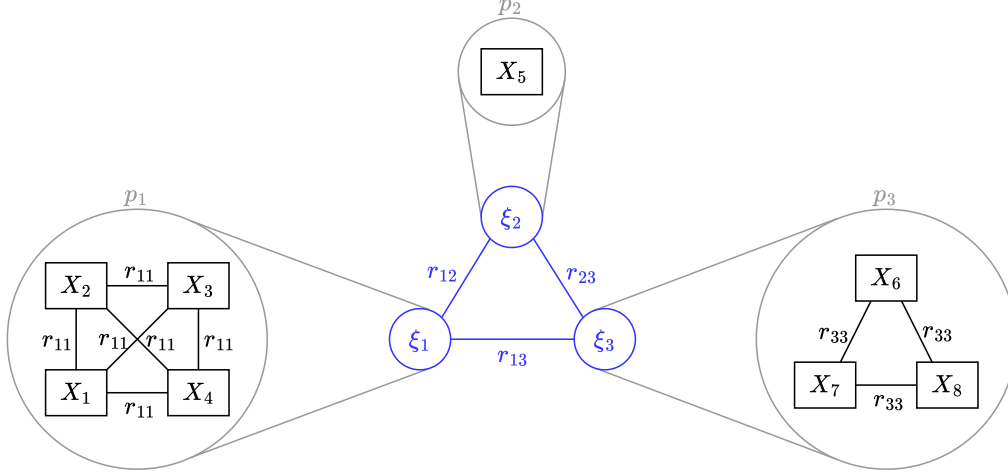


Figure 1: Toy example of a graph representing the clustered precision matrix with $K = 3$ clusters constructed from $p = 8$ variables. Cluster variable ξ_1 is the average of the $p_1 = 4$ variables X_1, X_2, X_3 , and X_4 having within-cluster conditional covariance r_{11} . Cluster variable ξ_2 is a singleton ($p_2 = 1$) and equal to the original variable X_5 . Cluster variable ξ_3 is the average of the $p_3 = 3$ variables X_6, X_7 , and X_8 having within-cluster conditional covariance r_{33} . The three cluster variables have conditional covariances r_{12}, r_{13} , and r_{23} .

interpreted as a clustering in the GGM of variables with identical conditional dependency structure. This cluster structure is a priori, however, unknown and our procedure (Section 2.2) jointly learns the clustering (including the number of clusters K and their composition) with the estimation of the GGM. At a coarse level, the clustered GGM visualizes the clustered variables as nodes and displays the conditional dependency structure (edges) among these—the central (blue) part in Figure 1. Since the cluster variables are the averages of its cluster members, this compact visualization implicitly represents the identical conditional dependency structure of all members of a particular cluster with all members of another cluster. At a more detailed level, the clustered GGM zooms into the members of the newly formed cluster variables—the outer (gray) parts in Figure 1. Cluster members have identical conditional covariances and variances (though the latter is not explicitly visualized in the figure).

Finally, the block structure of the precision matrix in equation (1) may display sparsity, where some of the entries in Θ are equal to zero. In particular, the absence of an edge in the clustered GGM between two cluster variables, for instance cluster variables one and two (i.e. $r_{12} = 0$), is equivalent to cluster variables one and two being conditionally independent given all other variables, see Proposition 1 in Wilms & Bien (2022). In turn, all members of cluster one are then conditionally independent of all members in cluster two.

2.2 Clusterpath Estimator

To estimate a possibly sparse precision matrix with a block structure corresponding to clustered variables, we use a convex penalization method of the form

$$\hat{\Theta} = \underset{\Theta}{\operatorname{argmin}} L(\Theta) \quad \text{s.t. } \Theta = \Theta^\top, \Theta \succ 0, \quad (2)$$

$$L(\Theta) = -\log |\Theta| + \operatorname{tr} \mathbf{S} \Theta + \mathcal{P}(\Theta), \quad (3)$$

where $\log |\cdot|$ denotes the logarithm of the determinant, $\operatorname{tr}(\cdot)$ is the trace, $\cdot \succ 0$ denotes a positive definite matrix, and $\mathcal{P}(\Theta)$ is the penalty term. Throughout the paper we take

$$\mathcal{P}(\Theta) = \lambda_c \sum_{j=1}^p \sum_{j'=1}^{j-1} w_{jj'} d_{jj'}(\Theta) + \lambda_s \sum_{\substack{j,j' \\ j' \neq j}} z_{jj'} |\theta_{jj'}|, \quad (4)$$

with

$$d_{jj'}(\Theta) = \sqrt{(\theta_{jj} - \theta_{j'j'})^2 + \sum_{\substack{m=1 \\ m \notin \{j, j'\}}}^P (\theta_{jm} - \theta_{j'm})^2}.$$

The first part in (4) represents the aggregation penalty penalizing differences $d_{jj'}(\Theta)$ between columns θ_j and $\theta_{j'}$ of the precision matrix Θ . Its role is thus to encourage estimation of a precision matrix with a G -block structure; or, put alternatively, a GGM with clustered variables. The second part represents the sparsity penalty on the unique off-diagonal elements of the precision matrix. Its role is to encourage estimation of precision matrices with zero off-diagonal entries; or, put alternatively, GGMs with edge-sparsity. The tuning parameters λ_c and λ_s control the degree of aggregation and sparsity respectively. The objective function is convex since each of the terms in (3) is convex in Θ and Θ lies in the convex cone of positive definite matrices.

Focusing on the aggregation penalty, the weights $w_{jj'}$, for every unique pair $1 \leq j, j' \leq p$, determine the fundamental attraction between two variables and are specified beforehand based on, for example, domain knowledge or the pairwise distances based on the sample precision matrix \mathbf{S}^{-1} . We further discuss the choice of weights in Section 2.3. If $d_{jj'}(\Theta) = 0$ and λ_c is positive, then two vectors θ_j and $\theta_{j'}$ have exactly the same elements while ignoring $\theta_{jj'}$ and $\theta_{j'j}$. The distance $d_{jj'}(\Theta)$ disregards the difference between $\theta_{jj'}$ and $\theta_{j'j}$ as these are identical due to the symmetry of Θ . The penalty term $d_{jj'}(\Theta)$ can thus be interpreted as a group lasso penalty (Yuan & Lin, 2006), where each pair of variables $j < j'$ forms a group whose elements are the differences between the corresponding entries in the columns θ_j and $\theta_{j'}$ of the precision matrix. If all respective differences are put to zero, the estimated entries are identical, which in turn effectively blocks columns j and j' in the precision matrix, or equivalently clusters nodes j and j' in the GGM. As λ_c increases (and λ_s is fixed), estimated GGMs are obtained in which more and more variables are clustered, thereby resulting in a *clusterpath* from the p original nodes (no clustering) until one clustered node (full clustering) in the GGM.

The notion of applying a distance-based penalty to pairs of objects stems from convex clustering (Pelckmans et al., 2005; Hocking et al., 2011; Lindsten et al., 2011). In convex clustering, a copy of the data matrix is estimated while penalizing the distances between rows, thereby facilitating the clustering of observations. Building upon this framework, convex biclustering also clusters variables by adding a penalty on the distances between columns (Chi et al., 2017). Leveraging this notion for clustering variables via the precision matrix offers insights into the underlying conditional dependencies within the data.

2.3 Cyclic Block Coordinate Descent Algorithm

We develop a cyclic block coordinate descent algorithm, tailored to minimizing objective function (2).¹ We opt for cyclic block coordinate descent since the blocks are naturally formed by the clusters $k = 1, \dots, K$. By exploiting this block structure in Θ , our algorithm builds on efficient expressions of the objective function, which allows us to minimize optimization problem (2) in a computationally efficient way.

For ease of exposition, first suppose that the block structure of the precision matrix in equation (1) is known; hence the number of clusters K and the cluster membership is known. To permit cyclic block updates, one for each block/cluster k , the objective $L(\Theta)$ needs to be separated into those parts that depend on cluster k and those that do not. To this end, we re-write the precision matrix as

$$\Theta = \begin{bmatrix} \Theta_{00} & \Theta_{0k} \\ \Theta_{0k}^\top & \Theta_{kk} \end{bmatrix} = \begin{bmatrix} \Theta_{00} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{0} & \Theta_{0k} \\ \Theta_{0k}^\top & \Theta_{kk} \end{bmatrix} = \Theta_{-k} + \Theta_k,$$

where the first equality splits the precision matrix into four blocks: Θ_{00} contains all elements not pertaining to cluster k , Θ_{0k} contains all cross-cluster conditional covariances involving cluster k , and Θ_{kk} contains all

¹One could opt for alternative algorithms such as the alternating directions method of multipliers (ADMM), provided it is tailored towards solving problem (2); as for instance Wilms & Bien (2022) do for their method. However, while all the subproblems of the ADMM by Wilms & Bien (2022) are solvable in closed-form, this would not be the case for the subproblem corresponding to our proposed aggregation penalty. This reason, in combination with the natural link between the block structure in Θ and the block structure of the cyclic block coordinate descent algorithm further explains our choice for the latter.

within-cluster k conditional variances and covariances. Without loss of generality, the assumption can be made that the variables are arranged to consistently position cluster k as the last cluster. Finally, we denote all parameters in Θ belonging to cluster k as Θ_k , thereby making use of single subscript notation. Similarly, Θ_{-k} collects all parameters in Θ not belonging to cluster k . The objective function can then be partitioned into four distinct components, as given by

$$L(\Theta_k) = L_{\det}(\Theta_k) + L_{\text{cov}}(\Theta_k) + L_{\text{clust}}(\Theta_k) + L_{\text{sparse}}(\Theta_k) + C, \quad (5)$$

where L_{\det} , L_{cov} , L_{clust} , and L_{sparse} represent the log-determinant, trace, and penalty parts (cluster- and sparsity-based) of the objective function that depend on cluster k , and finally C is a constant collecting parts independent of cluster k . Details of these expressions are in Appendix A. The cyclic block coordinate descent algorithm then simply cycles through all blocks $k = 1, \dots, K$, thereby optimizing $L(\Theta_k)$ in the k th cluster. When optimizing $L(\Theta_k)$, we do this computationally efficiently by building on the G -block format of the precision matrix which re-parametrizes Θ in terms of \mathbf{A} and \mathbf{R} , see equation (1). The re-parametrization permits more efficient updates when optimizing $L(\Theta_k)$; in particular more efficient expressions of $L(\Theta_k)$, and of the gradient and Hessian of (5), as detailed in Appendix A.

In practice, the block structure of the precision matrix is not known beforehand; our penalization problem (2) jointly identifies the block structure and estimates the precision matrix Θ . The identification of the block structure, meaning the estimation of the number of clusters K and cluster composition, is done as follows. Every variable is initialized in its own cluster (i.e. $\hat{K} = p$). For a fixed value of the aggregation penalty λ_c and initial estimate at $\hat{\Theta}$, the clusters that are eligible for fusion are determined by computing the distances $d_{jj'}(\hat{\Theta})$ for every variable pair j, j' . If this distance reduces to zero, the two corresponding variables j and j' are eligible for fusion. The number of clusters \hat{K} and cluster composition are then updated accordingly. Given this updated block structure, objective function (5) is optimized for every block/cluster $k = 1, \dots, \hat{K}$.

In sum, the proposed cyclic block coordinate descent algorithm involves two key steps. For each cluster k , it first assesses whether there is an eligible fusion candidate. In the instances where no suitable candidate is identified, the algorithm proceeds to the second step, wherein the parameters associated with cluster k are updated using Newton's method. Another reason we choose a cyclic block coordinate descent algorithm lies with the use of Newton's method since the computation of a Newton descent direction for all parameters in the precision matrix simultaneously would result in a computationally intractable algorithm even for moderately sized data sets.

In Section 2.3.1, we provide details on the two key steps of the optimization algorithm. We further discuss two important components related to the penalty terms, namely the weight matrix (Section 2.3.2) and the tuning parameter (Section 2.3.3). Finally, in Section 2.3.4, we discuss a refitting step to reduce bias in the CGGM estimate.

2.3.1 Outline of the Cyclic Block Coordinate Descent Algorithm

We outline the main two steps of the cyclic block coordinate descent algorithm minimizing the objective function in (5) iteratively for cluster $1 \leq k \leq K$ for fixed values of the tuning parameters λ_c and λ_s . The pseudo-code for the algorithm is presented and discussed in Appendix B.1. The computational complexity of the algorithm is $\mathcal{O}(K^4)$, which is particularly efficient when the number of clusters K is small relative to the number of variables p . Moreover, the cyclic updates facilitate the preservation of the positive definiteness of Θ as long as each update satisfies two straightforward inequalities, detailed in Appendix B.1.

Cluster fusions. The first step is to verify whether cluster k is close enough to another cluster to warrant their fusion for estimate $\hat{\Theta}$ (in a given iteration of the algorithm). Let \mathcal{C}_k denote the set of variables belonging to cluster k , analogously \mathcal{C}_ℓ for cluster ℓ . Then, cluster k is fused with cluster ℓ if $d_{\mathcal{C}_k \mathcal{C}_\ell}(\hat{\Theta})$, which measures the distance between blocks $\hat{\Theta}_k$ and $\hat{\Theta}_\ell$, is smaller than some user-defined threshold ε_f representing the minimum required similarity for fusion. In case of multiple candidates for which this applies, cluster ℓ is chosen as $\ell = \text{argmin}_{\ell'} d_{\mathcal{C}_k \mathcal{C}_{\ell'}}(\hat{\Theta})$. If a fusion is performed, it advances to the next cluster. If no eligible fusion candidate is found in the first step, we proceed to the second step where we update the parameter estimates for cluster k .

Parameter estimation. The second step consists of updating the parameters estimates for cluster k using Newton's method. We opt for a Newton descent direction as it offers efficient convergence properties.

Let $\nabla L(\hat{\Theta}_k)$ denote the gradient and $\nabla^2 L(\hat{\Theta}_k)$ the Hessian of $L(\hat{\Theta}_k)$. The descent direction is then computed as $\delta_k = -\nabla^2 L(\hat{\Theta}_k)^{-1} \nabla L(\hat{\Theta}_k)$, see Appendix B.2 for the derivations.

Finally, we determine the optimal step size s^* given the descent direction using an inexact line search. The reason for augmenting the Newton descent direction with a line search is twofold. First, the minimization of $L(\Theta_k)$ is constrained by the restriction that Θ should be positive definite. Hence, a step size should be chosen that ensures the update to adhere this restriction. In Appendix B.1, we derive the conditions under which the optimal step size can be computed that adheres to the positive definiteness restriction. A second reason to opt for a line search is the fact that the Hessian may not be well-behaved due to the presence of the ℓ_2 -norm in the clusterpath penalty. If, for the current value $\hat{\Theta}$, at least one of the distances $d_{C_k C_\ell}(\hat{\Theta})$ is close to zero, $L(\hat{\Theta}_k)$ is not locally smooth. As a result, the Hessian is not well-behaved and the favorable convergence properties of Newton's method do not hold. A line search for the optimal step size guarantees that the update does not increase the objective function.

2.3.2 Weight Matrix

An important element of the clusterpath estimator is the weight matrix \mathbf{W} that contains information about the preferences of clustering variables j and j' through the weight $w_{jj'}$. A large value for $w_{jj'}$ incentivizes clustering of the corresponding variables, but it does not guarantee this. In the convex clustering literature, weights are typically based on the squared distances between data points, with performance improving when using k -nearest neighbors to assign zero weights to nonneighbor pairs (e.g., Chen et al., 2015; Chi & Lange, 2015). Following these standard practices, we scale the squared distances by the mean squared distance and take

$$w_{jj'} = w_{j'j} = \begin{cases} \exp\left(-\phi \frac{d_{jj'}^2(\mathbf{S}^{-1})}{\sum_{(m,m') \in \mathcal{E}} d_{mm'}^2(\mathbf{S}^{-1})/|\mathcal{E}|}\right) & \text{if } (j, j') \in \mathcal{E}, \\ 0 & \text{otherwise,} \end{cases} \quad (6)$$

where ϕ is a tuning parameter, \mathcal{E} is the set of variable pairs (j, j') that should be assigned a nonzero weight, and $|\mathcal{E}|$ denotes the number of elements in \mathcal{E} . Note that the choice of $w_{jj'}$ in (6) is only available if \mathbf{S} is invertible. If \mathbf{S} is not invertible, one can use a regularized version instead. Throughout the paper, we use $(\mathbf{S} + \mathbf{I})^{-1}$ in such cases, but the user may resort to other regularized alternatives such as the graphical lasso. For the hyperparameter ϕ , it can be beneficial to choose candidate values such that the nonzero weights are distributed broadly over the interval $(0, 1]$ without large peaks at specific values; we provide practical guidance in the applications discussed in Section 5.

A potential issue with relying solely on k -nearest neighbors for the construction of \mathcal{E} is the occurrence of disconnected subgroups. To alleviate this issue, we add variable pairs to \mathcal{E} using the procedure described in Touw et al. (2022) until all disconnected subgroups are eliminated. This approach effectively integrates the k -nearest neighbors structure with a minimum spanning tree to obtain a weight matrix corresponding to a connected graph.

2.3.3 Tuning Parameters

The tuning parameters λ_c and λ_s are other key ingredients of CGGM as they control the degree of clustering and sparsity respectively.

To set a sequence for λ_c (for given values of λ_s and \mathbf{W}), we implement an automated procedure that ensures a continuum of solutions for Θ with smooth transition from minimal (where $\hat{K} = p$) to maximal regularization (where $\hat{K} = 1$). Details are provided in Appendix B.3. Furthermore, note that for each subsequent optimization problem with a larger value for λ_c , we make use of warm starts provided by the solution to the previous problem, see Appendix B.1. This warm start includes the clustering information for that solution.

For the sparsity parameter λ_s , we use a grid of ten values with each step being twice as large as the previous step. The largest value in the grid is based on the smallest value of λ_s that puts all off-diagonal elements in Θ to zero for the graphical lasso (i.e. for $\lambda_c = 0$). The other values in the grid are set as fractions (from zero to one) of this maximum value.

Finally, to select the tuning parameters, we adopt a cross-validation procedure. That is, we find the combination of tuning parameters that minimizes the cross-validated likelihood-based score given by

$$\frac{1}{G} \sum_{g=1}^G -\log |\hat{\Theta}_{-\mathcal{F}_g}| + \text{tr} \mathbf{S}_{\mathcal{F}_g} \hat{\Theta}_{-\mathcal{F}_g}, \quad (7)$$

where $\hat{\Theta}_{-\mathcal{F}_g}$ represents the precision matrix estimated on the sample excluding the observations in the g^{th} fold, and $\mathbf{S}_{\mathcal{F}_g}$ denotes the sample covariance matrix computed solely on the samples within the g^{th} fold.

2.3.4 Refitted CGGM

While minimization of the objective function in (2) yields a possibly sparse and/or clustered $\hat{\Theta}$, the CGGM estimate may be biased due to the shrinkage of the distances between different clusters and the shrinkage of the entries in the precision matrix to zero. Following Wilms & Bien (2022), we therefore also consider a refitted version of CGGM. First, CGGM is used to obtain a clustering and sparsity pattern in $\hat{\Theta}$. We then re-estimate the precision matrix Θ constrained by the obtained clustering and sparsity by maximizing the likelihood subject to the clustering and sparsity constraint. By re-estimating the precision matrix, we aim to capitalize on the clustering- and sparsity-inducing capabilities of CGGM while refining parameter estimates for improved accuracy.

3 Simulations

To investigate the behavior of the proposed method CGGM, we perform an extensive simulation study. We discuss the data generating processes in Section 3.1, the benchmark methods and evaluation criteria in Section 3.2, and the results in Section 3.3.

3.1 Simulation Designs

For most of the simulations, we follow the designs of Wilms & Bien (2022) together with additional variations. In all settings, we sample data from a multivariate normal distribution with mean zero and covariance $\Sigma = \Theta^{-1}$, and we repeat the process 100 times.

Baseline simulation designs. We simulate $n = 120$ observations on $p = 15$ variables using four different structures for the precision matrix Θ (see the top row of Figure 2): (i) *random*: the clusters are of equal size with $p_1 = p_2 = p_3 = 5$, and one pair of clusters is selected at random to be connected via an edge; (ii) *chain*: the clusters are of equal size with $p_1 = p_2 = p_3 = 5$, and adjacent clusters are connected via an edge; (iii) *unbalanced*: same as the chain design, but the clusters are of unequal size with $p_1 = 3$, $p_2 = 5$, and $p_3 = 7$; (iv) *unstructured*: each variable forms its own cluster, and edges between variables are drawn with probability $\pi = 0.1$. The first three designs thus employ a block structure with $K = 3$ clusters, while the fourth does not exhibit any variable aggregation structure. In all four designs, the diagonal elements of Θ are set to 1, the elements within a cluster of variables to 0.5, and the non-zero elements between clusters to 0.25.

Increasing the number of variables. We focus on the chain design as in Wilms & Bien (2022) and vary the number of variables $p \in \{15, 30, 60, 120\}$. We set the corresponding number of observations $n \in \{120, 240, 480, 960\}$ such that the ratio n/p is constant. The number of clusters is kept fixed at $K = 3$.

Increasing the number of clusters. We again focus on the chain design and vary the number of clusters $K \in \{3, 5, 6, 10\}$ while keeping the number of observations and variables fixed at $n = 240$ and $p = 30$, respectively, as in Wilms & Bien (2022).

Approximate block structure. We take the four baseline designs and modify the structure in Θ by uniformly drawing elements within a cluster of variables from the interval $[0.4, 0.6]$ and non-zero elements between clusters from the interval $[0.2, 0.3]$. The assumptions of CGGM are not (fully) met in the resulting designs, as the clusters do not correspond to blocks of equal values in Θ but blocks of approximately similar values.

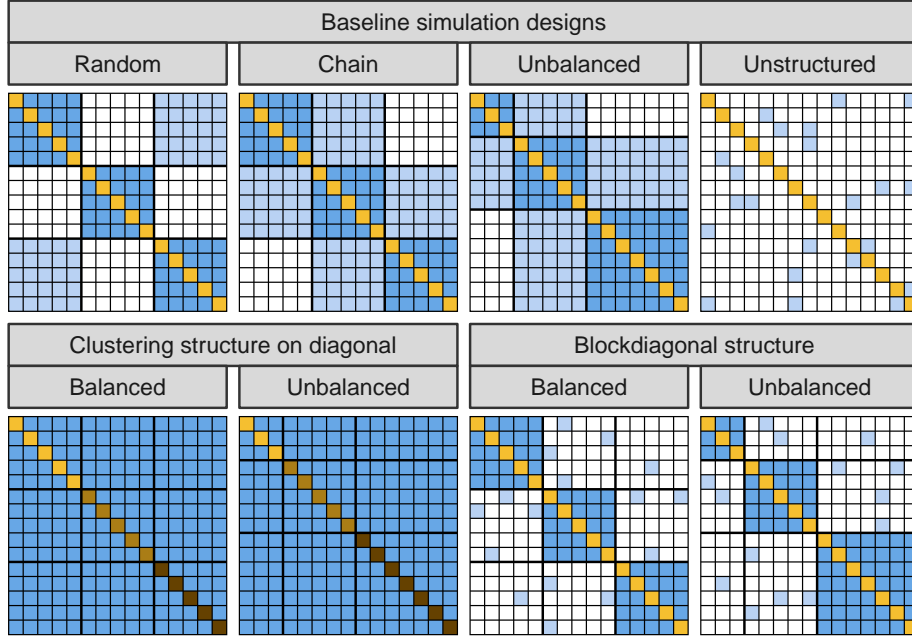


Figure 2: Precision matrices Θ in the baseline simulation designs (top), the designs with clustering structure on the diagonal and with blockdiagonal structure, respectively, using balanced and unbalanced clusters sizes (bottom). The color shade indicates the magnitude of the elements. Diagonal elements are on a separate color scale than the off-diagonal ones to highlight their differing roles in CGGM in comparison to the benchmark methods.

Clustering structure on diagonal. Using the same dimensions and number of clusters as in the baseline designs, all off-diagonal elements in Θ are set to 0.5. The diagonal elements are set to 1 for the variables in the first cluster, to 2 for the variables in the second cluster, and to 3 for the variables in the third cluster. We investigate both a *balanced* design with $p_1 = p_2 = p_3 = 5$, and an *unbalanced* design with $p_1 = 3$, $p_2 = 5$, and $p_3 = 7$ (see left two panels in the bottom row of Figure 2).

Blockdiagonal structure. In these designs, the precision matrix Θ does not exhibit a variable clustering structure, but a blockdiagonal structure inspired by the simulation design of Pircalabelu & Claeskens (2020). We use the same dimensions as in the baseline design and $K = 3$ blocks on the diagonal. Outside of the diagonal blocks, edges between variables are drawn with probability $\pi = 0.1$. The diagonal values of Θ are set to 1, the off-diagonal elements in the diagonal blocks are set to 0.5, and non-zero elements outside the diagonal blocks are set to 0.25. If such a randomly drawn precision matrix is not positive semi-definite, we repeat this process until a positive semi-definite precision matrix is obtained. We consider both a *balanced* design with $p_1 = p_2 = p_3 = 5$, and an *unbalanced* design with $p_1 = 3$, $p_2 = 5$, and $p_3 = 7$ (see right two panels in bottom row of Figure 2).

3.2 Methods and Evaluation Criteria

We apply CGGM with and without the parameter re-estimation step described in Section 2.3.4, which we refer to as *CGGM-raw* and *CGGM-refit*. We thereby set the convergence tolerance $\varepsilon_c = 10^{-7}$ and the maximum number of iterations to $t_{\max} = 5000$. For comparison, we include the tree-aggregated graphical lasso (*TAGL*) of Wilms & Bien (2022), the community-based group graphical lasso (*ComGGL*) of Pircalabelu & Claeskens (2020), the graphical lasso (*GL*) (Friedman et al., 2008), and the inverse of the sample covariance matrix (\mathbf{S}^{-1}).

TAGL performs node aggregation based on side-information in the form of a tree-based variable hierarchy, which needs to be specified a priori. We generate an *ideal* and a *realistic* tree hierarchy as described in Wilms & Bien (2022) (see Figure 5 in their paper for an illustration). As both trees contain the true variable

clustering, we also generate a *misspecified* tree in the same manner as the realistic tree, except that with probability 0.1, each variable is incorrectly assigned to the subsequent cluster (or the first cluster if the variable in fact belongs to the last cluster).

ComGGL does not perform node aggregation but community detection in the form of a blockdiagonal structure. Since ComGGL merely encourages block-diagonality in the precision matrix, the entries of the precision matrix estimate corresponding to features belonging to the same community may still vary. Hence, ComGGL does not induce block-structured precision matrix estimates but serves mostly as a relevant benchmark for the simulation designs with approximate block structure and blockdiagonal structure.

Concerning tuning parameter selection, we apply 3-fold cross-validation. For the weight matrix in CGGM, we use candidate values $k \in \{1, 3, 5\}$ for the number of neighbors and we set $\phi = 1$ to keep the computational burden low. We verified that for this choice of ϕ , the weights showed sufficient variation over the interval $(0, 1]$ to stimulate variable clustering. Moreover, we select candidate value for the regularization parameters λ_c and λ_s as described in Section 2.3.3. For TAGL, we first perform a binary search for the smallest value of the aggregation parameter that aggregates all variables into one cluster (for each of the tree hierarchies) while keeping the sparsity parameter at 0, and we determine the smallest value of the sparsity parameter that removes all edges while keeping the aggregation parameter at 0 (corresponding to the graphical lasso). Ten candidate values for the aggregation and sparsity parameters are then obtained as fractions of those aforementioned values, where the fractions vary from 0 to 1, with each step being twice as large as the previous step. For ComGGL, we similarly set ten candidate values for the grouping and sparsity parameters as fractions of maximum values. While we fix the maximum value of the grouping parameter at 1, we again set the maximum value of the sparsity parameter as the smallest value that removes all edges in the graphical lasso. As in Pircalabelu & Claeskens (2020), we set the balancing parameter to 1. For the graphical lasso, we choose ten candidate values for the sparsity parameter in the same manner.

We evaluate the methods in terms of their estimation accuracy, aggregation, and sparsity recognition performance. Regarding estimation accuracy, we compute the Frobenius norm $\|\Theta - \hat{\Theta}\|_F$. For aggregation performance, we report the estimated number of clusters \hat{K} as well as the adjusted Rand index (ARI) (Hubert & Arabie, 1985) of the obtained clustering of variables. Note that \hat{K} directly follows from the obtained clustering for the optimal values of the tuning parameters from cross-validation. For sparsity recognition, we report the false positive and false negative rates (FPR and FNR, respectively), where the FPR reports the fraction of true zero elements of the precision matrix that are estimated as nonzero and the FNR gives the fraction of true nonzero elements of the precision matrix that are estimated as zero.

3.3 Results

In Figures 3 and 4, we report the evaluation criteria for different simulation designs. As CGGM and TAGL outperform ComGGL, the graphical lasso (GL), and the inverse of the sample covariance matrix (\mathbf{S}^{-1}) in almost all cases, we focus on the former two in the following discussion of the results.

Baseline simulation designs. In terms of estimation accuracy and aggregation, CGGM-raw and CGGM-refit perform similar to TAGL-ideal, outperform TAGL-realistic in some designs, and outperform and TAGL-misspecified in all designs (see Figure 3). Regarding sparsity recognition, one variant of CGGM generally outperforms all variants of TAGL, depending on the setting. The refitting step of CGGM is advantageous in the three block designs but disadvantageous in the unstructured design. For the latter, the refitting step often aggregates all variables into one cluster, likely due to the high degree of sparsity in the unstructured design. This pooling of estimates for the large number of zero elements reduces noise, potentially improving the out-of-sample cross-validation score in Equation (7), despite introducing some bias in the relatively small number of nonzero elements. Conversely, CGGM-raw allows for shrinkage between elements without aggregation, thus improving estimation performance for designs in which many off-diagonal elements share the same value without belonging to the same cluster. While TAGL-misspecified often selects the correct number of clusters, the ARI reveals that it struggles to accurately classify the variables within those clusters due to the misspecified tree. This pattern persists across most of the simulation designs. Note that across all simulations, ComGGL detects a single community. This should, however, not be interpreted as a block-structured precision matrix estimate with a single cluster in the sense of equation (1), as ComGGL allows entries of the precision matrix estimate within the same community to vary.

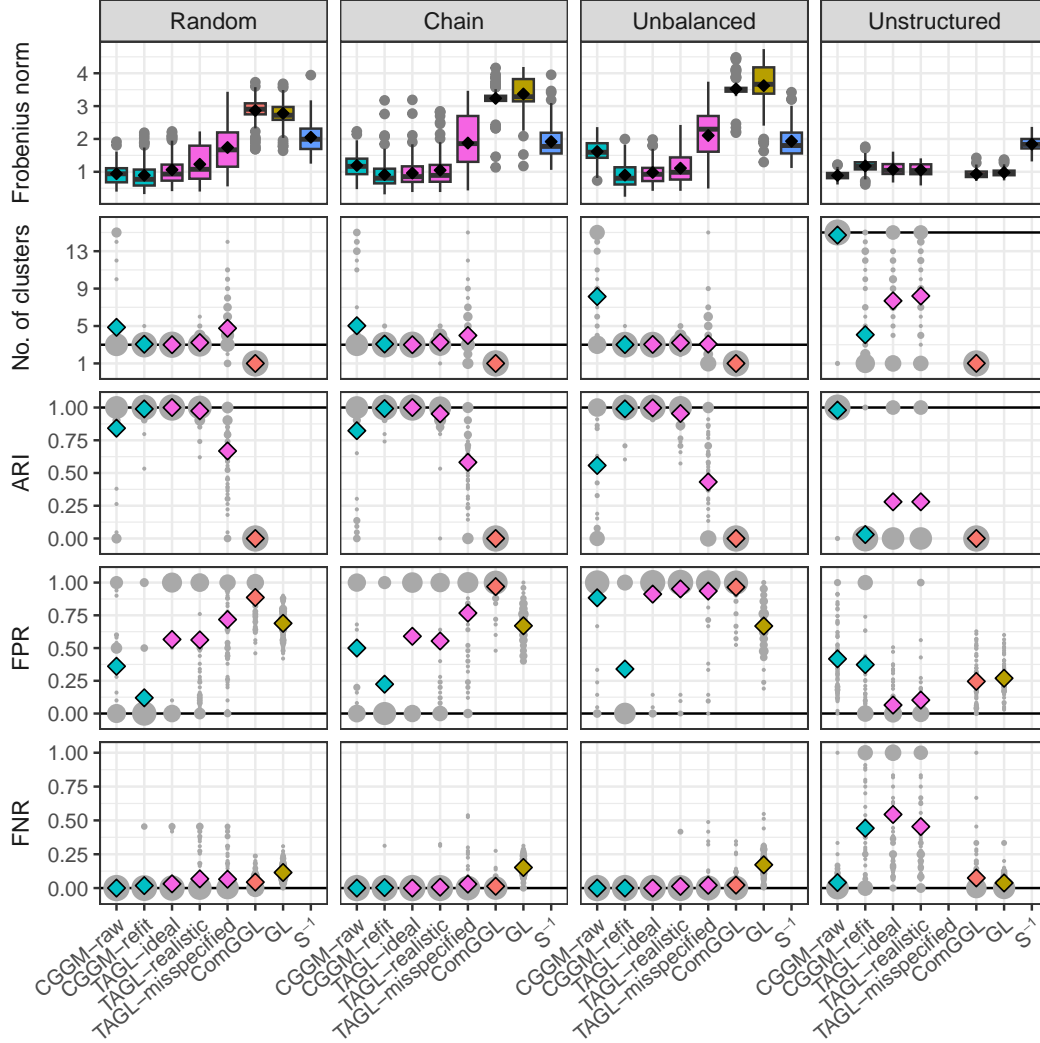


Figure 3: Results for the baseline simulation designs (columns). Top row: Boxplots of the Frobenius norm with black diamonds representing the average. Other rows: Diamonds displaying the average of the estimated number of clusters, ARI, FPR, and FNR. Reference lines are added for the true number of clusters, the ARI value of perfect clustering, and the FPR and FNR of perfect sparsity recognition, respectively. The size of the grey dots represents the frequency of different values across the replications. Aggregation performance is not applicable and omitted for GL and \mathbf{S}^{-1} , as is sparsity recognition performance for \mathbf{S}^{-1} . In the unstructured design, a misspecified tree for TAGL does not exist since any tree hierarchy contains the true clustering (each variable being its own cluster).

Increasing the number of variables or clusters. Given that CGGM-refit clearly outperforms CGGM-raw in the baseline chain design, we omit the results for the latter in the variations with increasing number of variables or clusters. The results remain relatively stable when increasing the number of variables or clusters and are similar to those in the baseline setting (see Figures 9 and 10 in Appendix C), likely due to maintaining a fixed ratio of n to p . CGGM-refit performs comparably to TAGL-ideal and TAGL-realistic in terms of estimation accuracy and aggregation, and clearly surpasses TAGL-misspecified. In terms of sparsity recognition, CGGM-refit and all variants of TAGL have a near-perfect FNR, but CGGM-refit has a lower FPR. This finding remains stable with an increasing number of variables (except for some variation in the FPR of TAGL-realistic and TAGL-misspecified). For an increasing number of clusters, the variants of TAGL exhibit a more prominent improvement in FPR than CGGM-refit, although the latter remains better.

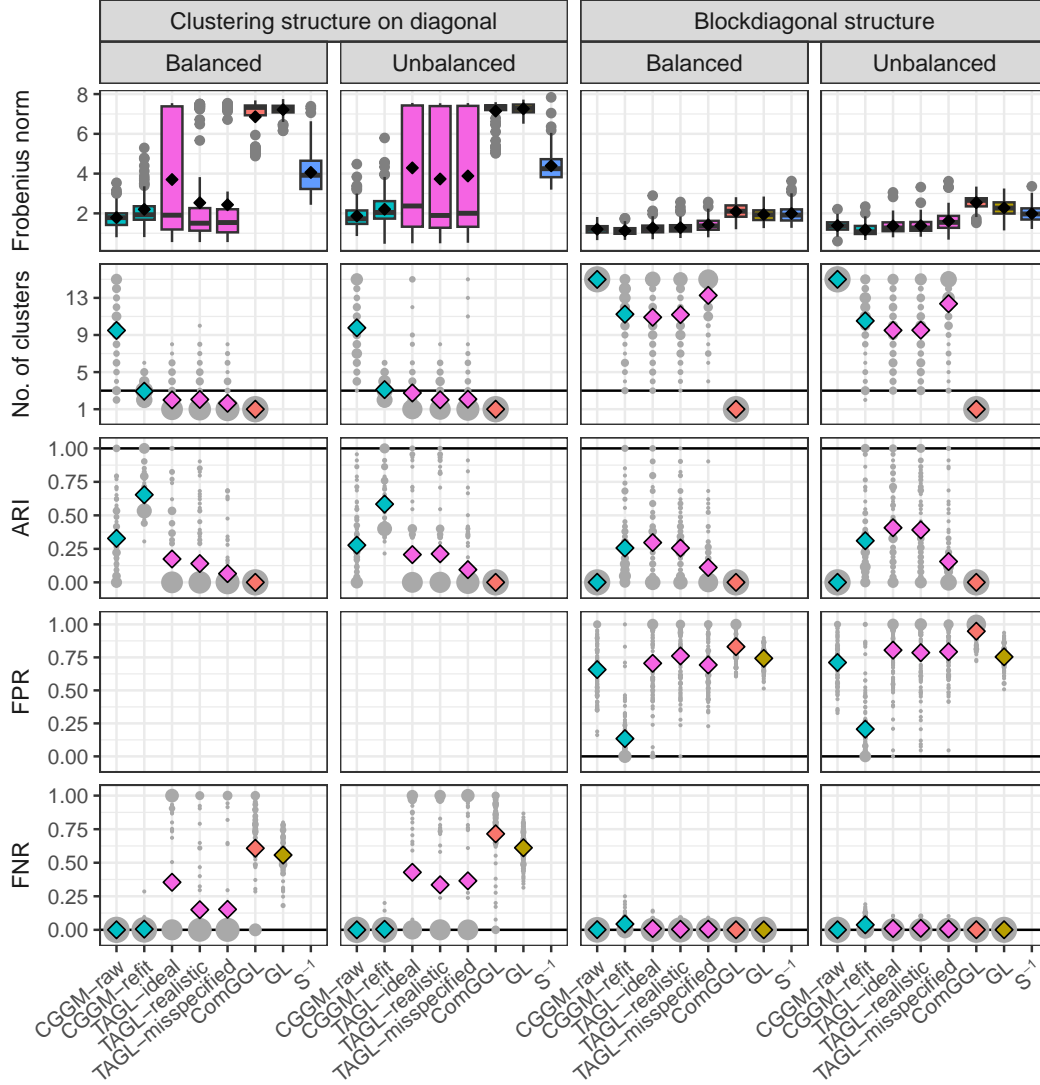


Figure 4: Results for the simulation designs with clustering structure on the diagonal (left) and with a blockdiagonal structure (right). See Figure 3 for explanatory notes. The FPR is not applicable in the design with clustering structure on the diagonal, since no elements of the true precision matrix are zero.

Approximate block structure. The results for estimation accuracy are highly similar to those of the baseline designs (see Figure 11 in Appendix C). In terms of aggregation performance, CGGM-raw struggles to find the block structure when it is no longer exact, but the refitting step overcomes this issue. TAGL-realistic also performs worse in terms of aggregation, though the effect is less pronounced than with CGGM-raw. Overall, aggregation performance of CGGM-refit and TAGL-ideal remains very similar to the baseline designs. It is important to note that, technically, the identified block structure is incorrect at the population level. However, the noise reduction in finite samples is still beneficial. Concerning sparsity recognition, the FNR of all variants of CGGM and TAGL stays near-perfect whereas the FPR generally increases. CGGM-refit thereby shows the smallest increase in FPR.

Clustering structure on diagonal. CGGM-refit clearly outperforms the alternative methods in terms of aggregation performance (see left two columns in Figure 4). All variants of TAGL tend to aggregate all variables into one cluster, as they exclude the elements on the diagonal from the aggregation penalty. As a result, the variability in the estimation error of TAGL is considerably larger compared to CGGM. Despite the better aggregation performance, the estimation error is slightly higher for CGGM-refit than for CGGM-raw.

A likely explanation of this phenomenon is that CGGM-refit no longer applies shrinkage to the difference between elements from different clusters. In this design, this behavior does not align with the structure of the underlying precision matrix, in which all off-diagonal elements share the same value. Consequently, by shrinking the off-diagonal elements from different clusters towards each other, CGGM-raw achieves superior estimation accuracy even without finding the true cluster structure. In addition, the two variants of CGGM yield a near-perfect and lower FNR than all three variants of TAGL.

Blockdiagonal structure. While CGGM-refit shows comparable estimation accuracy and aggregation performance to TAGL-ideal and TAGL-realistic, it demonstrates a sizable advantage over its alternatives in terms of sparsity recognition due to a much lower FPR (see right two columns in Figure 4). As in the designs with an approximate block structure, the identified block structure is technically incorrect at the population level. Nonetheless, identifying groups remains beneficial, considering the limited number of nonzero elements outside the diagonal blocks. Furthermore, it is noteworthy that CGGM-raw achieves competitive estimation accuracy with minimal aggregation.

Conclusions from the simulations. CGGM-refit exhibits excellent performance similar to TAGL-ideal while not requiring auxiliary information. Additionally, the simulations reveal that a misspecified tree that encodes the aggregation is detrimental to the performance of TAGL, positioning CGGM as a particularly strong alternative when accurate information on the aggregation structure is unavailable. CGGM consistently surpasses or at least keeps up with alternatives across the investigated simulation designs. In general, the refitting step is beneficial, but may lead to overaggregation and overly sparse estimates in unstructured settings. Correspondingly, in some scenarios, we find that there is a trade-off between CGGM-raw and CGGM-refit in terms of estimation accuracy, aggregation performance, and sparsity recognition.

Computation time. We compare the computation time of CGGM, TAGL, and ComGGL using the same data generating process as in the simulation design with increasing number of variables, which keeps the ratio n/p constant. For each method, we compute the total computation time over a grid of values for the aggregation parameter as used in the simulations, averaged over 10 replications. Other tuning parameters are fixed to values that were frequently found to be optimal (the fifth value in the grid for the sparsity parameter for all methods; $k = 5$ for the weight matrix in CGGM). Figure 12 in Appendix C displays the resulting computation times. ComGGL scales best as the number of variables increases. However, CGGM both has lower initial computation time and scales better than TAGL, thereby offsetting the additional computational burden for tuning additional hyperparameters.

4 Estimation of a Clustered Covariance Matrix

While we focused so far on estimating a clustered precision matrix to simplify the partial dependency structure among the variables, it can also be of interest to obtain a clustered covariance matrix. For instance, a block structure in the covariance matrix implies that the variables of a given block have equal loadings in latent factor models. In this context, a clustered covariance matrix may reduce uncertainty in estimating the latent factors.

Existing paradigms for regularized estimation either induce an appropriate simplicity structure on the precision matrix or the covariance matrix, and the induced simplicity structure is, in general, not maintained when taking the inverse of the object of interest (e.g., Yao & Allen, 2019, Pircalabelu & Claeskens, 2020, and Wilms & Bien, 2022). The proposed CGGM estimator connects both paradigms since it retains the found block structure when taking the inverse of the obtained estimate (Section 2.1). If model (1) holds for the precision matrix Θ , it may therefore be reasonable to obtain a clustered estimate of the covariance matrix Σ by taking $\hat{\Sigma} = \hat{\Theta}^{-1}$, where $\hat{\Theta}$ denotes the CGGM estimate of the precision matrix. While the induced shared block structure in the precision and covariance matrix is a distinct feature of our proposal, adherence to an exact block structure as in model (1) may, however, be a strong assumption in practice. If this model only holds approximately with somewhat similar values within the blocks of Θ , we have seen in the simulations that estimating a clustered precision matrix with CGGM is nevertheless beneficial. But if an approximate block structure is present in the covariance matrix Σ , the structure in Θ is much more noisy with larger variability between the elements in the corresponding blocks, see Figure 5. In such settings, first estimating a clustered precision matrix with CGGM and then taking the inverse may not succeed in finding the correct structure.

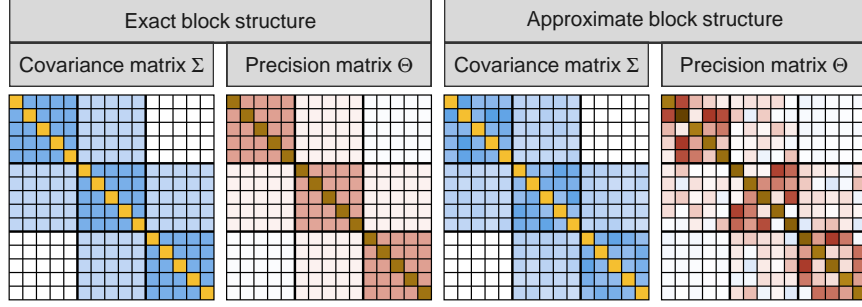


Figure 5: Illustration of how an exact block structure is retained between the covariance matrix Σ and the precision matrix Θ (left), whereas an approximate block structure in Σ corresponds to a much more noisy structure in Θ (right).

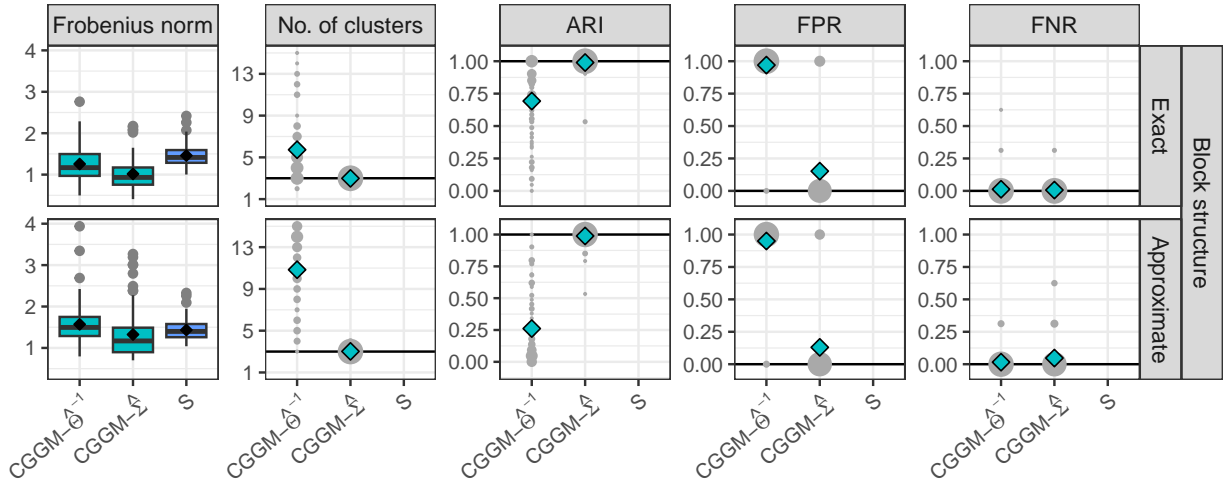


Figure 6: Simulation results for the estimation of a clustered covariance matrix, with different simulation designs in separate rows and different evaluation metrics in separate columns. Cf. Figure 3 for explanatory notes.

To estimate a clustered covariance matrix, one could attempt to adapt the clusterpath algorithm to be based on the objective function

$$-\log |\Sigma^{-1}| + \text{tr} \mathbf{S} \Sigma^{-1} + \lambda \mathcal{P}(\Sigma).$$

An algorithm optimizing this objective function must deal with an additional layer of complexity, as evaluation of Σ is required in the penalty part and evaluation of Σ^{-1} in the likelihood part of the objective function.

As an alternative, consider a random vector \mathbf{X} that follows a normal distribution with covariance matrix Σ . Then there exists an affine transformation matrix $\mathbf{A} = \mathbf{A}(\Sigma)$ so that $\mathbf{Y} = \mathbf{A}^\top \mathbf{X}$ follows a normal distribution with covariance matrix $\Theta = \Sigma^{-1}$. That is, Σ is the precision matrix for the random vector \mathbf{Y} , which implies that we can apply CGGM to minimize the objective function $L(\Sigma)$ from (2) with the following minor modification. As we do not observe realizations of \mathbf{Y} , we cannot compute the sample covariance matrix of those realizations. Instead, we take the inverse of the sample covariance matrix \mathbf{S}^{-1} of the observed realizations of \mathbf{X} as input for the CGGM algorithm. The corresponding optimization problem remains convex, but practitioners can benefit—in terms of estimation accuracy and interpretability—from tweaking the optimization problem towards directly solving for the covariance matrix instead of the precision matrix.

To demonstrate this numerically, we conduct simulations using the baseline chain design—which employs an exact block structure—and the approximate block structure chain design (see Section 3.1), but with

the block structure in the covariance matrix Σ rather than the precision matrix Θ . We apply CGGM for obtaining a clustered precision matrix followed by taking the inverse (denoted by CGGM- $\hat{\Theta}^{-1}$) and the modification for obtaining a clustered covariance matrix (denoted by CGGM- $\hat{\Sigma}$), as well as the sample covariance matrix \mathbf{S} . Both variants of CGGM include the parameter re-estimation step from Section 2.3.4. As the tuning parameter ϕ may have a different effect on clustering the covariance and precision matrix, respectively, we now also include candidate values $\phi \in \{1, 2, 3\}$ in the cross-validation scheme. The results are shown in Figure 6. Clearly, CGGM- $\hat{\Sigma}$ succeeds in finding the relevant aggregation and sparsity structure in most replications whereas CGGM- $\hat{\Theta}^{-1}$ struggles to do so, even for the setting with an exact block structure. While both variants of CGGM improve upon the sample covariance matrix \mathbf{S} in terms of the estimation error, CGGM- $\hat{\Sigma}$ exhibits the lowest error. Although a more thorough evaluation is beyond the scope of this paper, our findings indicate that applying CGGM- $\hat{\Sigma}$ may be preferable over CGGM- $\hat{\Theta}^{-1}$ when a clustered covariance matrix is of primary interest. We therefore recommend users to first decide on the object of interest, a block-structured covariance or precision matrix, and then use CGGM accordingly.

5 Applications

We demonstrate CGGM’s practical usefulness and versatility on 3 applications: a finance application using stock data (Section 5.1), an application with country-level well-being indicators (Section 5.2), and a survey-based behavioral science application (Section 5.3).

5.1 S&P 100 Stocks

We consider a financial data set containing daily realized ranges—a volatility measure—of $p = 101$ stocks of the companies constituting the S&P 100 on September 18th, 2023.² We study the conditional dependency structure of the stocks’ realized ranges over the period January 3rd, 2023, until December 29th, 2023 ($n = 250$), thereby comparing the performance of CGGM to that of TAGL applied to the precision matrix. While the former learns how to cluster the variables in an unsupervised and data-driven manner, the latter requires side-information in the form of a tree that encodes the similarity between the stocks to do so. To this end, we use the Global Industry Classification Standard (GICS). The tree then consists of the $p = 101$ stocks (leaves), 11 sectors as middle layer where each stock/company belongs to one industry sector, and one root node that aggregates all sectors.

Since the data are temporally dependent, we first—as a preprocessing step—fit the popular heterogeneous autoregressive (HAR) model of Corsi (2009) to the individual daily realized ranges to capture time dependencies, and then apply CGGM and TAGL to the standardized residuals to learn the conditional dependency structure among the stocks. See Appendix D.1 for details on data preprocessing and tuning parameter selection, including the selection of candidate values for ϕ based on the resulting weight distributions.

The cluster solution returned by both procedures differs considerably. CGGM groups the stocks into $K = 3$ clusters: the first contains BAC (issued by Bank of America Corp.), the second contains GOOG and GOOGL (both issued by Alphabet), and the last cluster contains the remaining stocks. CGGM thus puts the Bank of America and the technology company Alphabet in the spotlight as opposed to the rest of the market. TAGL, in contrast, results in $K = 9$ clusters where the industry sectors are assigned to the different clusters on an almost one-to-one basis. In Figure 7 (left panel), we present the sector distribution over the clusters for CGGM and TAGL. The unbalanced cluster solution of the former versus the more balanced one of the latter directly stands out. The role of BAC, GOOG, and GOOGL in our analysis is in line with the central, market-wide role of banks and technology companies as their influence on prices affects stocks across the whole market.

Given the considerable difference in clusters returned by both methods, a natural question is how both perform in capturing the conditional dependency structure. To this end, we conduct an out-of-sample exercise. An additional outer loop for 10-fold cross-validation is used to compute out-of-sample errors on each of the $G = 10$ test samples according to the likelihood-based score (7). These errors on the test data

²The S&P 100 contains 100 companies that may have one or more classes of stocks listed. During the time frame of our analysis, Alphabet was the sole company with more than one class, distinguished by the symbols GOOG and GOOGL, therefore resulting in $p = 101$ instead of $p = 100$ variables in our analysis.

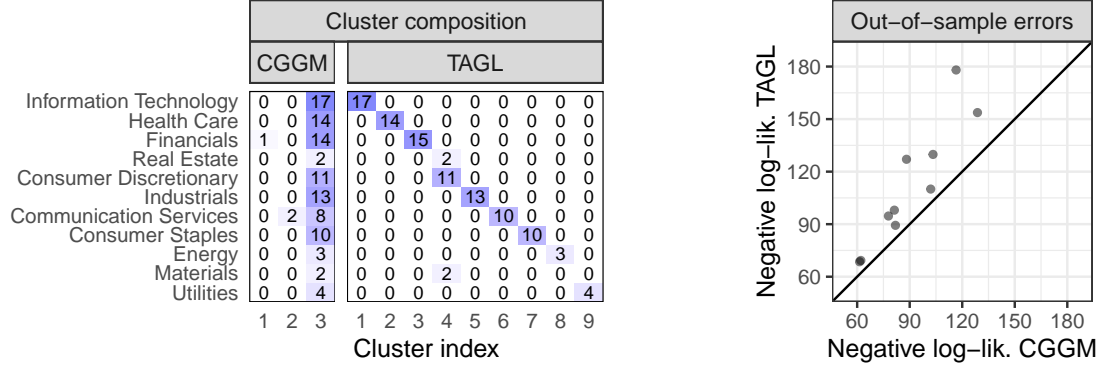


Figure 7: Finance application. Sector distribution over the clusters of CGGM and TAGL (left). Out-of-sample errors across the 10 replications (dots) for CGGM and TAGL (right; note that two of the dots are overplotted).

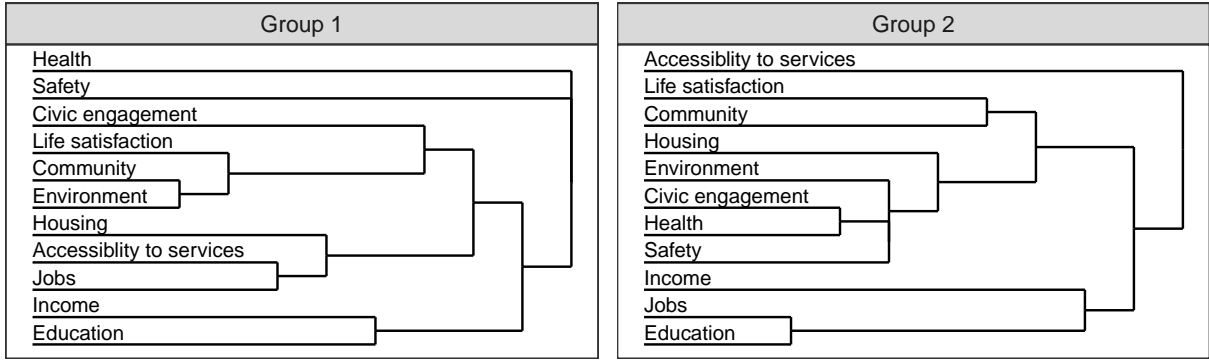


Figure 8: Well-being application. Clusterpath dendrograms obtained by CGGM on the two groups of countries.

are visualized in Figure 7 (right panel) with CGGM on the horizontal and TAGL on the vertical axis. For each test sample (dots), CGGM has a lower error thereby indicating that the more unbalanced clusters form a better description of the conditional dependency structure in this context.

5.2 OECD Well-Being Indicators

The second data set we analyze is one on OECD well-being indicators, collected in 2018 by Cavicchia et al. (2022) (see Appendix D.2). The data contain $p = 11$ variables related to well-being: education, jobs, income, safety, health, environment, civic engagement, accessibility to services, housing, community, and life satisfaction, on which two groups of countries with sample sizes $n_1 = 21$ and $n_2 = 14$ are given a score ranging from 0 to 10.

We highlight the capabilities of CGGM in estimating a cluster hierarchy on the standardized well-being data. Since our main goal is the retrieve the cluster hierarchy, we set $\lambda_s = 0$ and report the whole clusterpath solution obtained by applying CGGM to the precision matrix with different values of the tuning parameter λ_c for the two country groups. Due to the small group sample sizes, we do not use cross-validation to select the tuning parameters ϕ and k but use fixed values based on existing conventions, namely $\phi = 0.5$ and $k = 3$ (e.g., Chi & Lange, 2015; Wang et al., 2018). Since our primary interest is in the clusterpaths, the refitting step described in Section 2.3.4 is not required as it does not affect the obtained paths. Figure 8 visualizes the dendrograms as obtained from CGGM's complete clusterpath solution ranging from $p = 11$ clusters to one. If selection of a single solution along the clusterpath is requested, we advice practitioners to resort to domain expertise instead of data-driven methods when the sample size is that limited.

	Cluster 1	Cluster 2	Cluster 3	Cluster 4
Affiliative	{1, 5, 9, 13, 17, 21, 25, 29}			
Self-enhancing	{6, 30}	{2, 10, 14, 18, 22, 26}		
Aggressive			{3, 7, 11, 15, 19, 23, 27, 31}	
Self-defeating		{28}		{4, 8, 12, 16, 20, 24, 32}

Table 1: Humor styles questionnaire application. Allocation of the items for measuring the four humor styles (in rows) to the four clusters found by CGGM (in columns).

The analysis of the well-being indicators across the two groups of countries reveals distinct clustering patterns that highlight differences in their socioeconomic conditions. In the first group (e.g., Australia, France, Switzerland), the aggregation of civic engagement, life satisfaction, community, environment, housing, accessibility to services, and jobs appears to reflect the social and environmental dimensions of well-being. Conversely, in the second group (e.g., Chile, Hungary, Mexico), safety and health are combined with the variables representing social and environmental dimensions, possibly suggesting a greater reliance on the community for these services. This contrasts with more developed countries where such services may be perceived as more self-evident. Additionally, in the second group, income, jobs, and education are grouped together, reflecting their importance to socioeconomic progress. Finally, note that we use CGGM with the precision matrix as object of interest; its block structure directly transfers to the covariance matrix thereby making our results comparable to those of Cavicchia et al. (2022) who estimate the clustering structure on the covariance matrix.

5.3 Humor Styles Questionnaire

In behavioral science surveys, multiple rating-scale items are typically used to measure each latent construct of interest. For further analysis, mean scores across the respective items are commonly computed as measurements of each latent construct, which implies the assumption that the covariance matrix of the questionnaire items follows a block structure. The humor styles questionnaire (HSQ) (Martin et al., 2003) measures four latent constructs corresponding to different styles of humor: *affiliative*, *self-enhancing*, *aggressive*, and *self-defeating*, with each construct being measured by eight items ($p = 32$). If we can retrieve these four clusters of items in the covariance matrix, the assumption behind computing mean scores can be considered reasonable. Details on the HSQ data and tuning parameter selection in CGGM for clustering the covariance matrix are given in Appendix D.3.

CGGM returns four clusters of items that largely overlap with the grouping of Martin et al. (2003), as shown in Table 1. Looking at the wording of the questionnaire items, the clustering found by CGGM is intuitive. All items of the *affiliative* group, which CGGM places in Cluster 1, refer to humor in social settings. In addition, the two items of the *self-enhancing* group that complete Cluster 1—items 6 and 30—address being alone as opposed to being with people. On the other hand, the remaining six items of the *self-enhancing* group, which CGGM puts in Cluster 2, all refer to using humor for dealing with feeling depressed, upset, unhappy, sad, or having problems. Item 28 from the *self-defeating* group, which completes Cluster 2, also addresses having problems or feeling unhappy. In contrast, the other seven items of this group, which form Cluster 4 in the CGGM solution, refer to laughing or letting others laugh at oneself. Finally, all items from the *aggressive* group address teasing and offensive or inappropriate humor, and are placed in Cluster 3.

Regarding sparsity, Martin et al. (2003, Table 2) report significant positive correlations between most of the four humor styles, at least for male participants. CGGM on the given data, on the other hand, finds nonzero covariance only between Clusters 1 and 2 as well as Clusters 2 and 4. That is, through regularization,

the structure found by CGGM suggests latent variables that are more distinct than the factors found by Martin et al. (2003), which is desirable when characterizing different humor styles.

Although the above interpretation of the obtained clusters is subjective, our findings suggest that the survey design of the HSQ may benefit from further finetuning. Moreover, if subsequent analysis is based on mean scores, a slightly different allocation of the items to the latent constructs—the humor styles—may be more in line with the implicit assumption of a block structure in the covariance matrix.

6 Conclusion

We introduce a novel method to estimate Gaussian graphical models (GGMs) subject to node-clustering in addition to edge-sparsity. Our method, which we call CGGM, uses a clusterpath penalty to produce a hierarchical clustering of variables (nodes in the graph) without relying on pre-existing notions of the cluster composition. Our software package **CGGMR** for the statistical computing environment R (R Core Team, 2024) implements the proposed method and is available from <https://github.com/djwouw/CGGMR>.

In a comprehensive simulation study covering a wide range of graph structures, we compare CGGM to similar benchmarks such as TAGL, ComGGL, and the graphical lasso. CGGM oftentimes surpasses the benchmarks in terms of estimation accuracy as well as clustering and sparsity recognition performance, though the latter require a refitting step particularly in noisy simulation designs. Through a diverse set of applications, we also demonstrate the versatility of CGGM in (i) learning cluster hierarchies in a fully data-driven way as compared to other benchmarks such as TAGL that require additional side-information for this task, (ii) delivering accurate cluster hierarchies that are transferable between the precision and covariance matrix, and (iii) directly estimating clustered covariance matrices if these are the primary object of interest.

Concerning future work, CGGM may be extended to partial correlation matrices instead of precision matrices. In contrast to the graphical lasso (GL), Carter et al. (2024) apply the sparsity penalty to the elements of the partial correlation matrix and call their method the partial correlation GL. As a result, the sparsity structure estimated by this method is invariant to scalar multiplication of the data. CGGM can also be modified to accommodate clustering based on the partial correlation matrix, but this comes at the cost of the convexity of the objective function as with the partial correlation GL.

Computational Details

All computations were performed with R (R Core Team, 2024). *Replication files will be made publicly available upon acceptance and a link will be included here.*

Acknowledgments

Andreas Alfons is supported by a grant from the Dutch Research Council (NWO), research program Vidi, grant number VI.Vidi.195.141. Ines Wilms is supported by the same funding agency under grant number VI.Vidi.211.032.

References

- Ambroise, C., Chiquet, J., & Matias, C. (2009). Inferring sparse Gaussian graphical models with latent structure. *Electronic Journal of Statistics*, 3, 205–238.
- Archakov, I. & Hansen, P. R. (2024). A canonical representation of block matrices with applications to covariance and correlation matrices. *The Review of Economics and Statistics*, 106(4), 1099–1113.
- Brownlees, C., Guðmundsson, G. S., & Lugosi, G. (2022). Community detection in partial correlation network models. *Journal of Business & Economic Statistics*, 40(1), 216–226.
- Bunea, F., Giraud, C., Luo, X., Royer, M., & Verzelen, N. (2020). Model assisted variable clustering: minimax-optimal recovery and algorithms. *The Annals of Statistics*, 48(1), 111–137.

- Cai, T., Liu, W., & Luo, X. (2011). A constrained ℓ_1 minimization approach to sparse precision matrix estimation. *Journal of the American Statistical Association*, 106(494), 594–607.
- Carter, J. S., Rossell, D., & Smith, J. Q. (2024). Partial correlation graphical lasso. *Scandinavian Journal of Statistics*, 51(1), 32–63.
- Cavicchia, C., Vichi, M., & Zaccaria, G. (2022). Gaussian mixture model with an extended ultrametric covariance structure. *Advances in Data Analysis and Classification*, 16(2), 399–427.
- Chakraborty, S. & Xu, J. (2023). Biconvex clustering. *Journal of Computational and Graphical Statistics*, 32(4), 1524–1536.
- Chen, G. K., Chi, E. C., Ranola, J. M. O., & Lange, K. (2015). Convex clustering: An attractive alternative to hierarchical clustering. *PLoS Computational Biology*, 11(5), 1–31.
- Chi, E. C., Allen, G. I., & Baraniuk, R. G. (2017). Convex biclustering. *Biometrics*, 73(1), 10–19.
- Chi, E. C. & Lange, K. L. (2015). Splitting methods for convex clustering. *Journal of Computational and Graphical Statistics*, 24(4), 994–1013.
- Colombi, A., Argiento, R., Paci, L., & Pini, A. (2024). Learning block structured graphs in Gaussian graphical models. *Journal of Computational and Graphical Statistics*, 33(1), 152–165.
- Corsi, F. (2009). A simple approximate long-memory model of realized volatility. *Journal of Financial Econometrics*, 7(2), 174–196.
- Eisenach, C., Bunea, F., Ning, Y., & Dinicu, C. (2020). High-dimensional inference for cluster-based graphical models. *Journal of Machine Learning Research*, 21(53), 1–55.
- Friedman, J., Hastie, T., & Tibshirani, R. (2008). Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3), 432–441.
- Gower, J. C. & Groenen, P. J. F. (1991). Applications of the modified Leverrier-Faddeev algorithm for the construction of explicit matrix decompositions and inverses. *Utilitas Mathematica*, 40, 51–64.
- Grechkin, M., Fazel, M., Witten, D., & Lee, S.-I. (2015). Pathway graphical lasso. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29.
- Hocking, T. D., Joulin, A., Bach, F., & Vert, J.-P. (2011). Clusterpath: An algorithm for clustering using convex fusion penalties. In *The 28th International Conference on Machine Learning* Bellevue, Washington.
- Hubert, L. & Arabie, P. (1985). Comparing partitions. *Journal of Classification*, 2(1), 193–218.
- Lindsten, F., Ohlsson, H., & Ljung, L. (2011). *Just relax and come clustering!: A convexification of k-means clustering*. Technical report, Department of Electrical Engineering, Linköping University, Linköping, Sweden.
- Martin, R. A., Puhlik-Doris, P., Larsen, G., Gray, J., & Weir, K. (2003). Individual differences in uses of humor and their relation to psychological well-being: Development of the humor styles questionnaire. *Journal of Research in Personality*, 37(1), 48–75.
- Meinshausen, N. & Bühlmann, P. (2006). High-dimensional graphs and variable selection with the lasso. *The Annals of Statistics*, 34(3), 1436–1462.
- Mestres, A. C., Bochkina, N., & Mayer, C. (2018). Selection of the regularization parameter in graphical models using network characteristics. *Journal of Computational and Graphical Statistics*, 27(2), 323–333.
- Millington, T. & Niranjana, M. (2019). Quantifying influence in financial markets via partial correlation network inference. In *2019 11th International Symposium on Image and Signal Processing and Analysis (ISPA)* (pp. 306–311).: IEEE.

- Parkinson, M. (1980). The extreme value method for estimating the variance of the rate of return. *The Journal of Business*, 53(1), 61–65.
- Pelckmans, K., De Brabanter, J., Suykens, J. A. K., & de Moor, B. (2005). Convex clustering shrinkage. In *PASCAL Workshop on Statistics and Optimization of Clustering Workshop*.
- Peng, J., Wang, P., Zhou, N., & Zhu, J. (2009). Partial correlation estimation by joint sparse regression models. *Journal of the American Statistical Association*, 104(486), 735–746.
- Pircalabelu, E. & Claeskens, G. (2020). Community-based group graphical lasso. *Journal of Machine Learning Research*, 21(1), 1–32.
- R Core Team (2024). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.
- Rothman, A. J., Bickel, P. J., Levina, E., & Zhu, J. (2008). Sparse permutation invariant covariance estimation. *Electronic Journal of Statistics*, 2, 494–515.
- Shan, L., Qiao, Z., Cheng, L., & Kim, I. (2020). Joint estimation of the two-level Gaussian graphical models across multiple classes. *Journal of Computational and Graphical Statistics*, 29(3), 562–579.
- Shi, D., Wang, T., & Ying, Z. (2024). Simultaneous identification of sparse structures and communities in heterogeneous graphical models. *arXiv preprint arXiv:2405.09841*.
- Tan, K. M., Witten, D., & Shojaie, A. (2015). The cluster graphical lasso for improved estimation of Gaussian graphical models. *Computational Statistics & Data Analysis*, 85, 23–36.
- Tarzanagh, D. A. & Michailidis, G. (2018). Estimation of graphical models through structured norm minimization. *Journal of Machine Learning Research*, 18(1), 1–48.
- Touw, D. J. W., Groenen, P. J. F., & Terada, Y. (2022). Convex clustering through MM: An efficient algorithm to perform hierarchical clustering. *arXiv preprint arXiv:2211.01877*.
- Wang, B., Zhang, Y., Sun, W. W., & Fang, Y. (2018). Sparse convex clustering. *Journal of Computational and Graphical Statistics*, 27(2), 393–403.
- Weylandt, M., Nagorski, J., & Allen, G. I. (2020). Dynamic visualization and fast computation for convex clustering via algorithmic regularization. *Journal of Computational and Graphical Statistics*, 29(1), 87–96.
- Wilms, I. & Bien, J. (2022). Tree-based node aggregation in sparse graphical models. *Journal of Machine Learning Research*, 23(243), 1–36.
- Witten, D. M., Friedman, J. H., & Simon, N. (2011). New insights and faster computations for the graphical lasso. *Journal of Computational and Graphical Statistics*, 20(4), 892–900.
- Yao, T. & Allen, G. I. (2019). Clustered Gaussian graphical model via symmetric convex clustering. In *2019 IEEE Data Science Workshop (DSW)* (pp. 76–82).
- Yuan, M. (2008). Efficient computation of ℓ_1 regularized estimates in gaussian graphical models. *Journal of Computational and Graphical Statistics*, 17(4), 809–826.
- Yuan, M. & Lin, Y. (2006). Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B*, 68(1), 49–67.
- Yuan, M. & Lin, Y. (2007). Model selection and estimation in the Gaussian graphical model. *Biometrika*, 94(1), 19–35.

A Derivations for the Clusterpath Estimator

A.1 Re-expressing the Objective Function $L(\Theta)$

We detail how the objective function in terms of Θ , namely

$$L(\Theta) = -\log |\Theta| + \text{tr} \mathbf{S}\Theta + \mathcal{P}(\Theta), \quad (8)$$

can be re-expressed in terms of (\mathbf{A}, \mathbf{R}) by using the G -block format $\Theta = \mathbf{U}\mathbf{R}\mathbf{U}^\top + \mathbf{A}$ for a certain number of clusters K , where \mathbf{U} is the membership matrix with $u_{jk} = 1$ if variable j belongs to cluster k and zero otherwise, $\mathbf{R} = (r_{k\ell})_{1 \leq k, \ell \leq K}$ is a symmetric matrix, and $\mathbf{A} = \text{diag}(a_{11} \cdot \mathbf{I}, \dots, a_{KK} \cdot \mathbf{I})$ is a diagonal matrix. The main purpose of this re-expression is to obtain a more compact expression that permits computationally efficient updates for solving the optimization problem

$$\hat{\Theta} = \underset{\Theta}{\text{argmin}} L(\Theta) \quad \text{s.t. } \Theta = \Theta^\top, \Theta \succ 0.$$

To this end, we begin by defining the key variables and then work our way through the different terms of the objective function (8).

For any number of clusters $1 \leq K \leq p$, the precision matrix displays the block-structure

$$\Theta = \begin{bmatrix} a_{11}\mathbf{I} + r_{11}\mathbf{1}\mathbf{1}^\top & r_{12}\mathbf{1}\mathbf{1}^\top & \cdots & r_{1K}\mathbf{1}\mathbf{1}^\top \\ r_{21}\mathbf{1}\mathbf{1}^\top & a_{22}\mathbf{I} + r_{22}\mathbf{1}\mathbf{1}^\top & \cdots & r_{2K}\mathbf{1}\mathbf{1}^\top \\ \vdots & \vdots & \ddots & \vdots \\ r_{K1}\mathbf{1}\mathbf{1}^\top & r_{K2}\mathbf{1}\mathbf{1}^\top & \cdots & a_{KK}\mathbf{I} + r_{KK}\mathbf{1}\mathbf{1}^\top \end{bmatrix}. \quad (9)$$

Without loss of generality, we assume that the variables have been reordered such that those in cluster 1 form the first block, those in cluster 2 the second block, and so on. Let $\mathbf{P} = \mathbf{U}^\top \mathbf{U}$ be the diagonal matrix with the number of variables p_k per cluster as diagonal elements, $\mathbf{P}^{1/2}$ then simply contains their square roots, and $\mathbf{p} = \mathbf{1}^\top \mathbf{U}$ is the $K \times 1$ vector with elements p_k . It can be verified that $\mathbf{X}\mathbf{U}\mathbf{P}^{-1}$ computes the cluster averages of the original variables in \mathbf{X} .

Log determinant term. First, we focus on the first term in objective function (8), the log determinant of Θ , to obtain a convenient expression in \mathbf{A} and \mathbf{R} . We show that Θ can be split in a linear space that depends on \mathbf{A} and \mathbf{R} and an orthogonal linear space that is only dependent on the diagonal blocks of Θ and thus on \mathbf{A} and the diagonal elements of \mathbf{R} . To do so, we first consider what happens if we take cluster averages, that is,

$$\begin{aligned} \mathbf{P}^{-1}\mathbf{U}^\top \Theta \mathbf{U}\mathbf{P}^{-1} &= \mathbf{P}^{-1}\mathbf{U}^\top \left(\mathbf{U}\mathbf{R}\mathbf{U}^\top + \begin{bmatrix} a_{11}\mathbf{I} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & a_{22}\mathbf{I} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & a_{KK}\mathbf{I} \end{bmatrix} \right) \mathbf{U}\mathbf{P}^{-1} \\ &= \mathbf{R} + \mathbf{P}^{-1}\mathbf{A} \\ &= \mathbf{R}^*. \end{aligned}$$

Then,

$$\mathbf{U}\mathbf{R}^*\mathbf{U}^\top = \begin{bmatrix} r_{11}^*\mathbf{1}\mathbf{1}^\top & r_{12}\mathbf{1}\mathbf{1}^\top & \cdots & r_{1K}\mathbf{1}\mathbf{1}^\top \\ r_{21}\mathbf{1}\mathbf{1}^\top & r_{22}^*\mathbf{1}\mathbf{1}^\top & \cdots & r_{2K}\mathbf{1}\mathbf{1}^\top \\ \vdots & \vdots & \ddots & \vdots \\ r_{K1}\mathbf{1}\mathbf{1}^\top & r_{K2}\mathbf{1}\mathbf{1}^\top & \cdots & r_{KK}^*\mathbf{1}\mathbf{1}^\top \end{bmatrix},$$

where $r_{kk}^* = a_{kk}/p_k + r_{kk}$ so that $\Theta - \mathbf{UR}^*\mathbf{U}^\top$ has off-diagonal blocks equal to zero and diagonal blocks equal to $a_{kk}(\mathbf{I} - p_k^{-1}\mathbf{1}\mathbf{1}^\top) = a_{kk}\mathbf{J}_k$ with \mathbf{J}_k the $p_k \times p_k$ centering matrix. This yields

$$\Theta = \mathbf{UR}^*\mathbf{U}^\top + \begin{bmatrix} a_{11}\mathbf{J}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & a_{22}\mathbf{J}_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & a_{KK}\mathbf{J}_K \end{bmatrix}. \quad (10)$$

This presents an orthogonal decomposition where the precision matrix is split in two parts, the first part is a linear space that depends on \mathbf{A} and all elements of \mathbf{R} , the second part is an orthogonal linear space that only depends on the diagonal blocks of the precision matrix, namely \mathbf{A} . It can be verified that this decomposition is indeed orthogonal because post-multiplying the final term with \mathbf{U} yields zero due to the centering matrices.

Next, it is well known that the determinant is equal to the product of the eigenvalues of the matrix. Therefore, we require the eigenvalues of both parts of decomposition (10). Let the eigendecomposition of Θ be given by

$$\begin{aligned} \Theta &= \mathbf{Q}\mathbf{\Gamma}\mathbf{Q}^\top \\ &= \mathbf{Q}_1\mathbf{\Gamma}_1\mathbf{Q}_1^\top + \mathbf{Q}_2\mathbf{\Gamma}_2\mathbf{Q}_2^\top \\ &= \mathbf{UR}^*\mathbf{U}^\top + \begin{bmatrix} a_{11}\mathbf{J}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & a_{22}\mathbf{J}_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & a_{KK}\mathbf{J}_K \end{bmatrix}. \end{aligned}$$

We now need to find $\mathbf{\Gamma}_1$ in the eigendecomposition $\mathbf{UR}^*\mathbf{U}^\top = \mathbf{Q}_1\mathbf{\Gamma}_1\mathbf{Q}_1^\top$. Pre- and post-multiplying with an orthonormal matrix does not change the eigenvalues. A convenient orthonormal matrix for pre-multiplication is therefore $\mathbf{P}^{-1/2}\mathbf{U}$ giving

$$\begin{aligned} \mathbf{UR}^*\mathbf{U}^\top &= \mathbf{Q}_1\mathbf{\Gamma}_1\mathbf{Q}_1^\top \\ \mathbf{P}^{-1/2}\mathbf{U}^\top\mathbf{UR}^*\mathbf{U}^\top\mathbf{UP}^{-1/2} &= \mathbf{P}^{-1/2}\mathbf{U}^\top\mathbf{Q}_1\mathbf{\Gamma}_1\mathbf{Q}_1^\top\mathbf{UP}^{-1/2} \\ \mathbf{P}^{1/2}\mathbf{R}^*\mathbf{P}^{1/2} &= \mathbf{Q}_1^*\mathbf{\Gamma}_1\mathbf{Q}_1^{*\top} \\ \begin{bmatrix} p_1r_{11}^* & (p_1p_2)^{1/2}r_{12} & \cdots & (p_1p_K)^{1/2}r_{1K} \\ (p_1p_2)^{1/2}r_{21} & p_2r_{22}^* & \cdots & (p_2p_K)^{1/2}r_{2K} \\ \vdots & \vdots & \ddots & \vdots \\ (p_1p_K)^{1/2}r_{K1} & (p_2p_K)^{1/2}r_{K2} & \cdots & p_Kr_{KK}^* \end{bmatrix} &= \mathbf{Q}_1^*\mathbf{\Gamma}_1\mathbf{Q}_1^{*\top}, \end{aligned}$$

which shows that in the case of K clusters, the eigenvalues corresponding to the space spanned by $\mathbf{UR}^*\mathbf{U}^\top$ in decomposition (10) can be obtained by considering the eigenvalues of a $K \times K$ matrix.

The second part of the space in decomposition (10) is spanned by the diagonal centering matrices $a_{kk}\mathbf{J}_k$. The centering matrix is a projector matrix that has $p_k - 1$ eigenvalues of 1, so that $a_{kk}\mathbf{J}_k$ has $p_k - 1$ eigenvalues of a_{kk} .

Combining both parts, we can express

$$-\log|\Theta| = -\log|\mathbf{P}^{1/2}\mathbf{R}^*\mathbf{P}^{1/2}| - \sum_{k=1}^K (p_k - 1) \log a_{kk}.$$

This decomposition shows that the eigenvalues of Θ consist of the eigenvalues of $\mathbf{P}^{1/2}\mathbf{R}^*\mathbf{P}^{1/2}$, together with a_{kk} , which appears $p_k - 1$ times for each $1 \leq k \leq K$. Therefore, if \mathbf{R}^* is positive definite and $a_{kk} > 0$, it follows that Θ is also positive definite.

Trace term. Second, we address the second term in objective function (8), the trace of the matrix product $\mathbf{S}\Theta$. It is straightforward to express this term in terms of \mathbf{A} and \mathbf{R} , namely

$$\text{tr } \mathbf{S}\Theta = \text{tr } \mathbf{S}\mathbf{U}\mathbf{R}\mathbf{U}^\top + \sum_{\ell=1}^K a_{\ell\ell} \text{tr } \mathbf{S}_\ell,$$

where \mathbf{S}_ℓ is the sample covariance matrix computed from the p_ℓ variables in cluster ℓ .

Aggregation penalty. Third, the aggregation penalty term in objective function (8) can be written as

$$\sum_{j=1}^p \sum_{j'=1}^{j-1} w_{jj'} d_{jj'}(\Theta) = \sum_{k=1}^K \sum_{\ell=1}^{k-1} \sum_{j \in \mathcal{C}_k} \sum_{j' \in \mathcal{C}_\ell} w_{jj'} d_{jj'}(\Theta),$$

where \mathcal{C}_k denotes the set of variables belonging to cluster k . Because $d_{jj'}(\Theta)$ evaluates to the same value for all $j \in \mathcal{C}_k$ and $j' \in \mathcal{C}_\ell$, the penalty can be further reduced into

$$\sum_{k=1}^K \sum_{\ell=1}^{k-1} \left(\sum_{j \in \mathcal{C}_k} \sum_{j' \in \mathcal{C}_\ell} w_{jj'} \right) d_{\mathcal{C}_k \mathcal{C}_\ell}(\mathbf{A}, \mathbf{R}) = \sum_{k=1}^K \sum_{\ell=1}^{k-1} \mathbf{u}_k^\top \mathbf{W} \mathbf{u}_\ell d_{\mathcal{C}_k \mathcal{C}_\ell}(\mathbf{A}, \mathbf{R}),$$

where \mathbf{W} is the symmetric $p \times p$ weight matrix containing the individual weights $w_{jj'}$, \mathbf{u}_k denotes the k^{th} column of the membership matrix \mathbf{U} , and

$$\begin{aligned} d_{\mathcal{C}_k \mathcal{C}_\ell}^2(\mathbf{A}, \mathbf{R}) &= (a_{kk} + r_{kk} - a_{\ell\ell} - r_{\ell\ell})^2 + (p_k - 1)(r_{kk} - r_{k\ell})^2 + (p_\ell - 1)(r_{\ell\ell} - r_{k\ell})^2 \\ &+ \sum_{\substack{m=1 \\ m \notin \{k, \ell\}}}^K p_m (r_{km} - r_{\ell m})^2. \end{aligned}$$

Note that the subscripts $\mathcal{C}_k \mathcal{C}_\ell$ in $d_{\mathcal{C}_k \mathcal{C}_\ell}(\mathbf{A}, \mathbf{R})$ always refer to the cluster formulation of the Euclidean distance.

Sparsity penalty. Fourth, consider the sparsity penalty in objective function (8). By setting $z_{jj} = 0$ to avoid penalization of the diagonal elements of Θ , the sparsity penalty can be written as

$$\sum_{\substack{j, j' \\ j \neq j'}} z_{jj'} |\theta_{jj'}| = \sum_{k=1}^K \sum_{\ell=1}^K \sum_{j \in \mathcal{C}_k} \sum_{j' \in \mathcal{C}_\ell} z_{jj'} |\theta_{jj'}|.$$

Using the block structure of Θ , this expression can be reformulated in terms of \mathbf{R} via

$$\sum_{k=1}^K \sum_{\ell=1}^K \left(\sum_{j \in \mathcal{C}_k} \sum_{j' \in \mathcal{C}_\ell} z_{jj'} \right) |r_{k\ell}| = \sum_{k=1}^K \sum_{\ell=1}^K \mathbf{u}_k \mathbf{Z} \mathbf{u}_\ell^\top |r_{k\ell}|,$$

where \mathbf{Z} is the symmetric $p \times p$ weight matrix containing the sparsity weights $z_{jj'}$.

Finally, combining all decomposed terms we obtain the expression for the objective function $L(\Theta)$ in terms of \mathbf{A} and \mathbf{R} , namely

$$\begin{aligned} L(\Theta) &= L(\mathbf{A}, \mathbf{R}) \\ &= -\log |\mathbf{P}^{1/2} \mathbf{R}^* \mathbf{P}^{1/2}| - \sum_{k=1}^K (p_k - 1) \log a_{kk} + \text{tr } \mathbf{S}\mathbf{U}\mathbf{R}\mathbf{U}^\top \\ &\quad + \sum_{\ell=1}^K a_{\ell\ell} \text{tr } \mathbf{S}_\ell + \lambda_c \sum_{k=1}^K \sum_{\ell=1}^{k-1} \mathbf{u}_k^\top \mathbf{W} \mathbf{u}_\ell d_{\mathcal{C}_k \mathcal{C}_\ell}(\mathbf{A}, \mathbf{R}) + \lambda_s \sum_{k=1}^K \sum_{\ell=1}^K \mathbf{u}_k \mathbf{Z} \mathbf{u}_\ell^\top |r_{k\ell}|. \end{aligned} \quad (11)$$

A.2 Re-expressing the Objective Function $L(\Theta_k)$

In Section A.1, efficient expressions $L(\mathbf{A}, \mathbf{R})$ for $L(\Theta)$ have been obtained that make use of the block structure of Θ . What remains to be done is to separate $L(\mathbf{A}, \mathbf{R})$ into those parts that depend on cluster k and those that do not. This separation is needed to cycle through the K blocks, to—in turn—update the objective function $L(\Theta_k)$ for block/cluster k . In the following, we make the assumption, without loss of generality, that the variables are arranged to consistently position cluster k as the last cluster.

Consider the precision matrix split according to the elements that belong to cluster k (Θ_k) and those that do not (Θ_{-k})

$$\begin{aligned}\Theta &= \Theta_{-k} + \Theta_k = \left[\begin{array}{c|c} \Theta_{00} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{0} \end{array} \right] + \left[\begin{array}{c|c} \mathbf{0} & \Theta_{0k} \\ \hline \Theta_{0k}^\top & \Theta_{kk} \end{array} \right] \\ &= \left[\begin{array}{c|c} \Theta_{00} & \Theta_{0k} \\ \hline \Theta_{0k}^\top & \Theta_{kk} \end{array} \right] \\ &= \left[\begin{array}{ccc|c} a_{11}\mathbf{I} + r_{11}\mathbf{1}\mathbf{1}^\top & r_{12}\mathbf{1}\mathbf{1}^\top & \cdots & r_{13}\mathbf{1}\mathbf{1}^\top \\ r_{21}\mathbf{1}\mathbf{1}^\top & a_{22}\mathbf{I} + r_{22}\mathbf{1}\mathbf{1}^\top & \cdots & r_{2k}\mathbf{1}\mathbf{1}^\top \\ \vdots & \vdots & \ddots & \vdots \\ r_{k1}\mathbf{1}\mathbf{1}^\top & r_{k2}\mathbf{1}\mathbf{1}^\top & \cdots & a_{kk}\mathbf{I} + r_{kk}\mathbf{1}\mathbf{1}^\top \end{array} \right]\end{aligned}$$

Consequently, updating all parameters in \mathbf{A} and \mathbf{R} that are associated with cluster k updates an entire block of variables in Θ . To separate the term $-\log |\mathbf{P}^{1/2} \mathbf{R}^* \mathbf{P}^{1/2}|$ in objective function (11), it helps to write $\mathbf{P}^{1/2} \mathbf{R}^* \mathbf{P}^{1/2}$ as the 2×2 block matrix

$$\mathbf{P}^{1/2} \mathbf{R}^* \mathbf{P}^{1/2} = \left[\begin{array}{c|c} \mathbf{P}_0^{1/2} \mathbf{R}_0^* \mathbf{P}_0^{1/2} & p_k^{1/2} \mathbf{P}_0^{1/2} \mathbf{r}_k \\ \hline p_k^{1/2} (\mathbf{P}_0^{1/2} \mathbf{r}_k)^\top & p_k r_{kk}^* \end{array} \right]$$

with \mathbf{r}_k the $K - 1$ vector containing all elements in the k^{th} column of \mathbf{R} apart from r_{kk} , and

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_0 & \mathbf{0} \\ \mathbf{0}^\top & p_k \end{bmatrix}.$$

From linear algebra, we have the following expression for the log determinant of the 2×2 block matrix

$$-\log |\mathbf{P}^{1/2} \mathbf{R}^* \mathbf{P}^{1/2}| = -\log |\mathbf{P}_0^{1/2} \mathbf{R}_0^* \mathbf{P}_0^{1/2}| - \log(a_{kk} + p_k r_{kk} - p_k \mathbf{r}_k^\top (\mathbf{R}_0^*)^{-1} \mathbf{r}_k). \quad (12)$$

This allows us to rewrite the negative log determinant of Θ as

$$\begin{aligned}-\log |\Theta| &= -\log |\mathbf{P}_0^{1/2} \mathbf{R}_0^* \mathbf{P}_0^{1/2}| - \log(a_{kk} + p_k r_{kk} - p_k \mathbf{r}_k^\top (\mathbf{R}_0^*)^{-1} \mathbf{r}_k) \\ &\quad - (p_k - 1) \log a_{kk} - \sum_{\substack{\ell=1 \\ \ell \neq k}}^K (p_\ell - 1) \log a_{\ell\ell}.\end{aligned}$$

Using the re-expression from objective function (11), the trace of $\mathbf{S}\Theta$ separates readily into

$$\text{tr } \mathbf{S}\Theta = 2\mathbf{u}_k^\top \mathbf{S}\mathbf{U}_0 \mathbf{r}_k + \mathbf{u}_k^\top \mathbf{S}\mathbf{u}_k r_{kk} + a_{kk} \text{tr } \mathbf{S}_k + \text{tr } \mathbf{S}\mathbf{U}_0 \mathbf{R}_0 \mathbf{U}_0^\top + \sum_{\substack{\ell=1 \\ \ell \neq k}}^K a_{\ell\ell} \text{tr } \mathbf{S}_\ell.$$

It can be verified that the last two terms are constant in a_{kk} , \mathbf{r}_k , and r_{kk} , and the remaining terms are linear in these unknowns. Additionally, the sparsity penalty can be written as

$$\sum_{k=1}^K \sum_{\ell=1}^K \mathbf{u}_k \mathbf{Z} \mathbf{u}_\ell^\top |r_{k\ell}| = \sum_{\ell=1}^K \mathbf{u}_k \mathbf{Z} \mathbf{u}_\ell^\top |r_{k\ell}| + \sum_{\ell, \ell' \neq k} \mathbf{u}_\ell \mathbf{Z} \mathbf{u}_{\ell'}^\top |r_{\ell\ell'}|.$$

Finally, due to the symmetry of \mathbf{R} , the clusterpath penalty is not separable into parts dependent and independent of cluster k .

Combining the foregoing results, objective function $L(\boldsymbol{\Theta}_k)$ can be expressed in terms of parts a_{kk} , \mathbf{r}_k , and r_{kk} , and we obtain

$$\begin{aligned} L(a_{kk}, \mathbf{r}_k, r_{kk}) &= L_{\det}(a_{kk}, \mathbf{r}_k, r_{kk}) + L_{\text{cov}}(a_{kk}, \mathbf{r}_k, r_{kk}) + L_{\text{clust}}(a_{kk}, \mathbf{r}_k, r_{kk}) \\ &\quad + L_{\text{sparse}}(\mathbf{r}_k, r_{kk}) + C \\ &= -\log(a_{kk} + p_k r_{kk} - p_k \mathbf{r}_k^\top (\mathbf{R}_0^*)^{-1} \mathbf{r}_k) - (p_k - 1) \log a_{kk} \\ &\quad + 2\mathbf{u}_k^\top \mathbf{S} \mathbf{U}_0 \mathbf{r}_k + \mathbf{u}_k^\top \mathbf{S} \mathbf{u}_k r_{kk} + a_{kk} \text{tr} \mathbf{S}_k \\ &\quad + \lambda_c \sum_{k=1}^K \sum_{\ell=1}^{k-1} \mathbf{u}_k^\top \mathbf{W} \mathbf{u}_\ell d_{\mathcal{C}_k \mathcal{C}_\ell}(\mathbf{A}, \mathbf{R}) + \lambda_s \sum_{\ell=1}^K \mathbf{u}_k \mathbf{Z} \mathbf{u}_\ell^\top |r_{k\ell}| + C \end{aligned}$$

with

$$\begin{aligned} C &= -\log |\mathbf{P}_0^{1/2} \mathbf{R}_0^* \mathbf{P}_0^{1/2}| - \sum_{\substack{\ell=1 \\ \ell \neq k}}^K (p_\ell - 1) \log a_{\ell\ell} \\ &\quad + \text{tr} \mathbf{S} \mathbf{U}_0 \mathbf{R}_0 \mathbf{U}_0^\top + \sum_{\substack{\ell=1 \\ \ell \neq k}}^K a_{\ell\ell} \text{tr} \mathbf{S}_\ell + \lambda_s \sum_{\ell, \ell' \neq k} \mathbf{u}_\ell \mathbf{Z} \mathbf{u}_{\ell'}^\top |r_{\ell\ell'}|. \end{aligned}$$

B Derivations and Pseudocode for the Cyclic Block Coordinate Descent Algorithm

We provide a detailed explanation of the minimization procedure for the CGGM objective function. In Appendix B.1, we present the pseudocode of the algorithm. We derive the gradient and Hessian used in the algorithm in Appendix B.2. Finally, Appendix B.3 discusses the computation of a clusterpath, where the CGGM objective is minimized for a sequence of increasing values of the tuning parameter λ_c .

The reparameterization of $\boldsymbol{\Theta}$ into the diagonal matrix \mathbf{A} and the symmetric matrix \mathbf{R} (see Appendix A) allows for straightforward expressions of the objective function and the conditions for positive definiteness of $\boldsymbol{\Theta}$. In this appendix, we take an additional step to facilitate the derivations of the gradient and Hessian used for the minimization.

We define $a_{kk} = b_{kk} - r_{kk}$, so that the diagonal values of $\boldsymbol{\Theta}$ are solely determined by the diagonal matrix $\mathbf{B} = (b_{kk})_{1 \leq k \leq K}$. This allows for more compact representation of the gradient and Hessian with respect to r_{kk} . We work with the following definition of the objective function

$$\begin{aligned} L(b_{kk}, \mathbf{r}_k, r_{kk}) &= -\log(b_{kk} + (p_k - 1)r_{kk} - p_k \mathbf{r}_k^\top (\mathbf{R}_0^*)^{-1} \mathbf{r}_k) - (p_k - 1) \log(b_{kk} - r_{kk}) \\ &\quad + 2\mathbf{u}_k^\top \mathbf{S} \mathbf{U}_0 \mathbf{r}_k + \mathbf{u}_k^\top \mathbf{S} \mathbf{u}_k r_{kk} + (b_{kk} - r_{kk}) \text{tr} \mathbf{S}_k \\ &\quad + \lambda_c \sum_{k=1}^K \sum_{\ell=1}^{k-1} \mathbf{u}_k^\top \mathbf{W} \mathbf{u}_\ell d_{\mathcal{C}_k \mathcal{C}_\ell}(\mathbf{B}, \mathbf{R}) + \lambda_s \sum_{\ell=1}^K \mathbf{u}_k \mathbf{Z} \mathbf{u}_\ell^\top |r_{k\ell}| + C \end{aligned} \tag{13}$$

with

$$d_{\mathcal{C}_k \mathcal{C}_\ell}^2(\mathbf{B}, \mathbf{R}) = (b_{kk} - b_{\ell\ell})^2 + (p_k - 1)(r_{kk} - r_{k\ell})^2 + (p_\ell - 1)(r_{\ell\ell} - r_{k\ell})^2 + \sum_{\substack{m=1 \\ m \notin \{k, \ell\}}}^K p_m (r_{km} - r_{\ell m})^2.$$

B.1 Pseudocode for the Algorithm

Algorithm 1 contains the pseudocode for the cyclic block coordinate descent algorithm to compute CGGM.

Algorithm 1 Pseudocode for the algorithm that minimizes the CGGM objective function for fixed λ_c and λ_s

Input Initial estimates for $\mathbf{B}^{(1)}$ and $\mathbf{R}^{(1)}$, sample covariance matrix \mathbf{S} , weight matrix \mathbf{W} , tuning parameters λ_c and λ_s , fusion threshold ε_f , convergence threshold ε_c , maximum number of iterations t_{\max}

Output $\hat{\Theta}$ that minimizes $L(\Theta)$ and \hat{K} the number of clusters

```

1:  $\mathbf{U}^{(1)} \leftarrow \mathbf{I}$ 
2:  $L^{(1)} \leftarrow L(\mathbf{B}^{(1)}, \mathbf{R}^{(1)})$ 
3:  $L^{(0)} \leftarrow (1 + 2\varepsilon_c)L^{(1)}$ 
4:  $K \leftarrow p$ 
5:  $t \leftarrow 1$ 
6: while  $L^{(t-1)}/L^{(t)} - 1 > \varepsilon_c$  and  $t \leq t_{\max}$  do
7:    $t \leftarrow t + 1$ 
8:    $\mathbf{B}^{(t)} \leftarrow \mathbf{B}^{(t-1)}$ 
9:    $\mathbf{R}^{(t)} \leftarrow \mathbf{R}^{(t-1)}$ 
10:   $\mathbf{U}^{(t)} \leftarrow \mathbf{U}^{(t-1)}$ 
11:  for  $k = 1, \dots, K$  do
12:     $\ell \leftarrow \operatorname{argmin}_{\ell'} d_{k\ell'}(\mathbf{B}^{(t)}, \mathbf{R}^{(t)})$ 
13:    if  $d_{C_k C_\ell}(\mathbf{B}^{(t)}, \mathbf{R}^{(t)}) \leq \varepsilon_f$  then
14:      Fuse clusters  $k$  and  $\ell$  by modifying  $\mathbf{B}^{(t)}, \mathbf{R}^{(t)}, \mathbf{U}^{(t)}$ 
15:       $K \leftarrow K - 1$ 
16:    else
17:       $\boldsymbol{\delta}_k \leftarrow -\nabla^2 L(b_{kk}^{(t)}, \mathbf{r}_k^{(t)}, r_{kk}^{(t)})^{-1} \nabla L(b_{kk}^{(t)}, \mathbf{r}_k^{(t)}, r_{kk}^{(t)})$ 
18:      Compute maximum step size  $s_{\max}$  using equations (14) and (15)
19:      Select optimal step size  $s^* \in [0, s_{\max}]$ 
20:       $[b_{kk}^{(t)}, \mathbf{r}_k^{(t)}, r_{kk}^{(t)}] \leftarrow [b_{kk}^{(t)}, \mathbf{r}_k^{(t)}, r_{kk}^{(t)}] + s^* \boldsymbol{\delta}_k$ 
21:   $\hat{\mathbf{B}} \leftarrow \mathbf{B}^{(t)}$ 
22:   $\hat{\mathbf{R}} \leftarrow \mathbf{R}^{(t)}$ 
23:   $\hat{K} \leftarrow K$ 
24:   $\hat{\mathbf{A}} \leftarrow \operatorname{diag}(\hat{a}_{11}\mathbf{I}, \dots, \hat{a}_{KK}\mathbf{I})$  where  $\hat{a}_{kk} = \hat{b}_{kk} - \hat{r}_{kk}$ 
25:   $\hat{\mathbf{U}} \leftarrow \mathbf{U}^{(t)}$ 
26:   $\hat{\Theta} \leftarrow \hat{\mathbf{U}}\hat{\mathbf{R}}\hat{\mathbf{U}}^\top + \hat{\mathbf{A}}$ 

```

The algorithm requires initialization; the usage of high-quality initializations is important for the stability of the algorithm in terms of convergence. To initialize \mathbf{B} and \mathbf{R} for $\lambda_c = \lambda_s = 0$, one can use the inverse of the sample covariance matrix if it is available; then $\mathbf{B}^{(1)} = \text{diag}(\mathbf{S}^{-1})$ and $\mathbf{R}^{(1)} = \mathbf{S}^{-1}$. In other cases, one can start from $(\mathbf{S} + \mathbf{I})^{-1}$ or a regularized estimator such as the graphical lasso (e.g., Friedman et al., 2008; Witten et al., 2011). Next, when computing the clusterpath, namely a sequence of solutions for Θ for an increasing clustering parameter λ_c (for fixed λ_s), we leverage warm starts to ensure high-quality initializations. Indeed, after solving the minimization problem for a particular λ_c , we use that solution as initialization when solving the optimization problem for the next (higher) value of λ_c in the clusterpath. Loss function convergence of the algorithm was observed across all numerical experiments and empirical applications.

After the required initializations, the algorithm first checks for eligible clusters fusions using the fusion threshold ε_f (lines 12–15). The value of ε_f can be a small user-defined value, such as 10^{-3} , or based on the data itself. If the inverse of the sample covariance matrix is available, a data-driven choice is $\varepsilon_f = \tau \text{median}_{j,j'}(d_{jj'}(\mathbf{S}^{-1}))$, with $\tau = 10^{-3}$. If a fusion is performed, the optimization parameters are modified and the total number of clusters K is reduced by one. In case no cluster fusion occurs, the algorithm proceeds to the second step where the parameters pertaining to cluster k , denoted by the vector $[b_{kk}, \mathbf{r}_k, r_{kk}]$, are jointly updated (lines 17–20). Part of the update is a line search by means of a golden section search for the optimal step size s^* , which ensures the positive definiteness of Θ .

As mentioned in Appendix A, Θ is positive definite if \mathbf{R}^* is positive definite and $a_{kk} = b_{kk} + r_{kk} > 0$. Assume that \mathbf{R}^* is positive definite for the current values of $[b_{kk}, \mathbf{r}_k, r_{kk}]$. Then, Sylvester’s Criterion states that the upper left 1×1 corner of \mathbf{R}^* has a positive determinant, so does the 2×2 upper left corner, and so on until \mathbf{R}^* itself has a positive determinant. Without loss of generality, let the clusters be arranged so that k is the last cluster. Updating $[b_{kk}, \mathbf{r}_k, r_{kk}]$ does not alter the 1×1 through $(K - 1) \times (K - 1)$ upper left corners. Consequently, a positive determinant of \mathbf{R}^* after the update of the parameters related to cluster k is sufficient to keep \mathbf{R}^* positive definite.

Putting this together, the positive definiteness of Θ is preserved by a step size s that satisfies the aforementioned conditions. Using the decomposition of the log determinant in (12) and the expression on the first line of the objective in (13), this is achieved by a step size that satisfies the inequalities

$$\begin{aligned} (b_{kk} + s\delta_{b_{kk}}) + (p_k - 1)(r_{kk} + s\delta_{r_{kk}}) - p_k(\mathbf{r}_k + s\delta_{\mathbf{r}_k})^\top (\mathbf{R}^*)^{-1}(\mathbf{r}_k + s\delta_{\mathbf{r}_k}) &> 0 \\ b_{kk} + s\delta_{b_{kk}} - r_{kk} - s\delta_{r_{kk}} &> 0, \end{aligned} \quad (14)$$

where $\delta_{b_{kk}}$, $\delta_{\mathbf{r}_k}$, and $\delta_{r_{kk}}$ represent the descent directions for b_{kk} , \mathbf{r}_k , and r_{kk} . In case there is a single cluster left ($K = 1$), the first inequality reduces to

$$(b_{kk} + s\delta_{b_{kk}}) + (p_k - 1)(r_{kk} + s\delta_{r_{kk}}) > 0. \quad (15)$$

After the loss function has converged, the output of the algorithm consists of the estimated number of clusters \hat{K} and the estimates $\hat{\mathbf{B}}$, $\hat{\mathbf{R}}$, and $\hat{\mathbf{U}}$, which can be used to construct $\hat{\Theta}$.

The computational complexity of the algorithm is primarily determined by the computation of the descent direction δ_k , which requires solving a system of equations with a complexity of $\mathcal{O}(K^3)$. This step dominates the overall update cost, including the line search. To determine s^* , we use the golden section search, which iteratively shrinks the interval $[0, s_{\max}]$ to a pre-specified tolerance of $5 \cdot 10^{-3}$. During this search, the objective for cluster k in (13) is repeatedly evaluated, incurring a per-evaluation complexity of $\mathcal{O}(K^2)$. Consequently, a complete pass over all clusters results in a total complexity of $\mathcal{O}(K^4)$. Since the number of clusters K is bounded by the number of variables p , the overall complexity is capped at $\mathcal{O}(p^4)$.

B.2 Derivations of the Gradient and Hessian

For simplicity, we use the following smoothed version of the absolute value function for the sparsity penalty

$$|r_{k\ell}| = \begin{cases} \frac{r_{k\ell}^2 + \varepsilon^2}{2\varepsilon} & \text{if } |r_{k\ell}| < \varepsilon, \\ |r_{k\ell}| & \text{otherwise.} \end{cases}$$

If ε is chosen sufficiently small (e.g., $\varepsilon = 5 \cdot 10^{-3}$), this function closely approximates the absolute value function.

To shorten notation, let $\mathbf{V} = (\mathbf{R}_0^*)^{-1}$ and

$$h(b_{kk}, \mathbf{r}_k, r_{kk}) = b_{kk} + (p_k - 1)r_{kk} - p_k \mathbf{r}_k^\top \mathbf{V} \mathbf{r}_k.$$

For convenience, the elements r_{km} of the vector \mathbf{r}_k are treated individually. Consequently, we should note that $1 \leq m \leq K$, $m \neq k$. For the gradient, we obtain

$$\begin{aligned} \frac{\partial L_{\det}(b_{kk}, \mathbf{r}_k, r_{kk})}{\partial b_{kk}} &= -\frac{1}{h(b_{kk}, \mathbf{r}_k, r_{kk})} - \frac{p_k - 1}{b_{kk} - r_{kk}} \\ \frac{\partial L_{\det}(b_{kk}, \mathbf{r}_k, r_{kk})}{\partial r_{km}} &= \frac{2p_k}{h(b_{kk}, \mathbf{r}_k, r_{kk})} \mathbf{v}_m^\top \mathbf{r}_k \\ \frac{\partial L_{\det}(b_{kk}, \mathbf{r}_k, r_{kk})}{\partial r_{kk}} &= -\frac{p_k - 1}{h(b_{kk}, \mathbf{r}_k, r_{kk})} + \frac{p_k - 1}{b_{kk} - r_{kk}} \\ \frac{\partial L_{\text{cov}}(b_{kk}, \mathbf{r}_k, r_{kk})}{\partial b_{kk}} &= \text{tr } \mathbf{S}_k \\ \frac{\partial L_{\text{cov}}(b_{kk}, \mathbf{r}_k, r_{kk})}{\partial r_{km}} &= 2\mathbf{u}_m^\top \mathbf{S} \mathbf{u}_k \\ \frac{\partial L_{\text{cov}}(b_{kk}, \mathbf{r}_k, r_{kk})}{\partial r_{kk}} &= \mathbf{u}_k^\top \mathbf{S} \mathbf{u}_k - \text{tr } \mathbf{S}_k \\ \frac{\partial L_{\text{clust}}(b_{kk}, \mathbf{r}_k, r_{kk})}{\partial b_{kk}} &= \lambda \sum_{\substack{\ell=1 \\ \ell \neq k}}^K \frac{\mathbf{u}_k^\top \mathbf{W} \mathbf{u}_\ell}{d_{\mathcal{C}_k \mathcal{C}_\ell}(\mathbf{B}, \mathbf{R})} (b_{kk} - b_{\ell\ell}) \\ \frac{\partial L_{\text{clust}}(b_{kk}, \mathbf{r}_k, r_{kk})}{\partial r_{km}} &= \lambda \sum_{\substack{\ell=1 \\ \ell \notin \{k, m\}}}^K \left(\frac{\mathbf{u}_k^\top \mathbf{W} \mathbf{u}_\ell}{d_{\mathcal{C}_k \mathcal{C}_\ell}(\mathbf{B}, \mathbf{R})} p_m (r_{km} - r_{m\ell}) + \frac{\mathbf{u}_m^\top \mathbf{W} \mathbf{u}_\ell}{d_{\mathcal{C}_m \mathcal{C}_\ell}(\mathbf{B}, \mathbf{R})} p_k (r_{km} - r_{k\ell}) \right) \\ &\quad + \lambda \frac{\mathbf{u}_k^\top \mathbf{W} \mathbf{u}_m}{d_{\mathcal{C}_k \mathcal{C}_m}(\mathbf{B}, \mathbf{R})} ((p_k - 1)(r_{km} - r_{kk}) + (p_m - 1)(r_{km} - r_{mm})) \\ \frac{\partial L_{\text{clust}}(b_{kk}, \mathbf{r}_k, r_{kk})}{\partial r_{kk}} &= \lambda (p_k - 1) \sum_{\substack{\ell=1 \\ \ell \neq k}}^K \frac{\mathbf{u}_k^\top \mathbf{W} \mathbf{u}_\ell}{d_{\mathcal{C}_k \mathcal{C}_\ell}(\mathbf{B}, \mathbf{R})} (r_{kk} - r_{k\ell}) \\ \frac{\partial L_{\text{sparse}}(b_{kk}, \mathbf{r}_k, r_{kk})}{\partial r_{k\ell}} &= \begin{cases} \frac{\mathbf{u}_k \mathbf{Z} \mathbf{u}_\ell^\top}{\varepsilon} r_{k\ell} & \text{if } |r_{k\ell}| < \varepsilon, \\ \text{sign}(r_{k\ell}) & \text{otherwise.} \end{cases} \end{aligned}$$

In the derivations for the Hessian, we require an additional index m' to define the off-diagonal elements in the block $\partial^2 L(b_{kk}, \mathbf{r}_k, r_{kk}) / \partial \mathbf{r}_k^2$ that satisfies $1 \leq m' \leq K$, $m' \notin \{k, m\}$. For the Hessian, we obtain

$$\begin{aligned} \frac{\partial^2 L_{\det}(b_{kk}, \mathbf{r}_k, r_{kk})}{\partial b_{kk}^2} &= \frac{1}{h^2(b_{kk}, \mathbf{r}_k, r_{kk})} + \frac{p_k - 1}{(b_{kk} - r_{kk})^2} \\ \frac{\partial^2 L_{\det}(b_{kk}, \mathbf{r}_k, r_{kk})}{\partial b_{kk} \partial r_{km}} &= -\frac{2p_k}{h^2(b_{kk}, \mathbf{r}_k, r_{kk})} \mathbf{v}_m^\top \mathbf{r}_k \\ \frac{\partial^2 L_{\det}(b_{kk}, \mathbf{r}_k, r_{kk})}{\partial b_{kk} \partial r_{kk}} &= \frac{p_k - 1}{h^2(b_{kk}, \mathbf{r}_k, r_{kk})} - \frac{p_k - 1}{(b_{kk} - r_{kk})^2} \\ \frac{\partial^2 L_{\det}(b_{kk}, \mathbf{r}_k, r_{km})}{\partial r_{km}^2} &= \frac{2p_k}{h(b_{kk}, \mathbf{r}_k, r_{kk})} v_{mm} + \frac{4p_k^2}{h^2(b_{kk}, \mathbf{r}_k, r_{kk})} \mathbf{v}_m^\top \mathbf{r}_k \mathbf{r}_k^\top \mathbf{v}_m \\ \frac{\partial^2 L_{\det}(b_{kk}, \mathbf{r}_k, r_{km})}{\partial r_{km} \partial r_{km'}} &= \frac{2p_k}{h(b_{kk}, \mathbf{r}_k, r_{kk})} v_{mm'} + \frac{4p_k^2}{h^2(b_{kk}, \mathbf{r}_k, r_{kk})} \mathbf{v}_m^\top \mathbf{r}_k \mathbf{r}_k^\top \mathbf{v}_{m'} \end{aligned}$$

$$\begin{aligned}
\frac{\partial^2 L_{\text{det}}(b_{kk}, \mathbf{r}_k, r_{kk})}{\partial r_{km} \partial r_{kk}} &= -\frac{2p_k(p_k - 1)}{h^2(b_{kk}, \mathbf{r}_k, r_{kk})} \mathbf{v}_m^\top \mathbf{r}_k \\
\frac{\partial^2 L_{\text{det}}(b_{kk}, \mathbf{r}_k, r_{kk})}{\partial r_{kk}^2} &= \frac{(p_k - 1)^2}{h^2(b_{kk}, \mathbf{r}_k, r_{kk})} + \frac{p_k - 1}{(b_{kk} - r_{kk})^2} \\
\frac{\partial^2 L_{\text{clust}}(b_{kk}, \mathbf{r}_k, r_{kk})}{\partial b_{kk}^2} &= \lambda \sum_{\substack{\ell=1 \\ \ell \neq k}}^K \mathbf{u}_k^\top \mathbf{W} \mathbf{u}_\ell \left(\frac{1}{d_{\mathcal{C}_k \mathcal{C}_\ell}(\mathbf{B}, \mathbf{R})} - \frac{(b_{kk} - b_{\ell\ell})^2}{d_{\mathcal{C}_k \mathcal{C}_\ell}^3(\mathbf{B}, \mathbf{R})} \right) \\
\frac{\partial^2 L_{\text{clust}}(b_{kk}, \mathbf{r}_k, r_{kk})}{\partial b_{kk} \partial r_{km}} &= -\lambda \sum_{\substack{\ell=1 \\ \ell \notin \{k, m\}}}^K \frac{\mathbf{u}_k^\top \mathbf{W} \mathbf{u}_\ell}{d_{\mathcal{C}_k \mathcal{C}_\ell}^3(\mathbf{B}, \mathbf{R})} p_m (b_{kk} - b_{\ell\ell}) (r_{km} - r_{m\ell}) \\
&\quad - \lambda \frac{\mathbf{u}_k^\top \mathbf{W} \mathbf{u}_m}{d_{\mathcal{C}_k \mathcal{C}_m}^3(\mathbf{B}, \mathbf{R})} (b_{kk} - b_{mm}) \left((p_k - 1)(r_{km} - r_{kk}) \right. \\
&\quad \left. + (p_m - 1)(r_{km} - r_{mm}) \right) \\
\frac{\partial^2 L_{\text{clust}}(b_{kk}, \mathbf{r}_k, r_{kk})}{\partial b_{kk} \partial r_{kk}} &= -\lambda (p_k - 1) \sum_{\substack{\ell=1 \\ \ell \neq k}}^K \frac{\mathbf{u}_k^\top \mathbf{W} \mathbf{u}_\ell}{d_{\mathcal{C}_k \mathcal{C}_\ell}^3(\mathbf{B}, \mathbf{R})} (b_{kk} - b_{\ell\ell}) (r_{kk} - r_{k\ell}) \\
\frac{\partial^2 L_{\text{clust}}(b_{kk}, \mathbf{r}_k, r_{kk})}{\partial r_{km}^2} &= \lambda \sum_{\substack{\ell=1 \\ \ell \notin \{k, m\}}}^K p_m \frac{\mathbf{u}_k^\top \mathbf{W} \mathbf{u}_\ell}{d_{\mathcal{C}_k \mathcal{C}_\ell}(\mathbf{B}, \mathbf{R})} \left(1 - \frac{p_m (r_{km} - r_{m\ell})^2}{d_{\mathcal{C}_k \mathcal{C}_\ell}^2(\mathbf{B}, \mathbf{R})} \right) \\
&\quad + \lambda \sum_{\substack{\ell=1 \\ \ell \notin \{k, m\}}}^K p_k \frac{\mathbf{u}_m^\top \mathbf{W} \mathbf{u}_\ell}{d_{\mathcal{C}_m \mathcal{C}_\ell}(\mathbf{B}, \mathbf{R})} \left(1 - \frac{p_k (r_{km} - r_{k\ell})^2}{d_{\mathcal{C}_m \mathcal{C}_\ell}^2(\mathbf{B}, \mathbf{R})} \right) \\
&\quad + \lambda \frac{\mathbf{u}_k^\top \mathbf{W} \mathbf{u}_m}{d_{\mathcal{C}_k \mathcal{C}_m}(\mathbf{B}, \mathbf{R})} \left(p_k + p_m - 2 \right. \\
&\quad \left. - \frac{((p_k - 1)(r_{km} - r_{kk}) + (p_m - 1)(r_{km} - r_{mm}))^2}{d_{\mathcal{C}_k \mathcal{C}_m}^2(\mathbf{B}, \mathbf{R})} \right) \\
\frac{\partial^2 L_{\text{clust}}(b_{kk}, \mathbf{r}_k, r_{kk})}{\partial r_{km} \partial r_{km'}} &= -\lambda \sum_{\substack{\ell=1 \\ \ell \notin \{k, m, m'\}}}^K \frac{\mathbf{u}_k^\top \mathbf{W} \mathbf{u}_\ell}{d_{\mathcal{C}_k \mathcal{C}_\ell}^3(\mathbf{B}, \mathbf{R})} p_m p_{m'} (r_{km} - r_{m\ell}) (r_{km'} - r_{m'\ell}) \\
&\quad - \lambda \frac{\mathbf{u}_k^\top \mathbf{W} \mathbf{u}_{m'}}{d_{\mathcal{C}_k \mathcal{C}_{m'}}^3(\mathbf{B}, \mathbf{R})} \left((p_k - 1)(r_{km'} - r_{kk}) \right. \\
&\quad \left. + (p_m - 1)(r_{km'} - r_{m'm'}) \right) p_m (r_{km} - r_{mm'}) \\
&\quad + \lambda p_k \frac{\mathbf{u}_m^\top \mathbf{W} \mathbf{u}_{m'}}{d_{\mathcal{C}_m \mathcal{C}_{m'}}(\mathbf{B}, \mathbf{R})} \left(1 - \frac{p_k (r_{km'} - r_{km})^2}{d_{\mathcal{C}_m \mathcal{C}_{m'}}^2(\mathbf{B}, \mathbf{R})} \right) \\
&\quad - \lambda \frac{\mathbf{u}_k^\top \mathbf{W} \mathbf{u}_m}{d_{\mathcal{C}_k \mathcal{C}_m}(\mathbf{B}, \mathbf{R})} p_{m'} (r_{km'} - r_{mm'}) \left((p_k - 1)(r_{km} - r_{kk}) \right. \\
&\quad \left. + (p_m - 1)(r_{km} - r_{mm}) \right) \\
\frac{\partial^2 L_{\text{clust}}(b_{kk}, \mathbf{r}_k, r_{kk})}{\partial r_{km} \partial r_{kk}} &= -\lambda \sum_{\substack{\ell=1 \\ \ell \notin \{k, m\}}}^K \frac{\mathbf{u}_k^\top \mathbf{W} \mathbf{u}_\ell}{d_{\mathcal{C}_k \mathcal{C}_\ell}^3(\mathbf{B}, \mathbf{R})} p_m (p_k - 1) (r_{kk} - r_{k\ell}) (r_{km} - r_{m\ell}) \\
&\quad - \lambda (p_k - 1) \frac{\mathbf{u}_k^\top \mathbf{W} \mathbf{u}_m}{d_{\mathcal{C}_k \mathcal{C}_m}(\mathbf{B}, \mathbf{R})} \left(1 - \frac{(p_k - 1)(r_{km} - r_{kk})}{d_{\mathcal{C}_k \mathcal{C}_m}^2(\mathbf{B}, \mathbf{R})} (r_{km} - r_{kk}) \right)
\end{aligned}$$

$$\begin{aligned}
& + \frac{(p_m - 1)(r_{km} - r_{mm})}{d_{\mathcal{C}_k \mathcal{C}_m}^2(\mathbf{B}, \mathbf{R})} (r_{km} - r_{kk}) \Big) \\
\frac{\partial^2 L_{\text{clust}}(b_{kk}, \mathbf{r}_k, r_{kk})}{\partial r_{kk}^2} &= \lambda(p_k - 1) \sum_{\substack{\ell=1 \\ \ell \neq k}}^K \mathbf{u}_k^\top \mathbf{W} \mathbf{u}_\ell \left(\frac{1}{d_{\mathcal{C}_k \mathcal{C}_\ell}(\mathbf{B}, \mathbf{R})} - \frac{(p_k - 1)(r_{kk} - r_{k\ell})^2}{d_{\mathcal{C}_k \mathcal{C}_\ell}^3(\mathbf{B}, \mathbf{R})} \right) \\
\frac{\partial^2 L_{\text{sparse}}(b_{kk}, \mathbf{r}_k, r_{kk})}{\partial r_{k\ell}^2} &= \begin{cases} \frac{\mathbf{u}_k \mathbf{Z} \mathbf{u}_\ell^\top}{\varepsilon} & \text{if } |r_{k\ell}| < \varepsilon, \\ 0 & \text{otherwise.} \end{cases}
\end{aligned}$$

where the terms relating to $L_{\text{cov}}(b_{kk}, \mathbf{r}_k, r_{kk})$ are left out as they evaluate to zero. It should be noted that, due to the parameterization of a_{kk} as $b_{kk} - r_{kk}$, if the size of the k^{th} cluster is one, the parameter r_{kk} is effectively not a part of the objective function. Consequently, the k^{th} element of the gradient is zero and the k^{th} row and column of the Hessian are also filled with zeros.

B.3 Computing a Clusterpath

A primary objective of CGGM is to construct a clusterpath, ranging from p to a few clusters. This requires an appropriate sequence for λ_c , which determines the strength of the aggregation penalty. However, it is not known a priori which value for λ_c corresponds to which number of clusters. To aid finding such a sequence, we propose rescaling λ_c to reduce its sensitivity to the data and automating the computation of the clusterpath.

First, note that in the current form of objective function (8), the tuning parameter λ_c is sensitive to the number of variables p . To reduce this sensitivity, the different terms of the objective function may be scaled. We suggest to scale the terms pertaining to the log likelihood and sparsity penalty by p^{-1} and the clusterpath penalty by $\kappa = ((p - 1)^{1/2} \sum_{j < j'} w_{jj'})^{-1}$ to reduce this dependence on p . These scaling factors can then easily be absorbed into a rescaled tuning parameter for the clusterpath penalty $\gamma_c = p\kappa\lambda_c$, which then replaces λ_c in the objective function.

To set a sequence for λ_c , we implement an automated procedure. This procedure consists of two stages: a rough stage and a refinement stage. First, the goal is to find the value for λ_c as of which the minimum number of clusters is attained. Typically, this is one cluster, but it may be larger if the weight matrix used in the clusterpath penalty is very sparse and contains groups of variables that are not connected through nonzero weights. We initialize λ_c at 0.5 and iteratively increment it by 50% until the minimum number of clusters is reached. The initial choice $\lambda_c = 0.5$ hereby strikes a balance between the speed at which the maximum value for λ_c is found and the level of detail of the initial clusterpath. This yields an increasing series of values $\{\lambda_c^{(q)} : 1 \leq q \leq Q\}$ alongside their corresponding solutions $\{\hat{\Theta}^{(q)} : 1 \leq q \leq Q\}$.

In the refinement stage, the goal is to obtain a smooth clusterpath. To ensure smoothness in its trajectory, additional values of λ_c are inserted whenever the difference between consecutive solutions, as measured by $\|\hat{\Theta}^{(q-1)} - \hat{\Theta}^{(q)}\| / \|\hat{\Theta}^{(q-1)}\|$, exceeds 0.01. Consequently, a continuum of solutions for Θ is obtained, transitioning smoothly from minimal to maximal regularization, with a hierarchical clustering structure. Throughout this iterative process, the algorithm leverages existing solutions for the precision matrix as warm starts to speed up finding solutions for new values of λ_c .

C Additional Simulation Results

This appendix provides supplementary simulation results from the experiments discussed in the main paper. Figures 9 and 10 illustrate the results for the chain simulation design with varying numbers of variables and clusters, respectively. Figure 11 shows the results for designs featuring an approximate block structure. The performance of the methods is evaluated based on estimation accuracy (Frobenius norm), clustering quality (number of clusters and ARI), and sparsity recognition (FPR and FNR).

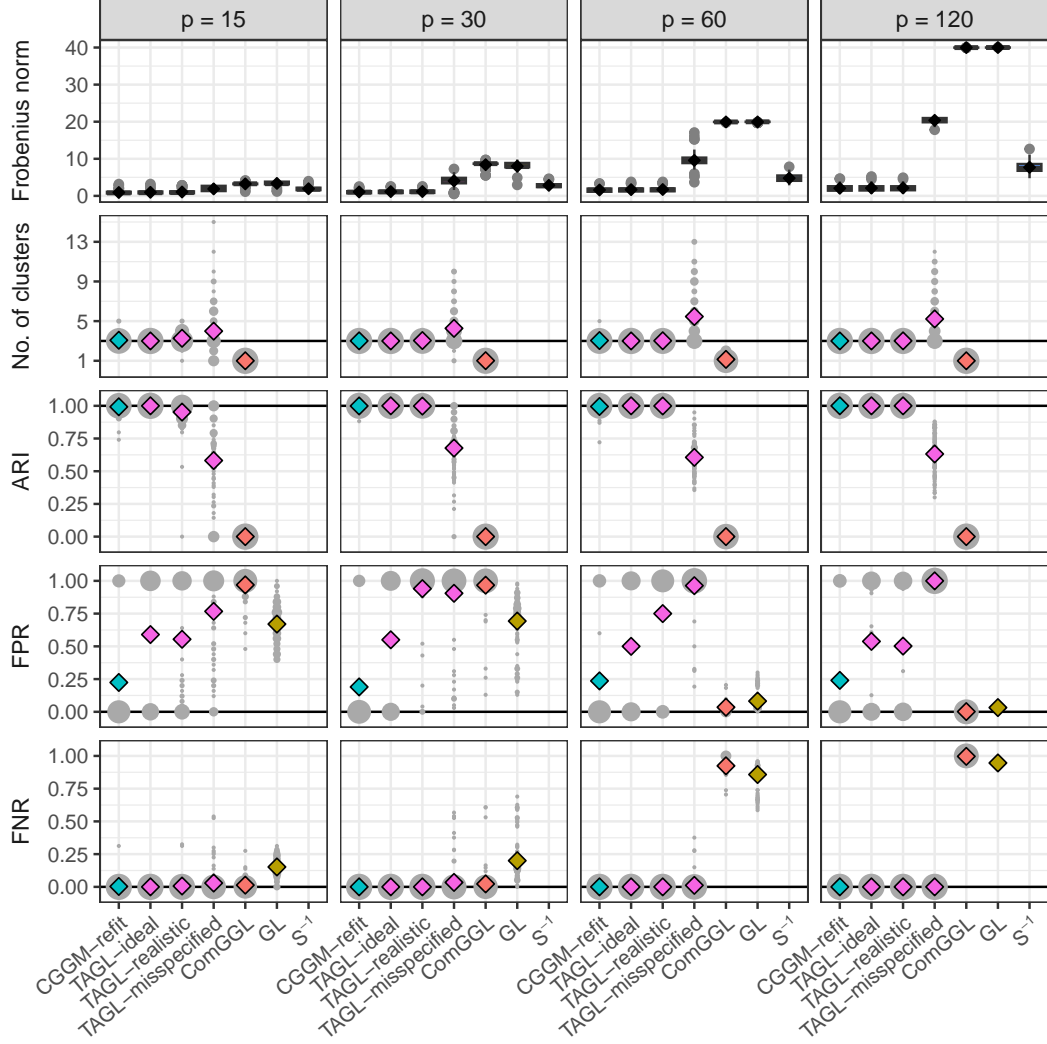


Figure 9: Results for the chain simulation design with an increasing number of variables (columns). Top row: Boxplots of the Frobenius norm with black diamonds representing the average. Other rows: Diamonds display the average of the estimated number of clusters, ARI, FPR, and FNR. Reference lines are added for the true number of clusters, the ARI value of perfect clustering, and the FPR and FNR of perfect sparsity recognition, respectively. The size of the gray dots represent the frequency of different values across the replications. Aggregation performance is not applicable and omitted for GL and \mathbf{S}^{-1} , as is sparsity recognition performance for \mathbf{S}^{-1} .

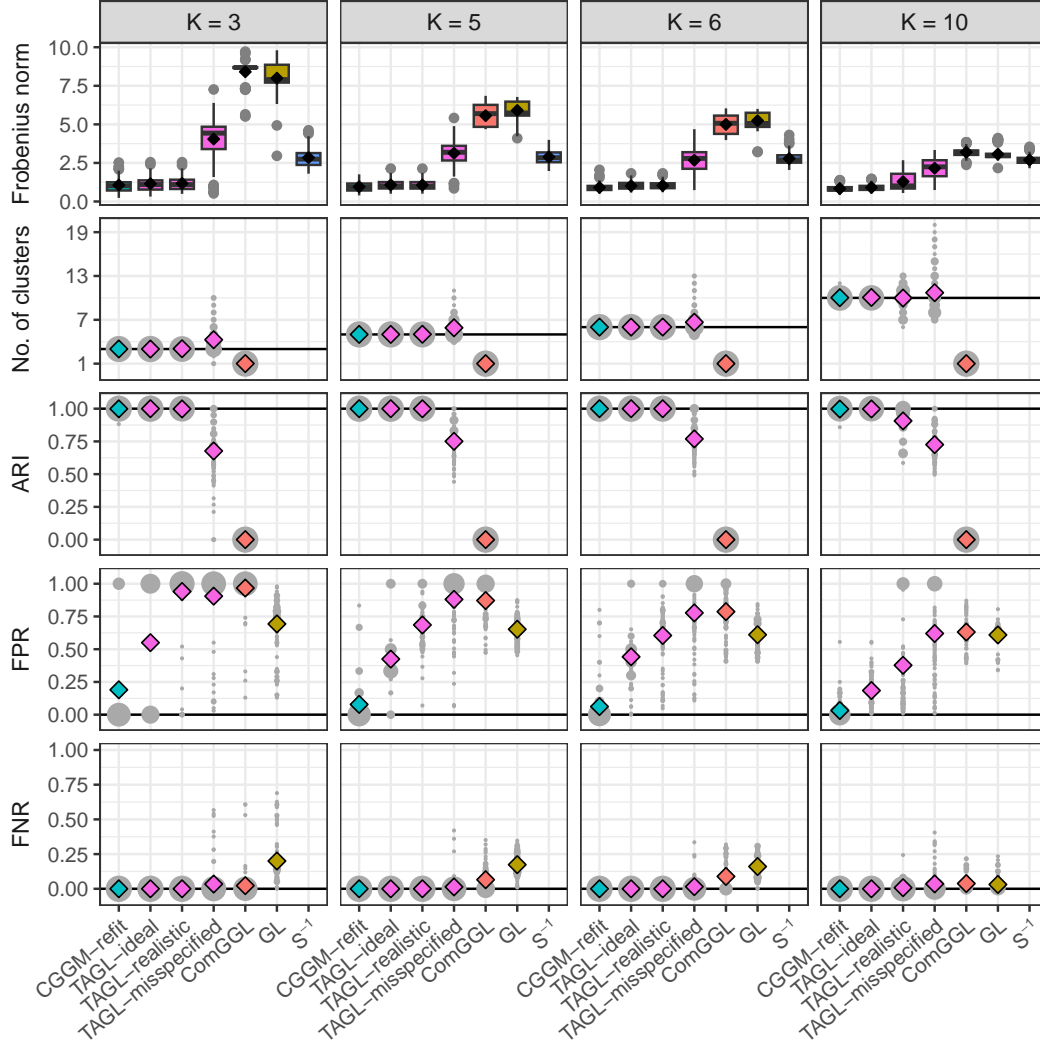


Figure 10: Results for the chain simulation design with an increasing number of clusters (columns). See Figure 9 for explanatory notes.

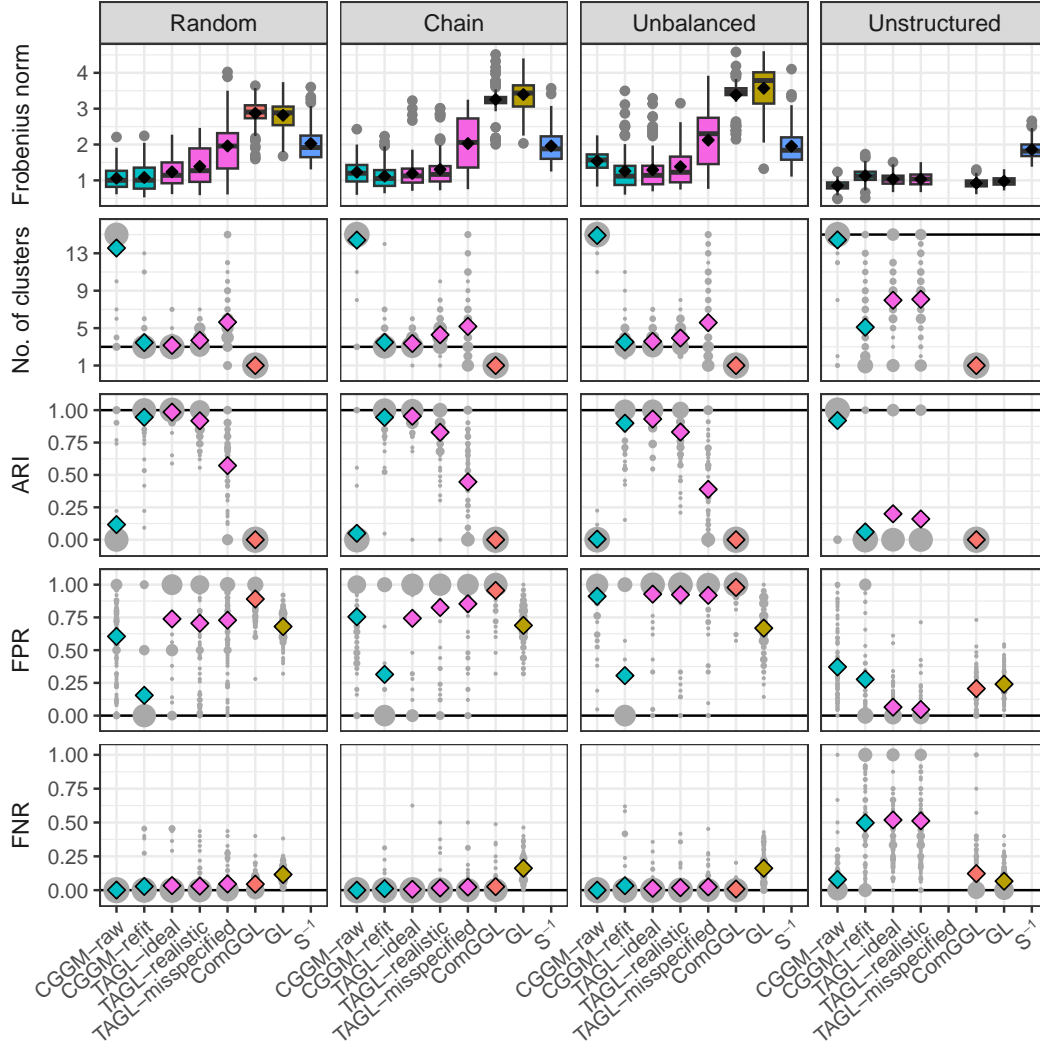


Figure 11: Results for the simulation designs with approximate block structure (columns). See Figure 9 for explanatory notes. In the unstructured design, a misspecified tree for TAGL does not exist since any tree hierarchy contains the true clustering (each variable being its own cluster).

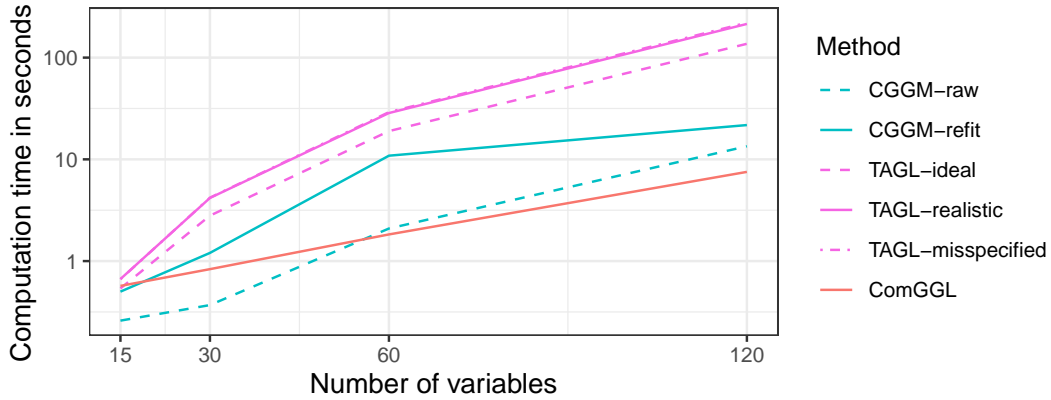


Figure 12: Total computation time in seconds across a grid of values for the corresponding aggregation parameter, averaged over 10 replications of the chain simulation design with an increasing number of variables.

D Additional Details on Applications

We provide additional details on the three empirical applications in Appendices D.1–D.3.

D.1 S&P 100 Stocks

We collect daily stock price information for the period ranging from January 3rd, 2023, until December 29th, 2023 ($n = 250$), and compute daily realized ranges as given by

$$r_{tj} = \frac{(\log H_{tj} - \log L_{tj})^2}{4 \log 2},$$

where H_{tj} and L_{tj} are the high and low prices for stock j during trading day t (Parkinson, 1980) to study the conditional dependency structure of the stocks’ realized ranges.

As a preprocessing step, we first fit the popular heterogeneous autoregressive (HAR) model of Corsi (2009) to the individual daily realized range series to capture time dependencies. The HAR model captures the temporal dependence in the daily realized range in a very simple yet parsimonious way, namely by explaining it through a weighted average based on the realized range of the preceding day, week and month. After estimating the HAR models, we obtain the $p = 101$ standardized residual series and then apply CGGM and TAGL to these to learn the conditional dependency structure among the stocks; our procedure is in line with Wilms & Bien (2022). Note that estimating a graphical model to these residuals series is useful in the context of a multivariate time series analysis to capture the contemporaneous relationships among the $p = 101$ realized ranges.

For both CGGM and TAGL, we use 5-fold cross-validation to select the tuning parameters (k , ϕ , λ_c , and λ_s for CGGM; aggregation and sparsity parameters for TAGL) and refit the precision matrix subject to the obtained variable clustering and sparsity structure. For the clustering weight matrix in CGGM, we use a grid of $k \in \{3, 6, 9, 12, 15\}$ and $\phi \in \{5, 15, 25\}$, based on a visual inspection of the distributions of the resulting nonzero weights (see Figure 13). Clearly, increasing ϕ results in a broader distribution of the weights across the interval $(0, 1]$, but all candidate values avoid that the distribution is mostly a singular spike around a specific value. Candidate values for the aggregation and sparsity parameters of CGGM and TAGL are determined via the same procedure as in the simulations from the main text.

To investigate the stability of the tuning parameter selection in CGGM, Figure 14 visualizes the cross-validation scores. Since there are four tuning parameters, the optimal combination is used as a baseline (highlighted by vertical reference lines), with separate panels displaying the score as we vary each tuning parameter. Except for some wiggles in the plot for λ_c and λ_s , the cross-validation score seems to be smooth with respect to the tuning parameters, offering reassurance in the stability of the results.

D.2 OECD Well-Being Indicators

We analyze the OECD well-being indicator dataset that was previously analyzed in Cavicchia et al. (2022), and we acknowledge the authors for providing us with data access. It contains data on $p = 11$ variables related to well-being: education, jobs, income, safety, health, environment, civic engagement, accessibility to services, housing, community, and life satisfaction on which $n = 36$ countries are given a score ranging from 0 to 10.

Note, however, that we split the sample of countries into two groups since Cavicchia et al. (2022) have found evidence for two groups of countries ($n_1 = 21$ and $n_2 = 14$) with distinct clustering structures. The first group consists of the countries Australia, Austria, Belgium, Canada, Denmark, Finland, France, Germany, Iceland, Ireland, Italy, Japan, Luxembourg, Netherlands, New Zealand, Norway, Spain, Sweden, Switzerland, United Kingdom, United States, the second group consists of the countries Chile, Czech Republic, Estonia, Greece, Hungary, Israel, Korea, Latvia, Lithuania, Mexico, Poland, Portugal, Slovak Republic, Slovenia, Turkey. In our analysis, we omit Lithuania from the second group due to a missing value, hence $n_1 = 21$ and $n_2 = 14$.

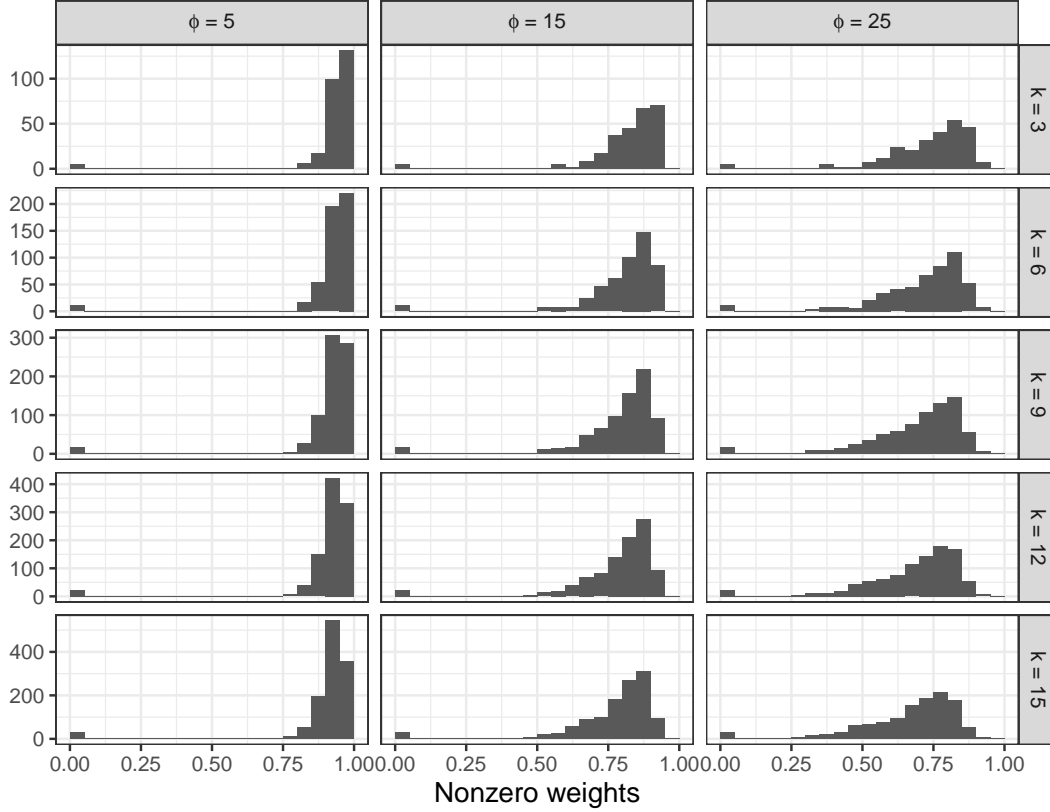


Figure 13: Histogram of nonzero clustering weights in CGGM for different candidate values of the tuning parameters k (in rows) and ϕ (in columns).

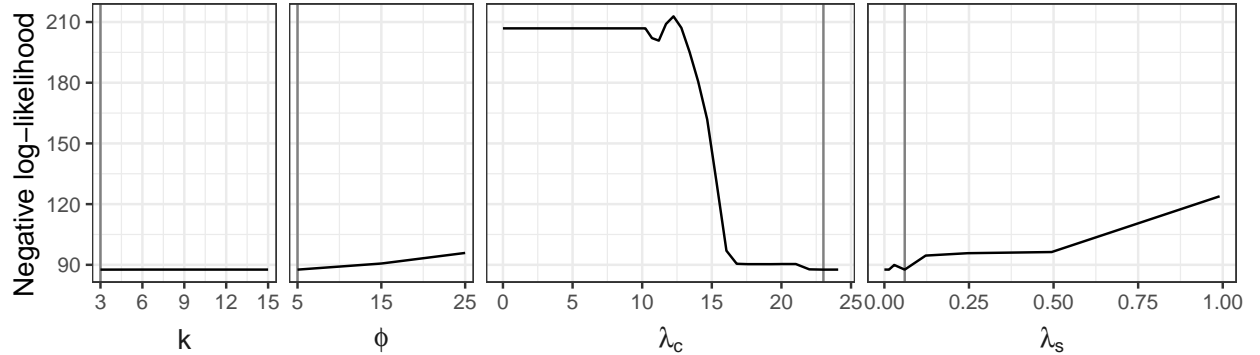


Figure 14: Cross-validation score (negative log-likelihood) using the optimal combination of tuning parameters as a baseline. Separate panels visualize the score as one of the tuning parameter varies, with vertical reference lines highlighting the respective optimal values.

D.3 Humor Styles Questionnaire

We analyze data on the humor styles questionnaire (HSQ) developed by Martin et al. (2003). The $p = 32$ items of the HSQ, grouped by the humor style being measured, are listed in Table 2 together with their position in the survey. We use responses to the HSQ on a five-point rating scale (anchored by 1 = “never or very rarely true” to 5 = “very often or always true”), which we obtained from https://openpsychometrics.org/_rawdata/. In addition to the aforementioned $p = 32$ items, participants were asked at the end of the questionnaire to indicate the accuracy of their responses. We restrict our analysis to participants who

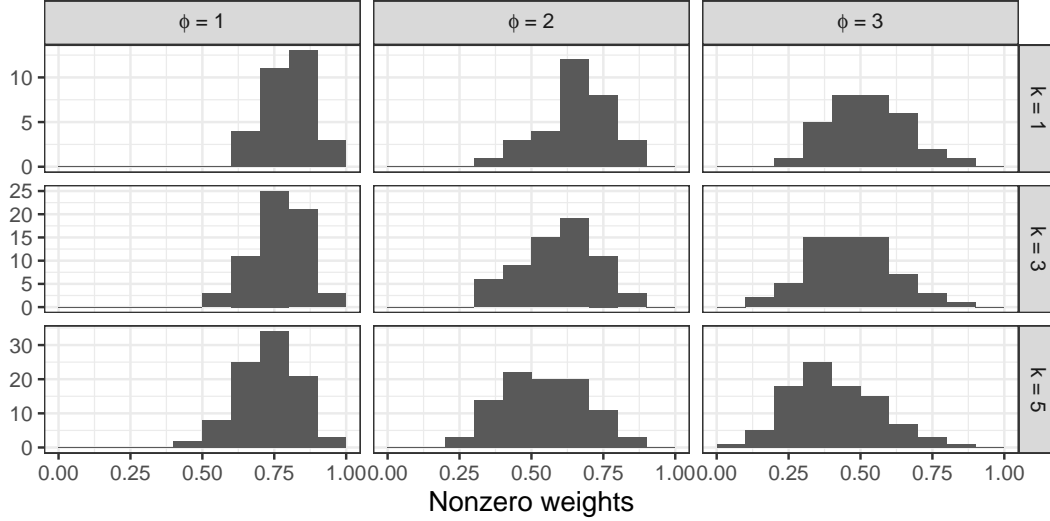


Figure 15: Histogram of nonzero clustering weights in CGGM for different candidate values of the tuning parameters k (in rows) and ϕ (in columns).

reported that their responses are fully accurate, and after removing 9 observations with missing responses, we retain $n = 182$ respondents.

We apply the CGGM algorithm for clustering the covariance matrix (including the refitting step). It is worth pointing out that the objective function of CGGM is based on the likelihood under the assumption of a normal distribution, which is clearly violated here due to the discrete nature of the data. Nevertheless, it is a common assumption in the behavioral sciences that such rating-scale data are discrete measurements of latent normally-distributed sentiments. Hence, the assumed normal distribution in CGGM may be viewed as a (crude) approximation. Moreover, it is interesting to see whether we can obtain meaningful results even if this normality assumption is violated.

Moreover, we use 5-fold cross-validation to determine the optimal values of the tuning parameters k , ϕ , λ_c , and λ_s . We thereby use the same candidate values as in the simulations from Section 4 of the main text. Figure 15 reveals that the choice of candidate values $k \in \{1, 3, 5\}$ and $\phi \in \{1, 2, 3\}$ is reasonable due to the amount of variation among the nonzero clustering weights, particularly for $\phi = 3$. In fact, we first simply set $\phi = 3$ after inspecting this plot (only tuning the remaining hyperparameters), and only afterward included $\phi \in \{1, 2, 3\}$ in the cross-validation as a robustness check. We obtained the same results in both cases.

Finally, we again investigate the stability of the tuning parameter selection by visualizing the cross-validation scores in Figure 16. Although the plot for λ_c , and to a lesser extent that of λ_s show some wiggles, they indicate a clear overall trend through the range of the tuning parameters. Hence, we can be confident that the found optimal values are in a good neighborhood of the tuning parameter space.

Humor style	Items
Affiliative	<ol style="list-style-type: none"> 1. I usually don't laugh or joke around much with other people.* 5. I don't have to work very hard at making other people laugh—I seem to be a naturally humorous person. 9. I rarely make other people laugh by telling funny stories about myself.* 13. I laugh and joke a lot with my closest friends.

Table 2: List of the $p = 32$ items of the humor styles questionnaire of Martin et al. (2003), grouped by the humor style being measured and reported together with their position in the survey. Items marked with an asterisk are reverse-keyed, i.e., the responses on the five-point rating scale are reversed prior to analysis.

Table continues on the next page

Humor style	Items
Self-enhancing	17. I usually don't like to tell jokes or amuse people.*
	21. I enjoy making people laugh.
	25. I don't often joke around with my friends.*
	29. I usually can't think of witty things to say when I'm with other people.*
	2. If I am feeling depressed, I can usually cheer myself up with humor.
	6. Even when I'm by myself, I'm often amused by the absurdities of life.
	10. If I am feeling upset or unhappy I usually try to think of something funny about the situation to make myself feel better.
	14. My humorous outlook on life keeps me from getting overly upset or depressed about things.
	18. If I'm by myself and I'm feeling unhappy, I make an effort to think of something funny to cheer myself up.
	22. If I am feeling sad or upset, I usually lose my sense of humor.*
Aggressive	26. It is my experience that thinking about some amusing aspect of a situation is often a very effective way of coping with problems.
	30. I don't need to be with other people to feel amused—I can usually find things to laugh about even when I'm by myself.
	3. If someone makes a mistake, I will often tease them about it.
	7. People are never offended or hurt by my sense of humor.*
	11. When telling jokes or saying funny things, I am usually not very concerned about how other people are taking it.
	15. I do not like it when people use humor as a way of criticizing or putting someone down.*
	19. Sometimes I think of something that is so funny that I can't stop myself from saying it, even if it is not appropriate for the situation.
	23. I never participate in laughing at others even if all my friends are doing it.*
Self-defeating	27. If I don't like someone, I often use humor or teasing to put them down.
	31. Even if something is really funny to me, I will not laugh or joke about it if someone will be offended.*
	4. I let people laugh at me or make fun at my expense more than I should.
	8. I will often get carried away in putting myself down if it makes my family or friends laugh.
	12. I often try to make people like or accept me more by saying something funny about my own weaknesses, blunders, or faults.
	16. I don't often say funny things to put myself down.*
	20. I often go overboard in putting myself down when I am making jokes or trying to be funny.
	24. When I am with friends or family, I often seem to be the one that other people make fun of or joke about.
	28. If I am having problems or feeling unhappy, I often cover it up by joking around, so that even my closest friends don't know how I really feel.
	32. Letting others laugh at me is my way of keeping my friends and family in good spirits.

Table 2: List of the $p = 32$ items of the humor styles questionnaire of Martin et al. (2003), grouped by the humor style being measured and reported together with their position in the survey. Items marked with an asterisk are reverse-keyed, i.e., the responses on the five-point rating scale are reversed prior to analysis.

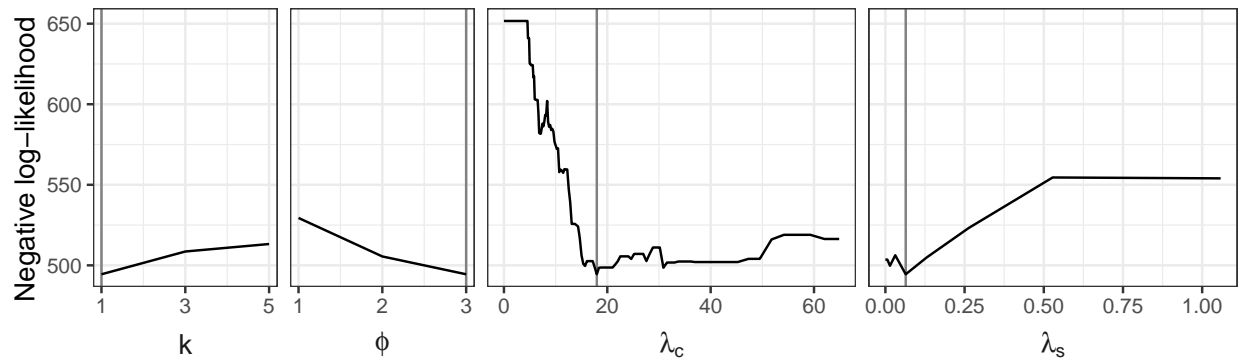


Figure 16: Cross-validation score (negative log-likelihood) using the optimal combination of tuning parameters as a baseline. Separate panels visualize the score as one of the tuning parameter varies, with vertical reference lines highlighting the respective optimal values.