Università degli Studi di Padova
Dipartimento di Scienze Statistiche

Corso di Laurea Magistrale in
Scienze Statistiche

Study of across-graph association in Gaussian graphical models for paired data

Relatore Prof.ssa Manuela Cattelan
Dipartimento di Scienze Statistiche

Laureando Francesco Bonato
Matricola N 2089736

Anno Accademico 2023/2024

Contents

Introduction

The subject of this work will be the family of coloured graphical models, introduced by Højsgaard & Lauritzen (2008), which are graphical models characterized by the presence of symmetries between edges and vertices. A particular interest will be given to a subfamily of coloured graphical models, called pdRCON models (Ranciati & Roverato, 2024a), used in the paired data setting, meaning that the graph is comprised of two groups that are dependent of each other and present similarities. This dependence structure in the data needs to be accounted for in the learning process. Special attention will be brought to the role played by the hypotheses on the across-graph dependence structure in the inference on the two group-level subgraphs.

The first chapter will give a brief introduction to the family of *Gaussian Graphical Models* and their notation, and the general graphical lasso (Friedman et al., 2008) will be presented.

The second chapter will focus on the family of coloured Gaussian graphical models, and some examples of model selection procedures in the recent literature will be explained. The symmetric graphical lasso by Ranciati et al. (2021) will be introduced, along with its extension given by the graphical lasso for paired data (Ranciati & Roverato, 2024a), which gives a solution to the problem of model selection within the paired data context through a fused-type lasso penalized log-loglikelihood approach.

The third chapter will show the results of a simulation study, making use of the R package `pdglasso` (Ranciati & Roverato, 2024b), investigating whether a model reduction obtained by assuming that all across-group edges are either absent or symmetric leads to improved structure and symmetry recovery of the two subgraphs, as well as whether parameter estimation is more accurate.

The fourth chapter will present an instance of a real-world paired data problem, where the variables are genes from a breast cancer tissue and healthy adjacent tissue, and interest is in estimating the similarities and differences of the conditional dependence structures of the the gene-level transcription estimates of the cancerous and healthy samples; the coloured graphs selected within the submodel classes considered in Chapter three will be compared.

Chapter 1

Gaussian Graphical Models

The objective of this chapter is to give some background on Gaussian graphical models (or GGMs) and the graphical lasso estimation approach, while establishing the notation. A complete overview of graphical modelling can be found in Lauritzen (1996).

1.1 Graphs

A graph G is defined as the pair $G = (V, E)$, where

- $V = \{1, 2, ..., p\}$ is the set of vertices, sometimes also called nodes or points;
- $E \subseteq V \times V$ is the set of the edges, sometimes also called links or arcs, representing relationships between pairs of edges.

There is a distinction between *directed* graphs, in which case E is a set of ordered pairs of vertices and edges are usually represented graphically via the use of arrows, and *undirected* graphs, in which case E is a set of unordered pairs of vertices; this work will be focused on this second instance of graphical models. Directed graphs are used for example in the fields of probabilistic reasoning and causal inference in the form of *Directed Acyclic Graphs*, or DAGs, sometimes reffered to also as *Bayesian networks*; see e.g. Koski & Noble (2009).

In an undirected graph, the absence of an edge between two vertices implies that the *partial correlation* between the two corresponding variables given the rest of the graph is zero. That is, once the linear dependence on all other variables in the graph is removed by fitting a regression model of each of the two variables on all others, the residuals of these two regressions are uncorrelated. If joint normality of the variables in the graph is assumed, which will be the case with Gaussian Graphical Models, then lack of correlation is equivalent to independence, and absence of an edge implies *conditional independence* between the two corresponding variables given the rest of the graph. In the graph of Figure 1.1 the absence of an edge between A and H, for example, implies that the two variables are conditionally independent given the rest of the graph, that is, $A \perp H | V \setminus \{A, H\} \iff A \perp H | \{B, C, D, E, F, G\}$.

A different type of graphical model was introduced by Cox & Wermuth (1993), known as the *covariance graph model*, in which the graph restricts the marginal independence structure of the variables, rather than the conditional one; these models are however analytically more complex and inferential procedures are more challenging.

1.1.1 Clique factorization

A graph *clique* is a subset of the vertex set $C \subseteq V$ that is *fully connected*: that is, $(s, t) \in E$ for all $s, t \in C$. A clique is *maximal* if it is not strictly contained in any other clique or, in other words, a clique for which the addition of any further vertices

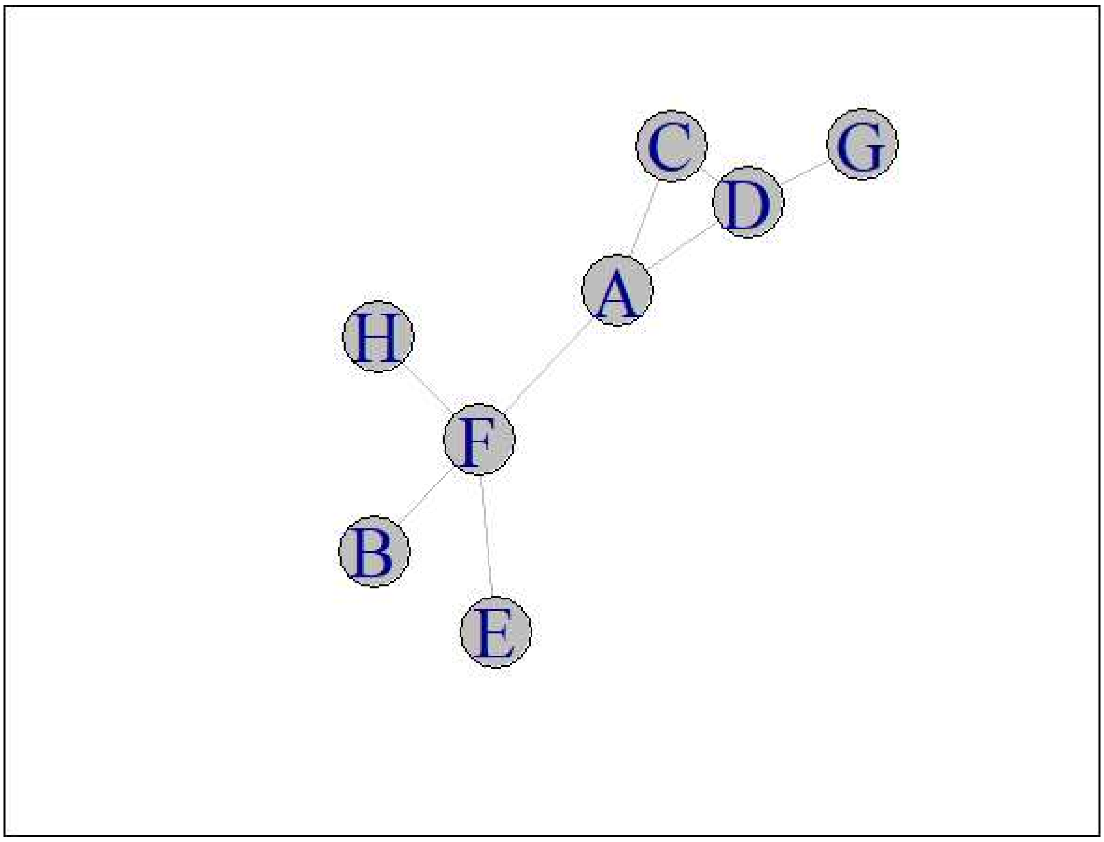


FIGURE 1.1: Example of an undirected graph having having vertex set $V = \{A, B, C, D, E, F, G, H\}$ and edge set $E\{(A, C), (A, D), (C, D), (A, F), (B, F), (E, F), (D, G), (F, H)\} \subseteq V \times V$.

to the clique would make the subset not fully connected anymore. Figure 1.2 illustrates the maximal cliques of the undirected graph of Figure 1.1. The set of all cliques in the graph, both maximal and non-maximal, is denoted by $\mathcal{C}$.

A collection of random variables $X = (X_1, ..., X_p)$, each taking values in the space $\mathcal{X}$ can be associated with the graph G, thus constraining the distribution of the random vector X via the clique structure of G.

For a given clique $C \in \mathcal{C}$ a *compatibility function* is a real-valued function ψ_C of the sub-vector $x_C = (x_s, s \in C)$ taking positive real values. The probability distribution $\mathbb{P}$ of the random vector X *factorizes* over the graph G if the decomposition

$$\mathbb{P}(x_1, ..., x_p) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi_C(x_C) \tag{1.1}$$

holds true, where $Z = \sum_{x \in \mathcal{X}^p} \prod_{C \in \mathcal{C}} \psi_C(x_C)$ is the *partition function*, a normalising constant ensuring that the quantity is a valid probability distribution, with total probability summing to one.

Such factorization makes evident the usefulness of graph models. Once a graph G fitting to the random vector X is identified, it is possible to redefine the probability distribution $\mathbb{P}$, taking values in the space $\mathcal{X}^p = X \times X \times ... \times X$, that would then require to specify a potentially large number of parameters, into a much easier to handle and compact product of compatibility function. This factorization is thus particularly convenient in terms of storage and computational resources when p is large and the clique sizes are not too big. Graphical models are then a useful framework for building parsimonious models in high-dimensionality settings.

The structure of the graph is in general not known beforehand, and will need to be learned from the data. This is a process called *graphical model selection*; more about this will be presented later.

1.1.2 Markov property

Another possible way to look at how the graph constrains the distribution of the random vector X is by considering its *cut set* structure.

A cut set is a subset of vertices $S \subset V$ such that when every node in S is removed, the graph is separated into two fully disconnected subgraphs, having vertex sets denoted here by $A, B \subset V$. An example of a cut set is given in Figure 1.3.

The random vector X is said to be *Markov with respect to* the graph G if

$$X_A \perp X_B \mid X_S \quad \text{for all cut sets} \quad S \subset V\ . \tag{1.2}$$

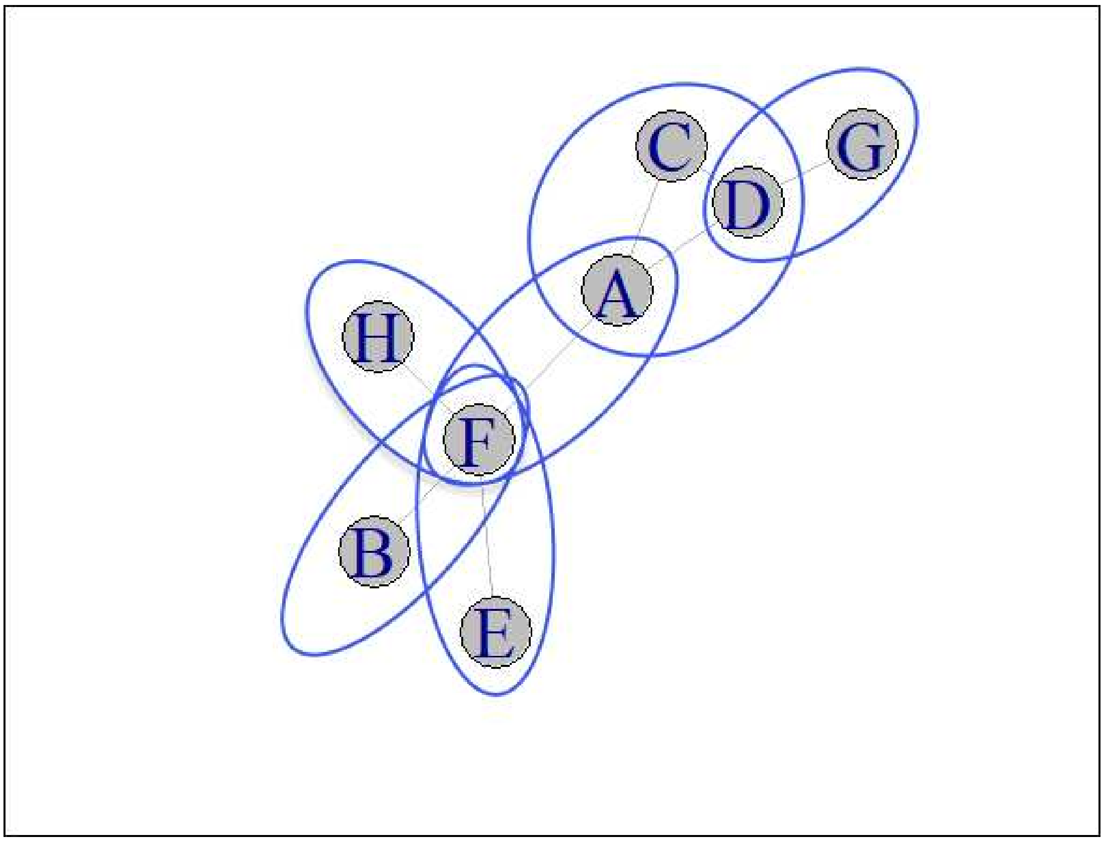


FIGURE 1.2: Maximal cliques of the undirected graph of Figure 1.1. The maximal cliques are $\{B, F\}, \{E, F\}, \{F, H\}, \{A, F\}, \{A, C, D\}, \{D, G\}$

As an example, one can think of a Markov chain: it can be considered a graph with a chain structure, with edge set $E = \{(1, 2), (2, 3), ..., (p-1, p)\}$. Any single vertex $s \in \{2, 3, .., p-1\}$ forms then a cut set, separating the graph into the past $P = \{1, ..., s-1\}$ and the future $F = \{s+1, ..., p\}$. The Markov property in this context states that the future X_F is conditionally independent of the past X_P given the present X_s.

1.1.3 Hammersley-Clifford theorem

Remarkably, these two different characterizations are equivalent. This was proved by Clifford & Hammersley (1971) in the so called Hammersley-Clifford theorem, known also as the *fundamental theorem of Markov fields*, which states that for any strictly positive distribution, that is, $\mathbb{P}$ for which $\mathbb{P}(x) > 0$ for all $x \in \mathcal{X}^p$, the distribution of the random vector X factorizes according to the graph G, qquation (1.1), if and only if X is Markov with respect to the graph G, as in Equation (1.2). An easier proof of the theorem is provided by Grimmett (1973).

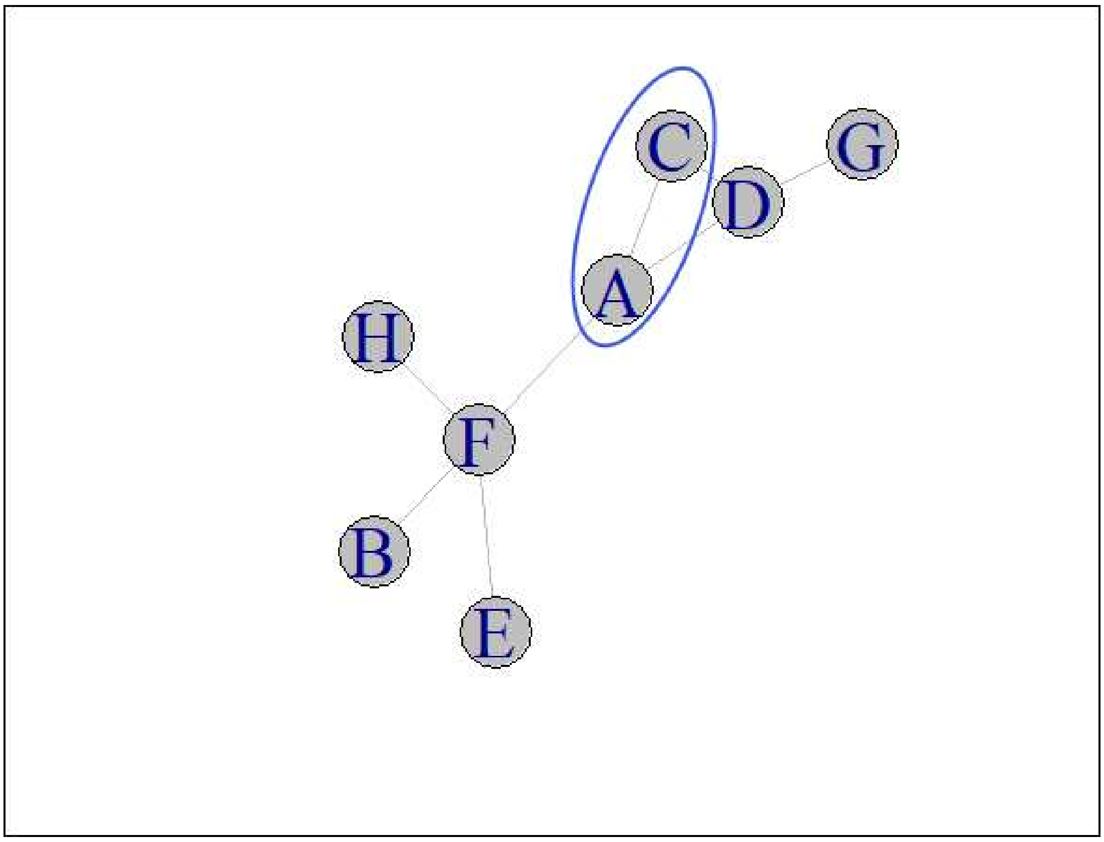


FIGURE 1.3: Example of a cut set of the undirected graph of Figure 1.1. The cut set $\{A, C\}$ separates the graph into two fully disconnected subgraphs, one having vertex set $\{B, E, F, H\}$ and the other having vertex set $\{D, G\}$.

1.2 Gaussian graphical models

In the presence of continuous variables, the Gaussian distribution is almost always assumed, because of its convenient analytical properties. A p-variate random vector X follows a Gaussian distribution in p dimensions with mean vector $\mu \in \mathbb{R}^p$ and variance-covariance matrix Σ ($X \sim N(\mu, \Sigma)$) if its probability distribution can be written in the form

$$\mathbb{P}_{\mu,\Sigma}(x) = \frac{1}{(2\pi)^{\frac{p}{2}} \det(\Sigma)^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)} ,$$

which is part of the exponential family (see e.g. Salvan et al. (2020), Chapter 2). The *mean parametrization* is being used here; in the context of Gaussian graphical models, however, it is more convenient to use the *canonical parametrization* instead. Any non-degenerate multivariate Gaussian distribution, meaning Σ is a strictly positive definite matrix, can be represented in the form

$$\mathbb{P}_{\gamma,\Theta}(x) = \exp\left\{ \sum_{s=1}^{p} \gamma_s x_s - \frac{1}{2} \sum_{s,t=1}^{p} \theta_{st} x_s x_t - A(\Theta) \right\} ,$$

where $\theta_{st} = [\Theta]_{(s,t)}$ and $A = -\frac{1}{2}\log\det(\frac{\Theta}{2\pi})$, thus ensuring that the quantity is a valid probability distribution, i.e. $\int \mathbb{P}_{\gamma,\Theta}(x)dx = 1$.

The rescaling by the $-\frac{1}{2}$ factor allows for the interpretation of Θ as $\Theta = \Sigma^{-1}$, the inverse of the covariance matrix, which is called the *precision* or *concentration* matrix. The convenience of this representation stems from the fact that it allows to discuss the factorization property of the graph in terms of the sparsity pattern of Θ: whenever X factorizes according to the graph G, it must hold that $\theta_{st} = 0$ for any pair $(s,t) \notin E$. There is then a strict correspondence between the zero-pattern of the concentration matrix Θ and the edge structure E of the underlying graph. The elements of the concentration matrix Θ are directly interpretable in terms of the *partial covariances* between variables: θ_{st} is the covariance between variables X_s and X_t, conditioned on all other variables. If $\theta_{st} = 0$, then the two variables are conditionally independent given the rest of the graph, and the corresponding edge is absent.

The diagonal entries of Θ represent the inverse of the *partial variances* of each variable, denoted as $\sigma^2_{ss|V\setminus\{s\}} = \theta^{-1}_{ss}$; the *partial correlation* between X_s and X_t is then given by $\rho_{st|V\setminus\{s,t\}} = -\frac{\theta_{st}}{\sqrt{\theta_{ss}\theta_{tt}}}$ (see Lauritzen (1996), section 5.1.3). Lastly, the regression coefficient of X_s on X_t given $X_{V\setminus\{s,t\}}$ is $\beta_{s\leftarrow t|V\setminus\{s,t\}} = -\frac{\theta_{st}}{\theta_{ss}}$.

1.2.1 Log-likelihood function for GGMs

A common assumption with GGMs is that the data has zero-mean (i.e. $\mu = 0_{p\times 1} = (0,...,0)^T$), so that only the symmetric concentration matrix $\Theta \in \mathbb{R}^{p\times p}$ is needed to identify the model. Given a collection of N independent and identically distributed samples $X = \{x_1,...,x_N\}$ from a graphical model, the log-likelihood function is

$$\begin{aligned}\ell(\Theta;X) &= \frac{2}{N}\sum_{i=1}^{N}\log(\mathbb{P}_\Theta(x_i))\\ &= \frac{2}{N}\sum_{i=1}^{N}\log\left(\frac{\det(\Theta)^{\frac{1}{2}}}{(2\pi)^{\frac{p}{2}}}\exp(-\frac{1}{2}{x_i}^T\Theta x_i)\right)\\ &= \frac{2}{N}\sum_{i=1}^{N}\left(\frac{1}{2}\log\det(\Theta) - \frac{p}{2}\log(2\pi) - \frac{1}{2}{x_i}^T\Theta x_i\right)\\ &= \log\det(\Theta) - \mathrm{tr}(S\Theta)\ , \end{aligned} \tag{1.3}$$

where $S = \frac{1}{N}\sum_{i=1}^{N} x_i {x_i}^T$ is the sample covariance matrix, and $\mathrm{tr}(C)$ denotes the trace, i.e. the sum of the diagonal entries of the square matrix C. If the sample covariance S is non-rank degenerate, and is thus invertible, then the ML estimate for the concentration matrix is $\hat{\Theta}_{\mathrm{MLE}} = S^{-1}$ and it converges to the true concentration matrix Θ as the sample size N tends to infinity. In most scenarios, however, the sample covariance will not be a full rank matrix, so that the MLE fails to exist. In particular, this will be the case whenever $p < N$, which is a common setting in a lot of GGMs application scenarios.

Moreover, the interest is often in obtaining a sparse estimate of the concentration matrix, in which a lot of the off-diagonal entries are estimated exactly to be zero, so that the corresponding graph is more easily interpretable and only the relevant associations between variables are encoded via an edge. Regularization methods for the likelihood function of the GGM are then particularly attractive.

1.3 Graphical lasso

A first approach could be to introduce a regularization that controls for the number of edges itself, measured by the ℓ_0 quantity

$$\rho_0(\Theta) = \sum_{s \neq t} \mathbb{I}[\theta_{st} \neq 0] \ ,$$

where $\mathbb{I}[\cdot]$ is the indicator function. Note that $\rho_0(\Theta) = 2|E|$, where $|E|$ denotes the cardinality, i.e. the number of elements, of the set R. The optimazion problem over the set of all positive definite matrices obtained,

$$\hat{\Theta} \in \operatorname*{arg\,max}_{\rho_0(\Theta) \leq k, \Theta \in \mathcal{S}_+^p} \{\log\det(\Theta) - \operatorname{trace}(S\Theta)\} \ ,$$

where $\mathcal{S}_+^p = \{A \in \mathbb{R}^{p \times p} | A = A^T, A \succeq 0\}$ denotes the set of all $p \times p$ symmetric positive semi-definite matrices, is however highly non convex, as it essentially explores the set of all $\binom{\binom{p}{2}}{k}$ possible subsets of k edges (Hastie et al. (2015), Section 9.3.1), and becomes quickly untractable for large values of p.

It is often best to consider the convex relaxation obtained by replacing the ℓ_0-based constraint with a ℓ_1-based constraint

$$\hat{\Theta} = \operatorname*{arg\,max}_{\Theta \in \mathcal{S}_+^p} \{\log\det(\Theta) - \operatorname{trace}(S\Theta) - \lambda ||\Theta||_1\} \ , \tag{1.4}$$

where $||\Theta||_1 = \sum_{s,t \in V} |\theta_{st}|$ is the ℓ_1-norm of Θ, i.e. the sum of the absolute values of the entries of Θ and λ is the regularization parameter, controlling for the amount of sparsity in the solution. This is an instance of a ℓ_1-regularized log-determinant program, which is a convex optimization problem with a unique optimum for any $\lambda > 0$ (Ravikumar et al., 2011). The advantage to this approach is then two-fold: it both regularizes the estimation process yielding a unique solution, and the estimate of Θ obtained leads to a sparse and more easily interpretable graph.

This optimization problem is referred to as the *graphical lasso*, proposed by Friedman et al. (2008), who provided an extension of the lasso procedure, first introduced in the seminal paper by Tibshirani (1996), to the family of GGMs.

1.3.1 Graphical lasso algorithm

The subgradient equation corresponding to Equation (1.4) is

$$\Theta^{-1} - S - \lambda\Psi = 0 \ , \tag{1.5}$$

where Ψ is a symmetric $p \times p$ matrix having zeros on the diagonal entries and $\psi_{st} = \text{sign}(\theta_{st})$ if $\theta_{st} \neq 0$, else $\psi_{st} \in [-1, 1]$ if $\theta_{st} = 0$ for $s \neq t$.

The problem can be solved with a block coordinate descent approach (Banerjee et al., 2008), partitioning each matrix partitioned into four blocks in this way, with respect here for example to the p-th row and column:

$$\Theta = \begin{pmatrix} \Theta_{11} & \theta_{12} \\ \theta_{12}^T & \theta_{22} \end{pmatrix}, S = \begin{pmatrix} S_{11} & s_{12} \\ s_{12}^T & s_{22} \end{pmatrix}, \Psi = \begin{pmatrix} \Psi_{11} & \psi_{12} \\ \psi_{12}^T & \psi_{22} \end{pmatrix}, W = \begin{pmatrix} W_{11} & w_{12} \\ w_{12}^T & w_{22} \end{pmatrix}, \tag{1.6}$$

where W is the current version of the estimate of Θ^{-1} in this iterative process. If all but the p-th row and column of W is considered to be fixed, Equation (1.5) leads to

$$W_{11}\beta - s_{12} + \lambda\psi_{12} = 0 \tag{1.7}$$

where $\beta = -\frac{\theta_{12}}{\theta_{22}}$.

If the estimate of a lasso regression is considered to be a function of the inner products $X^T X$ and $X^T y$ rather than the raw data X and y itself, then Equation (1.7) can be seen as a modified version of the estimating equations for a lasso regression of the p-th variable on all others, where the input inner products are W_{11} and s_{12}, rather than S_{11} and s_{12}. Equation (1.4) is then not equivalent to p separate lasso regression problems, but to p coupled lasso regression problems sharing the same W (Friedman et al., 2008). The graphical lasso algorithm is presented in Algorithm 1.1.

This is the algorithm as it was originally proposed by Friedman et al. (2008); a slightly different formulation is given in Hastie et al. (2015), Section 9.3.2, where W is simply initialized as $W = S$ so that only the off-diagonal terms of Θ are penalized,

Algorithm 1.1: Graphical Lasso

```
1. Initialize $W = S + \lambda I_p$. Note that the diagonal of $W$ will remain unchanged
   throughout the whole process.
2. For $j = 1, 2, ..., p, 1, 2, ..., p, ...$ until convergence:
   (a) partition the matrices like in Equation (1.6), permuting the rows and
       columns when necessary so that the target column $j$ is the last.
   (b) solve $W_{11}\beta - s_{12} + \lambda\psi_{12} = 0$ using a coordinate descent algorithm (see
       Appendix A), obtaining the $(p-1) \times 1$ vector solution $\hat{\beta}$.
   (c) Update the corresponding row and column of $W$ with $w_{12} = W_{11}\hat{\beta}$
```

solving thus the slightly different optimization problem

$$\hat{\Theta} = \arg\max_{\Theta \in \mathcal{S}_+^p} \{\log\det(\Theta) - \operatorname{trace}(S\Theta) - \lambda\rho_1(\Theta)\} \ ,$$

where $\rho_1(\Theta) = \sum_{s \neq t} |\theta_{st}|$ is the sum of the absolute values of the off-diagonal terms of Θ only.

It is possible to specify edge-specific penalty parameters λ_{st}, so that the optimization problem becomes

$$\hat{\Theta} \in \arg\max_{\Theta \in \mathcal{S}_+^p} \{\log\det(\Theta) - \operatorname{trace}(S\Theta) - \rho_1(\Theta \circ \Lambda)\} \ ,$$

where $\Lambda_{p \times p} = \{\lambda_{st}\}$ with $\lambda_{st} = \lambda_{ts}$ and $\circ$ is the component-wise multiplication, also known as the Hadamard product, taking two matrices of the same dimensions and producing a matrix where each element is the product of the corresponding elements of the input matrices. Note that $\lambda_{st} = \infty$ will force $\hat{\theta}_{st} = 0$ in the solution.

1.3.2 Exploiting the block diagonal structure of the concentration matrix

Witten et al. (2011) point out that if the true concentration matrix Θ has a block diagonal structure, that is, Θ has the form

$$\Theta = \begin{bmatrix} \Theta_{11} & 0 & \cdots & 0 \\ 0 & \Theta_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \cdots \\ 0 & 0 & \cdots & \Theta_{KK} \end{bmatrix} ,$$

then the graphical lasso problem can be solved separately for each block Θ_{kk}, $k \in 1, ..., K$, and the overall solution is constructed from the individual solutions for each block, leading to considerable computational gains. Letting $D_1, D_2, ..., D_k$ denote a partition of the p variables, i.e. $D_k \cap D_{k'} = \emptyset$ for all $k \neq k'$ and $D_1 \cup D_2 \cup ... \cup D_K = \{1, 2, ..., p\}$, so that the elements of each block D_k are fully disconnected from elements of all other blocks, the Authors prove that a necessary and sufficient for the solution of the graphical lasso to be block diagonal is that $|S_{ii'}| < \lambda$ for all $i \in D_k, i' \in D_{k'}, k \neq k'$. Note that in order to achieve the block diagonal structure the order of the rows and columns of Θ can be permuted.

1.3.3 Consistency of the graphical lasso estimator

Rothman et al. (2008) demonstrate that the rate of convergence of the graphical lasso estimator in the Frobenius norm, where the Frobenius norm of a given matrix $A \in \mathbb{R}^{m \times n}$ is $||A||_F = \sqrt{\sum_{i=1}^m \sum_{j=1}^n |a_{ij}|^2}$, as both p and N increase depends on how sparse the true concentration matrix Θ_0 is. The authors prove that, letting $T_{\Theta_0} = \{(i,j) : \Theta_{0ij} \neq 0, i \neq j\}$ denote the true edge set, if $\phi_{\min}(\Sigma_0) > 0$, $\phi_{\max}(\Sigma_0) < \infty$ and λ proportional to $\sqrt{\frac{\log p}{N}}$, then

$$||\hat{\Theta} - \Theta_0||_F = O_P\left(\sqrt{\frac{(p+s)\log p}{N}}\right) ,$$

where $s \geq |T_{\Theta_0}|$, hence the dependence on the sparsity of the true concentration matrix, and $\phi_{\min}(\cdot)$, $\phi_{\max}(\cdot)$ denote the minimum and maximum eigenvalue, respectively. O_P is big O in probability, meaning that for any $\epsilon > 0$ there exists a constant $M > 0$ such that

$$\lim_{N,p \to \infty} \mathbb{P}\left(||\hat{\Theta} - \Theta_0||_F > M\sqrt{\frac{(p+s)\log p}{N}}\right) < \epsilon ,$$

that is, the difference measured in terms of Frobenius norm between the graphical lasso estimator of the concentration matrix $\hat{\Theta}$ and the true concentration matrix Θ_0 grows at most on the order of $\sqrt{\frac{(p+s)\log p}{N}}$ with high probability as both N and p grow to infinity.

Ravikumar et al. (2011) identify a lower bound to the sample size N for the graphical lasso estimator to be *model selection consistent*, meaning that edge set yielded by the estimator $T_{\hat{\Theta}}$ is contained within the true edge set T_{Θ_0}, i.e. it correctly excludes all non-edges, with high probability as $p \to \infty$.

Chapter 2

Coloured GGMs

This chapter will first introduce the family of *coloured Gaussian graphical models* in general, obtained by imposing equality restrictions on the concentration/partial correlation matrix of a GGM, thus imposing symmetries between edges and vertices in the graph. A sub-family of coloured GGMs is suited to deal with the paired data setting, which will be the object of the second part of this chapter.

2.1 Coloured Gaussian graphical models

The family of coloured Gaussian graphical models was first presented in the seminal paper by Højsgaard & Lauritzen (2008). The Authors introduce a new type of Gaussian graphical models that impose symmetry restrictions on the concentration matrix Θ, and therefore also on the conditional independence structure of the model. Three types of coloured GGMs are defined:

1. *RCON* models, imposing equalities between specified entries of the concentration matrix;
2. *RCOR* models, imposing equalities between specified entries of the partial correlation matrix;
3. *RCOP* models, with restrictions generated by permutation symmetry, i.e. symmetry under permutation of variable labels.

The symmetry restrictions can be graphically represented by colouring with the same colour edges and vertices that are constrained to be equal, hence the name coloured GGMs. It is important to note that these constraints lead to a reduction of the model and fewer parameters need to be estimated when coloured GGMs are compared to the general GGMs without symmetry restrictions. This family of model is therefore useful when dealing with high dimensional settings, in which estimation of the concentration matrix can be challenging.

Here only the RCON models will be presented in detail, RCOR and RCOP models are mostly beyond the scope of this work. The reason for this is that the graphical lasso for paired data, which will later be introduced, provides a robust and transparent approach to perform model identification within a subfamily of RCON models, the so called paired data RCON models, or pdRCON models, which is a novelty in the context of coloured GGMs, as model identification is in general challenging and is a big barrier for the family of models to become widely applicable.

2.2 Notation

Colouring the vertices of $G = (V, E)$ with $A \leq |V|$ different colours induces a partition of V into the disjointed sets $V_1, .., V_A$, called *vertex colour classes*, with $V = V_1 \cup ... \cup V_A$ where all vertices belonging to the same vertex colour class V_i have the same colour. In the same way, colouring the edges with $B \leq |E|$ different colours induces a partition of E into the disjointed sets $E_1, .., E_B$, called *edge colour classes*, with $V = V_1 \cup ... \cup V_B$, where all edges belonging to the same edge colour class V_j have the same colour. $\mathcal{V} = \{V_1, ..., V_A\}$ and $\mathcal{E} = \{E_1, ..., E_B\}$ are called *vertex* and *edge colouring*, respectively; the pair $(\mathcal{V}, \mathcal{E})$ defines the *coloured graph* $\mathcal{G}$. The underlying, uncoloured graph $G = (V, E)$ is sometimes referred to as the *skeleton* (Li et al., 2016). A colour class is called *atomic* if it contains a single element; if it is not atomic, it is called *composite*.

2.3 RCON models

An RCON$(\mathcal{V}, \mathcal{E})$ model with vertex colouring $\mathcal{V}$ and edge colouring $\mathcal{E}$ is obtained by restricting entries of the concentration matrix Θ in the following way.

1. For vertices in the same vertex colour class, the corresponding diagonal entries of Θ, i.e. the inverse of the partial variances given the rest of the graph, must be equal. For example, if the vertices $s, t \in V$ have the same colour, then $\theta_{ss} = \theta_{tt}$.

2. For edges in the same edge colour class, the corresponding off-diagonal entries of Θ, i.e. the opposite of the partial covariances, given the rest of the graph, must be equal. For example, if the edges $(s, t), (v, w) \in E$ have the same colour, then $\theta_{st} = \theta_{vw}$.

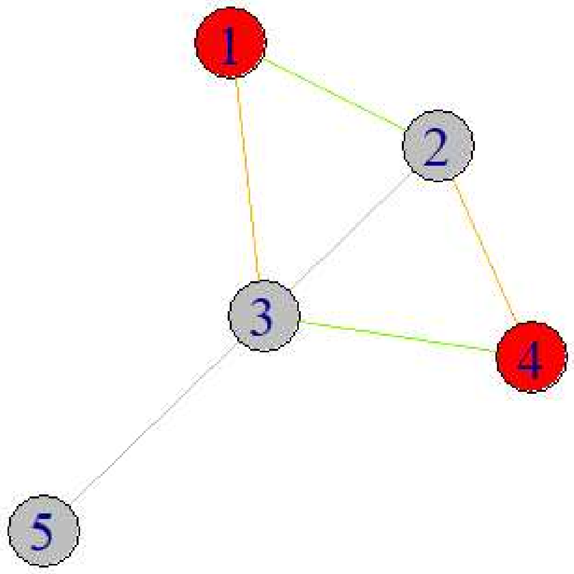


FIGURE 2.1: Example of a coloured graph. The vertex colouring is $\mathcal{V} = \{\{1,4\},2,3,5\}$; the edge colouring is $\mathcal{E} = \{\{(1,3),(2,4)\},\{(1,2),(3,4)\},(2,3),(3,5)\}$.

The RCON model encoded by the coloured graph of Figure 2.1, for example, has a symmetry restriction on the vertices 1 and 4, the two red vertices form the vertex colour class $\{1,4\}$, and the respective concentrations θ_{11} and θ_{44} are constrained to be equal; θ_{22}, θ_{33} and θ_{55} can vary freely as the vertices 2, 3 and 5 are not coloured. Each one of these vertices forms an atomic vertex colour class. The two orange edges form the edge colour class $\{(1,3),(2,4)\}$ and the respective concentrations θ_{13} and θ_{24} are constrained to be equal; the two green edges form the edge colour class $\{(3,4),(1,2)\}$, with the equality restriction $\theta_{34} = \theta_{12}$. The remaining non-coloured edges $(2,3)$ and $(3,5)$ form their own atomic edge colour class, and the respective concentrations θ_{23} and θ_{35} can vary freely. It is then possible to specify the diagonal of Θ with a $R \times 1$ vector of parameters η, and the off-diagonal entries of Θ with a $S \times 1$ vector of parameters δ. Knowing η and δ one can determine Θ, so it is possible to re-write the concentration matrix as $\Theta = \Theta(\eta, \delta)$. $\mathcal{S}^+(\mathcal{V},\mathcal{E})$ denotes the set of all positive definite matrices that satisfy the restrictions of the RCON$(\mathcal{V},\mathcal{E})$ model. Note also that the restrictions defined here are linear in the concentration matrix, so an RCON model is a linear exponential model and the concentration matrix belongs to the class of matrices with a linear structure, as defined in Anderson (1973), meaning that it can be expressed as a linear combination of known symmetric matrices, where the coefficients of the linear combination are the unknown parameters to be estimated.

2.3.1 Likelihood equations for the RCON model

Let T^u be the $|V| \times |V|$ diagonal matrix having entries $T^u_{ss} = 1$ if $s \in u$, where $u \in \mathcal{V}$, and 0 otherwise. Similarly, let T_u be the $|V| \times |V|$ symmetric matrix having entries $T^u_{st} = 1$ if $(s,t) \in u$, where $u \in \mathcal{E}$, and 0 otherwise. Note that u is here a general *generator* for the model RCON$(\mathcal{V}, \mathcal{E})$, and there is no need to distinguish whether u refers to a vertex colour class or an edge colour class.

Denoting with $\xi = (\eta, \delta)$ the $(A+B) \times 1$ vector of unknown parameters, the concentration matrix $\Theta = \Theta(\xi)$ can be re-written as $\Theta = \sum_{u \in \mathcal{V} \cup \mathcal{E}} \xi_u T^u$, and the term $\mathrm{tr}(S\Theta)$ in Equation (1.3) can thus be re-written as $\mathrm{tr}(S\Theta) = \sum_{u \in \mathcal{V} \cup \mathcal{E}} \xi_u \mathrm{tr}(ST^u)$. The maximum likelihood estimate is obtained by equating the canonical statistics to their expectation, yielding the system of equations

$$\mathrm{tr}(ST^u) = \mathrm{tr}(\Sigma T^u) \ , u \in \mathcal{V} \cup \mathcal{E} \ ,$$

and the solution to the system, when it exists, is unique. The conditions for the MLE to exist are however less restrictive then for the general GGM model without symmetry restrictions, and fewer observations are needed.

These likelihood equations can be solved by applying the algorithm for a general linear exponential family that was proposed by Jensen et al. (1991), using a Newton's iterative method on the N-th root of the reciprocal of the likelihood function, maximizing with respect to one parameter at a time. This algorithm is globally convergent in the one-parameter case, and empirical evidences suggests that it is quite stable and may be globally convergent in the multi-parameter case (Højsgaard & Lauritzen, 2007). In the context of the RCON model, this translates to an algorithm that consists of two nested loops, where the outer loop iterates over the elements of $\mathcal{V} \cup \mathcal{E}$ until convergence, and the inner loop maximizes the log-likelihood $\ell(\Theta)$ with respect to ξ_u, while keeping all other parameters fixed. The iterative step in the inner loop is given by the update rule for ξ_u

$$\xi_u \leftarrow \xi_u + \frac{\Delta_u}{\mathrm{tr}(T^u \hat{\Sigma} T^u \hat{\Sigma}) + \Delta_u^2/2} \ ,$$

where Δ_u is the discrepancy $\Delta_u = \mathrm{tr}(T^u \hat{\Sigma}) - \mathrm{tr}(T^u S)$ and $\hat{\Sigma} = \hat{\Theta}^{-1}$ is the current estimate of Σ in this iterative process.

It is worth noting that if the random vector X is rescaled to form X', where $X_s' = a_s X_a$, $s \in V$, then X' will *not* in general satisfy the same restrictions of an RCON model on X, unless $a_s = a$ for all $s \in V$. A standardization procedure, for example, will not preserve the original structure of colour classes. The Authors suggest therefore to use RCON models when all variables are on comparable scales. On the other hand, RCOR models, imposing restrictions on the partial correlations rather than the concentrations, i.e. the partial covariances, have the property of invariance under rescaling if variables in the same colour class are transformed in the same way, i.e. under transformations of the form

$$X^* = \Upsilon X \ , \quad \text{for} \quad \Upsilon = \sum_{u \in \mathcal{V}} \upsilon_u T^u \ . \tag{2.1}$$

Restrictions on the partial correlations, however, are not linear in Θ anymore and the models obtained are curved exponential families. Joint concavity cannot in general be guaranteed and the likelihood function may have multiple local maxima. Moreover, Højsgaard & Lauritzen (2008) point out that a RCON model is also a RCOR model, and is thus invariant for a transformation like in Equation (2.1) if and only if any pair of edges in the same edge colour class $(s,t),(v,w) \in u \in \mathcal{E}$ connects the same vertex colour classes.

2.4 Coloured Gaussian graphical models for paired data

When constructing a graph model, the observations may not always be an i.i.d. sample, but may be organised in two or possibly more dependent groups sharing the same variables, so that there is an association structure between the two groups, as the two corresponding sub-graphs are expected to share some of the features, while retaining some individual, group-specific ones. An example of such scenario is with cancer genomics, where the two groups can be defined as cancer cells and a control sample from histologically normal tissue adjacent to the tumor (see Chapter 4). These two groups will clearly form a dependent sample as the tissues have similar characteristics and belong to the same patient. The *coloured Gaussian graphical model for paired data* was first introduced by Roverato & Nguyen (2022) to deal with this type of situation.

The Authors consider the random vector $X_V = (X_1, ..., X_p)$ to be split into two subvectors $X = (X_L, X_R)^T$ with $|L| = |R| = p/2 = q$, where each variable in the Left block can be naturally paired with the corresponding variable in the Right block, called its *twin* or *homologous* variable. This is denoted by assuming the two edge sets are $L = \{1, ..., q\}$ and $R = \{q + 1, ..., p\}$ and setting $i' = i + q$ for $i \in L$, so that $R = \{1', ..., q'\}$, where every $\{i, i'\}$ for $i \in L$ is a pair of twin variables from the left and right block respectively. A vertex colour class is said to be *twin-pairing* if it is formed by a pair of twin vertices; a twin-pairing edge colour class is a pair of twin edges. Two possible kinds of twin-pairing edge colour class can be distinguished: the *inside-group*, or *inside-block*, type, where the two edges link vertices inside the same group (so pairs of edges such as $\{(i, j), (i', j')\}$, with $i \in L$), and the *across-group*, or *across-block*, type, where the two edges link a vertex from each of the two groups (so pairs of edges such as $\{(i, j'), (i', j)\}$, with $i \in L$). A RCON$(\mathcal{V}, \mathcal{E})$ model is a *RCON model for paired data* (pdRCON model) if all its vertex colour classes and edge colour classes are either atomic or twin-pairing.

An aspect worth noting is that with pdRCON models the concentration values that are compared when imposing symmetry restrictions are associated with twin variables only, measuring common features, so it is to be expected that the scales will be similar. The remark described in Section 2.3.1 about the need for the variables to be measured in comparable scales in RCON models is generally not a problem for the subfamily of pdRCON models.

2.5 Structural learning of a coloured GGM

Learning a coloured GGM is a challenging task, and different solutions have been proposed. Algorithms that perform model search by moving locally between neighbouring models are challenging, as the dimensionality of the search space is higher than the corresponding classic GGM, as p increases (Roverato & Nguyen, 2022) and identifying neighbouring structures is computationally very expensive. To better understand this, one can think of a complete graph with p vertices: for the GGM case, only one possible model can be associated to this graph, while the number of possible coloured GGM for paired data to choose from is equal to $2^{(p/2)^2}$. For the RCON family, Gehrmann (2011) showed that the number of possible RCON models grows super-exponentially with the number p of variables.

A few examples of algorithms that perform structural learning in the context of coloured GGMs will be here presented briefly.

2.5.1 Penalized composite likelihood

Li et al. (2021) proposed a model selection method for the RCON family based on a ℓ_1-penalized composite likelihood function. When a full likelihood function is computationally expensive or infeasible, the composite likelihood is a pseudo-likelihood approach allowing to define an estimation function given from multiplying a sequence of low-dimensional, and easier to deal with, conditional or marginal distributions. Maximizing the log-composite likelihood can still provide consistent estimation and has good asymptotic properties (Varin et al., 2011).

The method is based on the conditional distribution of X_j given the rest of the graph, which is given by (see Appendix B for details)

$$X_{j|V\setminus\{j\}} \sim N\left(-\sum_{i\neq j} X_i \frac{\theta_{ij}}{\theta_{jj}}, \frac{1}{\theta_{jj}}\right) ,$$

with the density function

$$f(x_j|X_{V\setminus\{j\}};\Theta) = \frac{\theta_{jj}^{1/2}}{\sqrt{2\pi}} \exp\left\{-\frac{1}{2}\theta_{jj}\left[x_j + \theta_{jj}^{-1}\left(\sum_{i=1,i\neq j}^{p} \theta_{ij}x_i\right)\right]^2\right\} .$$

The conditional composite likelihood is then given by

$$L_c(\Theta) = \prod_{k=1}^{N}\prod_{j=1}^{p} \frac{\theta_{jj}^{1/2}}{\sqrt{2\pi}} \exp\left\{-\frac{1}{2}\theta_{jj}\left[x_kj + \theta_{jj}^{-1}\left(\sum_{i=1,i\neq j}^{p} \theta_{ij}x_ki\right)\right]^2\right\} ,$$

where x_{ki} is the entry in row k and column i of the $N \times p$ data matrix X. The conditional composite log-likelihood is then

$$\begin{aligned}\ell_c(\Theta;X) &= \sum_{k=1}^{N}\sum_{j=1}^{p}\left\{\frac{1}{2}\log\theta_{jj} - \frac{1}{2}\theta_{jj}\left[x_{kj} + \theta_{jj}^{-1}\left(\sum_{i\neq j}\theta_{ij}x_{ki}\right)\right]^2\right\} \\ &= \frac{1}{2}\sum_{j=1}^{p}\left\{N\log\theta_{jj} - \theta_{jj}\sum_{k=1}^{N}\left[x_{kj} + \theta_{jj}^{-1}\left(\theta_{ij}x_{ki}\right)\right]^2\right\} .\end{aligned}$$

It can be rewritten in a matrix format as

$$\ell_c(\Theta;X) = \frac{1}{2}\sum_{j=1}^{p}\left\{N\log\theta_{jj} - \theta_{jj}||X_{(j)} - XB_j||^2\right\} ,$$

where $X_{(j)}$ is the j-th column of X and B_j is the j-th column of the matrix $(\beta_{j\leftarrow i|V\setminus\{i,j\}})_{p\times p}$, recalling that $\beta_{j\leftarrow i|V\setminus\{i,j\}} = -\frac{\theta_{ij}}{\theta_{jj}}$ from Chapter 1. The off-diagonal elements $\theta_{ij}, i \neq j$, are rewritten according to their lexicographical order as the vector $\zeta = (\zeta_1, ..., \zeta_{\frac{p(p-1)}{2}})^T$, so that the concentration matrix Θ can be expressed as the parameter vector $\theta = (\theta_{11}, \theta_{22}, ..., \theta_{pp}, \zeta^T)^T$.

The regularized conditional composite log-likelihood approach suggested by the Authors is given by the minimization problem

$$\hat{\theta}^{\text{pcl}} = \arg\min_{\theta} \Bigg\{ -\frac{1}{N}\ell_c(\theta) + \lambda_1 \sum_{j<j'} J_\tau(|\theta_{jj} - \theta_{j'j'}|) + \lambda_2 \sum_{j=1}^{\frac{p(p-1)}{2}} J_\tau(|\zeta_j|) + \lambda_3 \sum_{j<j'} J_\tau(|\zeta_j - \zeta'_j|) \Bigg\} ,$$

where λ_1, λ_2 and λ_3 are non-negative tuning parameters, τ is a treshold parameter determining the strength of penalization on off-diagonal elements and differences between element pairs of the concentration matrix, and $J_\tau(x) = \min\left(\frac{x}{\tau}, 1\right)$. These four parameters can be tuned minimizing in a four-dimensional parameter space using a grid search procedure the Bayesian information criterion for composite likelihood (Gao & Song, 2010),

$$\text{BIC}_c = -2\ell_c(\hat{\theta}^{\text{pcl}}) + df \log N ,$$

where *df* denotes the total number of parameters in the concentration matrix.

The Authors propose a computationally efficient method to solve the minimization problem by combining a difference of convex functions (DC) algorithm, the augmented Lagrangian approach and coordinate descent optimization. A simulation study generating data from different symmetric graph structures shows that the method performs fairly well both in recovering the conditional dependence relationships, i.e. the structure of the skeleton, as well as the symmetry structures.

2.5.2 Bayesian model selection approach

Bayesian inference for undirected graphs often uses a conjugate Wishart prior (see e.g. Letac & Massam, 2007) for the concentration matrix Θ. A G-Wishart prior distribution for the concentration matrix of the graph $G = (V, E)$, having then off-diagonal entries constrained to be equal to zero whenever $(i, j) \notin E$, $\text{Wis}_G(\eta, D)$ has the density

$$p(\Theta|G, \eta, D) = \frac{1}{I_G(\eta, D)}(\det\Theta)^{(\eta-2)/2} \exp\left\{-\frac{1}{2} < \Theta, D >\right\} ,$$

defined over the space of positive definite matrices $\mathcal{S}_+^p$, where $< A, B >= \operatorname{tr}(A^T B)$ denotes the trace inner product, η is the shape parameter controlling for the amount of prior confidence about the structure of Θ and D is a symmetric scale matrix serving as a prior estimate for Θ, setting the prior mean for the concentration matrix. The normalizing constant $I_G(\eta, D)$ is finite if $\eta > 2$ and D is positive definite. When G is the full graph, i.e. $E = V \times V$, the G-Wishart distribution becomes the classic Wishart distribution $\operatorname{Wis}(\eta, D)$. The Wishart distribution is the distribution of the sample covariance matrix from observations from a multivariate normal distribution. Its extension for the coloured GGM is the coloured G-Wishart distribution (Massam et al., 2018), which is a Diaconis-Ylvisaker distribution having density

$$p(\Theta|\eta, D, G) = \frac{1}{I_G(\eta, D)} (\det \Theta)^{(\eta-2)/2} \exp\left\{-\frac{1}{2} < \Theta, D >\right\} \mathbb{I}_{P_\mathcal{G}}(\Theta) \ ,$$

defined over the space of positive definite matrices $\mathcal{S}_+^p$, and is also the conjugate prior. Here $\mathbb{I}(\cdot)$ is the indicator function and $P_\mathcal{G}$ is the cone of $p \times p$ positive definite matrices obeying to the symmetry constraints of the coloured graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$:

$$P_\mathcal{G} = \{\Theta \in \mathcal{S}_+^p | \text{if } V_k \text{ is a vertex colour class in } \mathcal{V}\text{, then for all } i \in V_k,\ \theta_{ii} \text{ are equal}$$
$$\text{and if } E_l \text{ is an edge colour class in } \mathcal{E}\text{, then for all } (i,j) \in E_l\text{, the entries } \theta_{ij} \text{ are equal}\} \ .$$

The posterior density given the $N \times p$ data matrix X is proportional to

$$\begin{aligned} p(G, \Theta|X) &= p(X|G, \Theta) p(\Theta|G) p(G) \\ &\propto \frac{1}{I_G(\eta, D)} (\det \Theta)^{(N+\eta-2)/2} \exp\left\{-\frac{1}{2} < \Theta, S + D >\right\} p(G) \mathbb{I}_{P_\mathcal{G}}(\Theta) \ , \end{aligned}$$

where S is the sample covariance matrix and $p(G)$ is the prior distribution on the graph structure. As mentioned before, the number of possible coloured graphical models grows super-exponentially with the number of variables p so an extensive search through the space of coloured graph is computationally impossible, except in very-low dimensional cases.

Qiong Li & Massam (2020) propose instead an hill-climbing algorithm exploring neighbouring configurations that combines linear regression with a double reversible jump. The double reversible jump makes it possible to calculate the Bayes factor without having to compute the prior normalizing constants, which is a computational stumbling block. The approach is as follows:

1. Start with the graph $G^{[1]}$ with p vertices, no edges, and no symmetry restrictions.

2. Let $G^{[t]}$ be the current version of the graph. Repeat for $j = 1, ..., p$:

   (a) fit a linear regression to search for potential edges to add to the graph $G^{[t]}$

   $$X_j \sim \beta_1^j X_1 + ... + \beta_{j-1}^j X_{j-1} + \beta_{j+1}^j X_{j+1} + ... + \beta_p^j X_p \ ;$$

   (b) set $\beta^j_{G^{[t]}} = \{\beta^j_i | (i,j) \in G^{[t]}\}$ and find the significant regression coefficients β_i^j with p-values less than the specified α level among $\{\beta_1^j, ..., \beta_{j-1}^j, \beta_{j+1}^j, ..., \beta_p^j\} \setminus \beta^j_{G^{[t]}}$;

   (c) order the edges from the most significant to the least significant. Whether an edge is accepted or not and its colour class is determined on the basis of the Bayes factor as follows:

      i. Let G^* be the graph obtained by adding the edge (i,j) to $G^{[t]}$. Estimate the Bayes factor $p(G^*|X)/p(G^{[t]}|X)$ with the double reversible jump algorithm. Whenever the ratio is greater the one, the graph G^* is accepted and $G^{[t+1]} = G^*$; otherwise, $G^{[t+1]} = G^{[t]}$.

      ii. It is then checked if the new edge can be added to one of the existing edge colour classes. For every existing edge colour class $l = 1, ..., k$:

         A. if (i,j) was accepted in the previous step, let G^{**} be the graph obtained by adding (i,j) to the l-th edge colour class. Estimate the Bayes factor $p(G^{**}|X)/p(G^{[t+1]}|X)$ and if $G^{[**]}$ is accepted, $G^{[t+2]} = G^{**}$ is set, and the next most significant edge in step 2.(c) is investigated;

         B. if (i,j) is not accepted in the l-th colour class and $l \neq k$, let $l = l + 1$ and the algorithm goes back to the previous step;

         C. if $l = k$ and the new edge (i,j) cannot be merged into any of the existing colour classes, (i,j) forms a new atomic edge colour class and the graph remains $G^{[t+2]} = G^{[t+1]}$

3. Determine the vertex colour classes. For $r = 1, ..., p$:

   (a) for every existing vertex colour class $m = 1, ..., w$:

      i. Let G^{**} be the graph obtained from $G^{[t+2]}$ by setting the vertex r to be in the m-th colour class. Estimate the Bayes factor $p(G^{**}|X)/p(G^{[t+2]}|X)$. If the Bayes factor is larger than one, the proposed vertex colour merging is accepted and $G^{[t+3]} = G^{**}$. Let $r = r + 1$ and the algorithm goes back to step 3;

ii. If the Bayes factor is less than one and $m \neq w$ let $m = m + 1$ and the algorithm goes back to step 3.(a) for the next existing vertex colour class;

iii. if $l = m$ and the r-th vertex cannot be merged into any existing vertex colour classes, it becomes a new atomic vertex colour class.

The Authors prove that the probability that the Bayes factor comparing the true model versus any other model is greater than one converges to one as the sample size N increases. Although this does not guarantee model selection convergence for the procedure as a whole as it is computationally impossible to compare all possible candidate models, the algorithm still moves towards the true graph.

2.5.3 Stepwise procedure for the pdRCON model

Roverato & Nguyen (2022) show how the subclass of pdRCON models form a sublattice of RCON models, although a non-distributive one, which makes the computation of neighbouring models difficult. The Authors then implemented a stepwise backward elimination procedure to perform model selection for the pdRCON class, in which, starting from the largest model, at each step all the neighbouring submodels form the set of candidate models; all the submodels of previously rejected models are instantly discarded, and the remaining submodels are classified as either accepted or rejected on the basis of a likelihood ratio test at the 0.05 significance level, and the accepted model with the highest p-value is the starting point at the next step of the algorithm. The procedure stops whenever all candidate models are rejected.

2.6 The symmetric graphical lasso

Ranciati et al. (2021) proposed a penalized likelihood method with a fused-type lasso penalization to make inference on the brain connectivity network by measuring the BOLD level, a measure of brain activity reflecting changes in blood oxygenation levels in response to neural activity, via resting state functional magnetic imaging (*R-fMRI*) scans data, collected on subjects at rest, without external stimuli. The objective is to understand the neuronal organization of the brain through the investigation of spontaneous neural activity.

2.6.1 Description of the problem

The cerebral cortex is subdivided into $p = 70$ spatial regions of interest (*ROIs*) by clustering anatomically and functionally similar voxels from the R-fMRI scans according to Desikan et al. (2006). It is a well known fact that the human brain has a natural symmetric structure, so that every spatial ROI on the left hemisphere has an homologous spatial ROI on the right hemisphere, and in the same way in the graph modelling the brain network one can identify pairs of homologous vertices and edges.

Despite this strong structure-wise symmetry, however, the two hemispheres exhibit *functional* asymmetry, also known as *lateralization*, referring to activity-related differences, which are usually detected while the patients perform tasks connected to language organization and handedness, (e.g., spatial reasoning is more often associated with the right hemisphere and language processing with the left hemisphere). This can be compared to how the right and left hand are anatomically symmetric, but can operate differently. Previous R-fMRI studies, which represent spontaneuos, low-frequency fluctuations in brain activity, measured via the BOLD signal when the subject is not performing any explicit task and is simply asked to stay awake with eyes open, suggested that brain networks show strong bilateral synchronization. Brain activity in one ROI is most often mirrored with activity in its homologous ROI in the other hemisphere.

2.6.2 Methodological framework

The data is the $p \times T$ matrix measuring brain activity over the $p = 70$ spatial ROIs at $t = 1, ..., T$ equally spaced time points, with $T = 404$. An additive signal and noise decomposition for a generic column $X^{(t)}$ is assumed,

$$X^{(t)} = M^{(t)} + Y^{(t)} ,$$

where $M^{(t)}$ is the vector collecting the BOLD signal and $Y^{(t)}$ is the idiosyncratic noise component. The ROI network association structure will be based on the estimate of $Y^{(t)}$, obtained by extracting the BOLD signal $M^{(t)}$ from $X^{(t)}$ via three different filtering techniques:

1. a first-order vectorial autoregressive model, VAR(1);
2. a score-driven model;
3. local polynomial regression.

While a detailed discussion over the different detrending techniques is beyond the scope of this work, it is worth noting that a more robust approach, such as the score-driven model, leaves more information to the vector of residuals, leading to an estimated brain association network with more symmetries; on the other hand, an approach that may tend to overfit the data, such as a high-degree local polynomial regression, may put most of the dependence structure in the signal rather than the noise component, leading to a less symmetric graph. As there is no general consensus on how to perform the detrending, it may be best to use different methods and compare the results.

Letting $V = \{1,..,p\}$ denote denote the set indexing the $p = 70$ ROIs, $Y_V = (Y_1,...,Y_{70})^T$ is the zero-mean residual vector, where the time index can be dropped as it is assumed that the time series dynamics have been fully captured by the BOLD signal $M^{(t)}$. The vector Y_V can be naturally partitioned into the subvectors Y_L and Y_R, associated, respectively, with the left and right hemispheres, with $L = \{1,...,q\}$ and $R = \{q+1,...,p\}$ where $q = p/2$. In this way, the region Y_i of Y_L is homologous to the region Y_{i+q} of Y_R. To simplify the notation, $i' = i + q$ for every $i \in L$.

Joint normality is assumed, $Y_V \sim N_p(0, \Sigma)$, and $\Theta = \begin{pmatrix} \Theta_{LL} & \Theta_{LR} \\ \Theta_{RL} & \Theta_{RR} \end{pmatrix}$ is the partition of the concentration matrix $\Theta = \Sigma^{-1}$ with respect to the two groups. Interest is in estimating an undirected graph taking into account the symmetric structure between the two hemispheres represented by Y_L and Y_R.

2.6.3 The symmetric graphical lasso estimator

Ranciati et al. (2021) define the *symmetric graphical lasso* estimator as

$$\hat{\Theta}^{\text{sgl}} = \underset{\Theta \in \mathcal{S}_+^p}{\arg\min} \{-\log\det(\Theta) + \text{tr}(S\Theta) + \lambda_1 ||\Theta||_1 + \lambda_2 ||\Theta_{LL} - \Theta_{RR}||_1\} ,$$

minimizing a (minus) penalized log-lihelihood, where λ_1 is the regularization parameter encouraging sparsity in Θ, remarking that this first penalization term is equal to the one in Equation (1.4), and λ_2 is the regularization parameter encouraging elements of $\hat{\Theta}^{\text{sgl}}_{LL}$ in the solution to be equal to the corresponding elements of $\hat{\Theta}^{\text{sgl}}_{RR}$, $\hat{\Theta}^{\text{sgl}} = \begin{pmatrix} \hat{\Theta}^{\text{sgl}}_{LL} & \hat{\Theta}^{\text{sgl}}_{LR} \\ \hat{\Theta}^{\text{sgl}}_{RL} & \hat{\Theta}^{\text{sgl}}_{RR} \end{pmatrix}$ being the partition of the solution with respect to the two groups. This is a convex optimization problem, and the Authors provided an alternating directions of multipliers (ADMM) algorithm to solve it in a computationally feasible way, even in high dimensional settings. It is a first instance of a lasso method that performs simultaneously model selection and estimation within the class of RCON models, as the solution obtained is a pdRCON model, identifying twin-pairing vertex colour classes, corresponding to $\hat{\theta}^{\text{sgl}}_{ii} = \hat{\theta}^{\text{sgl}}_{i'i'}$ in the solution, and inside-group twin-pairing edge colour classes, corresponding to $\hat{\theta}^{\text{sgl}}_{ij} = \hat{\theta}^{\text{sgl}}_{i'j'}$ in the solution, which are of natural interest in the analysis of brain networks.

It does not, however, provide a general solution to perform model selection in the pdRCON class, as it cannot identify across-group twin-pairing edge colour classes. The graphical lasso for paired data (Ranciati & Roverato, 2024a) extends further on the idea of the symmetric graphical lasso in order to deal with the whole family of pdRCON models.

The Authors carry out a simulation study analyzing the performance of the symmetric graphical lasso on generated data mimicking the structure of the fMRI data. Choosing the values of the regularization parameters with an oracle procedure that selects the value of λ_1 yielding a graph with equal edge density of the true graph used to simulate the data, the results of the simulation study show that the symmetric graphical lasso performs only slightly worse than the general graphical lasso in recovering the overall structure of the true graph, with even smaller differences in the denser scenarios. Moreover, the symmetric graphical lasso exhibits a satisfying performance in terms of recovery of symmetries in the true graph, which is something that, on the other hand, the graphical lasso cannot do.

The Authors then apply the symmetric graphical lasso to the real-world data from the R-fMRI scans after filtering out the time-dependency structure, selecting the values of λ_1 and λ_2 by looking at the extended Bayesian information criterion,

$$\text{eBIC} = -2\ell(\hat{\Theta}^{\text{mle}}) + \log(N)d + 4d\gamma\log(p)\ , \tag{2.2}$$

where $\ell(\hat{\Theta}^{\text{mle}})$ is the maximized unpenalized log-likelihood function for the pdRCON model, d is the number of free parameters in the model and γ is the parameter that controls the amount of preference expressed by the criterion towards more parsimonious models, with $\gamma = 0$ coinciding with the standard BIC, choosing in this case $\gamma = 0.5$. As a joint grid search over λ_1 and λ_2 is computationally intensive, first λ_2 is fixed to an arbitrarily low value and a dense grid search over λ_1 is performed, selecting λ_1^*; then after a conditional sweep over a grid of 20 logarithmically spaced values for λ_2, the final pair of optimal values for the regularization parameters $(\lambda_1^*, \lambda_2^*)$ is selected. After estimating two symmetric graphical lasso model on the filtered R-fMRI data from two subjects, one of which has a mental disorder and the other one is healthy, the graphs obtained show that, regardless of the filtering method chosen in the preliminary step, the mentally ill patient has a more symmetric brain network association structure than the healthy one, which is line with the evidence of lack of asimmetry in schizophrenic patients shown in Li et al. (2019). An implementation in R of the symmetric graphical lasso can be found at https://github.com/savranciati/sgl.

2.7 The graphical lasso for paired data

2.7.1 Structural and parametric symmetries

Consider the partition of the concentration matrix in a paired data setting according to the split of the random vector X in the left and right block, described in Section 2.4, $\Theta = \begin{pmatrix} \Theta_{LL} & \Theta_{LR} \\ \Theta_{RL} & \Theta_{RR} \end{pmatrix}$.

Denoting by $G_A = (A, E_A)$ the subgraph of the graph $G = (V, E)$ induced by the subset $A \subseteq V$, where $(i, j) \in E_A$ if and only if both $i, j \in A$ and $(i, j) \in E$, then if Θ is adapted to G, i.e. every missing edge of G implies a zero off-diagonal entry in Θ, Θ_{LL} and Θ_{RR} are adapted to G_L and G_R, respectively. G_L and G_R are thus the group (or block) specific graphs.

Ranciati & Roverato (2024a) introduce a distinction between *structural* and *parametric* symmetries for the pdRCON context, where structural symmetries deal with the fact that two edges are either both present or not, whereas parametric symmetries are more restrictive and deal with the fact that the corresponding entries of the concentration matrix are also equal.

There is an *inside-block structural symmetry* if, for a pair $i, j \in L$, with $i \neq j$, the edges (i, j) and (i', j') are either both present or both missing in the two subgraphs G_L and G_R, respectively. There is an *across-block structural symmetry* if the edges (i, j') and (i', j) are either both present or both missing in G, concerning therefore the association structure present between the two groups. A parametric symmetry can be inside-block (if $\theta_{ij} = \theta_{i'j'}$), across-block (if $\theta_{ij'} = \theta_{i'j}$) and it can also involve vertices: there is a *vertex parametric symmetry* if $\theta_{ii} = \theta_{i'i'}$, for $i \in L$.

It might be worth noting that every structural symmetry due to a pair of missing edges corresponds also to parametric symmetry, as the two corresponding entries of the concentration matrix will both be zero, and thus equal.

Ranciati & Roverato (2024a) also point out that in the context of pdRCON models the colouring of edges and vertices is redundant, as a pdRCON can be uniquely identified by simply determining whether each parameter is constrained or not, distinguishing thus only between coloured and non-coloured edges and vertices.

2.7.2 The graphical lasso for paired data estimator

In order to perform simultaneously estimation and model selection within the family of pdRCON models Ranciati & Roverato (2024a) introduce the graphical lasso for paired data estimator, obtained by minimizing the penalized log-likelihood

$$\hat{\Theta}^{\text{glpd}} = \underset{\Theta \in \mathcal{S}_p^+}{\arg\min} \left\{ -\ell(\Theta) + \mathcal{P}_{\lambda_1}(\Theta) + \mathcal{Q}_{\lambda_2}(\Theta) \right\} \ , \tag{2.3}$$

where $\ell(\Theta) = \log\det(\Theta) - \text{tr}(S\Theta)$ is the log-likelihood function of a GGM defined in Equation (1.3), $\mathcal{P}_{\lambda_1} = \lambda_1 ||\Theta||_1$ is the penalization term of the standard graphical lasso of Equation (1.4) encouraging sparsity in Θ, $\mathcal{Q}_{\lambda_2}$ is a fused-type lasso penalty defined as

$$\mathcal{Q}_{\lambda_2} = \lambda_2 (||\Theta_{LL} - \Theta_{RR}||_1 + ||\Theta_{LR} - \Theta_{RL}||_1) \ , \tag{2.4}$$

encouraging parametric symmetries and $\lambda_1, \lambda_2 \geq 0$ are regularization parameters.

The first term of the fused-type lasso penalty,

$$\lambda_2 ||\Theta_{LL} - \Theta_{RR}||_1 = \lambda_2 \sum_{i \in L} |\theta_{ii} - \theta_{i'i'}| + \lambda_2 \sum_{i,j \in L, i \neq j} |\theta_{ij} - \theta_{i'j'}| \ ,$$

encourages both vertex parametric symmetries and inside-block parametric symmetries, whereas the second term,

$$\lambda_2 ||\Theta_{LR} - \Theta_{RL}||_1 = \lambda_2 \sum_{i,j \in L} |\theta_{ij'} - \theta_{i'j}| \ ,$$

encourages across-block symmetries. Note that $|\theta_{ij'} - \theta_{i'j}|$ will be equal to zero whenever $i = j$ because Θ is symmetric.

2.7.3 Path of solutions

In order to define a grid of penalty parameters values to compute the path of solutions, it is of interest to identify the value of λ_1 that yields a diagonal matrix as the solution and the value of λ_2 that yields a fully symmetric solution, with both $\Theta_{LL} = \Theta_{RR}$, i.e. full vertex parametric symmetry and full structural and parametric inside-block symmetry, and $\Theta_{LR} = \Theta_{RL}$, i.e. full structural and parametric across-block symmetry, to be set as the maximum values of the grid.

From the result mentioned in Section 1.3.2 about the necessary and sufficient condition for the solution of the graphical lasso to be block diagonal, it can easily be proven by considering each diagonal entry to be a block of its own that the solution is a diagonal matrix if and only if $\lambda_1 \geq \lambda_1^{\text{diag}}$, where $\lambda_1^{\text{diag}} \max_{i,j \in V, i \neq j} |s_{ij}|$. Moreover, the result in Section 1.3.2 implies also that the solution of the graphical lasso is here block diagonal ($\hat{\Theta}_{LR}^{\text{glpd}} = \hat{\Theta}_{LR}^{\text{glpd}} = O_q$), meaning that there are no across-block edges and the two groups are then completely indepedent, whenever $\lambda_1 \geq \max_{i,j \in L} |s_{ij'}|$. Ranciati & Roverato (2024a) prove that both of these results still hold true for the graphical lasso for paired data even when $\lambda_2 > 0$. Additionally, the Authors prove that a sufficient solution for the solution $\hat{\Theta}^{\text{glpd}}$ to be fully symmetric is that $\lambda_2 \geq \lambda_2^{\text{sym}}$, where $\lambda_2^{\text{sym}} = \max\{|s_{ij} - s_{i'j'}|/2, |s_{i'j} - s_{ij'}|/2 | i, j \in L\}$.

As the model selection criterion the eBIC (2.2) is considered, defined this time as

$$\text{eBIC} = -N\ell(\hat{\Theta}^{\text{mle}}) + \log(N)d + 4d\gamma \log(p) ,$$

where the maximized log-likelihood of a given pdRCON model can be computed using a penalized log-likelihood (2.3) where every entry of the concentration matrix is penalized differently. $\lambda_1 = \infty$ is set for every missing edge in the corresponing coloured graph, and $\lambda_1 = 0$ for every non-missing edge; $\lambda_2 = \infty$ for every present symmetry and $\lambda_2 = 0$ for every non-symmetry.

First, λ_2 is fixed to zero and the graphical lasso solution is computed for a sequence of m equally spaced on the log-scale values from $\lambda_1^{\text{diag}}/m$ to λ_1^{diag}, thus identifying the optimal value of λ_1, λ_1^*; fixing $\lambda_1 = \lambda_1^*$, the graphical lasso for paired data solution is computed for a sequence of m equally spaced on the log-scale values from λ_2^{sym}/m to λ_2^{sym}. The optimal pair of regularization parameters values $(\lambda_1^*, \lambda_2^*)$ is then chosen comparing the eBIC values for these last m solutions, plus the case $\lambda_2 = 0$, i.e. the standard graphical lasso.

2.7.4 Relevant pdRCON submodel classes

The Authors propose also a different way to define the fused-type penalty (2.4) in order to implement a wider flexibility by associating every type of parametric symmetry with a regularization parameter of its own. This different specification of the fused-type penalty is given by

$$\mathcal{Q}_{\lambda_2} = \lambda_2^{(V)} ||\text{diag}(\Theta_{LL}) - \text{diag}(\Theta_{RR})||_1 + \lambda_2^{(I)} ||\Theta_{LL}^* - \Theta_{RR}^*||_1 + \lambda_2^{(A)} ||\Theta_{LR} - \Theta_{RL}||_1 ,$$

where $\Theta_{LL}^* = \Theta_{LL} - \text{diag}(\Theta_{LL})$, $\Theta_{RR}^* = \Theta_{RR} - \text{diag}(\Theta_{RR})$ and $\text{diag}(A)$ is a diagonal matrix having the same diagonal as A. $\lambda_2^{(V)}$ is then associated with vertex symmetry, $\lambda_2^{(I)}$ with inside-block symmetry and $\lambda_2^{(A)}$ with across-block symmetry. Each of the three parameters can take the value:

- $0 \rightarrow$ no contraints regarding that type of symmetry;
- $\lambda_2 \rightarrow$ the amount of symmetry regularization implied by λ_2;

- $\infty \rightarrow$ full symmetry.

One can then select a model within one of $|\{0, \lambda_2, \infty\}|^3 = 27$ different pdRCON submodel classes.

A relevant pdRCON submodel class is given for example by assuming full vertex symmetry, i.e. $\lambda_2^{(V)} = \infty$, thus imposing all partial variances $\sigma^2_{ii|V\setminus\{i\}} = 1/\theta_{ii}$ to be equal. If this (strong) assumption is valid, such a model has the advantage that equalities between off-diagonal entries of the concentration matrix can directly be interpretable in terms of equality of the corresponding partial correlations (see Section 1.2), whereas that does not hold true for a pdRCON model in general.

If, for example, interest is in the two subgraphs G_L and G_R and the across-group association structure encoded by Θ_{LR} is considered to be a nuisance parameter, choosing $\lambda_2^{(A)} = \infty$ allows the number of parameters to be estimated to be lower. Considerations on the across-group association structure will be investigated in further detail in the simulation study of Chapter 3.

A *fully symmetric model*, having $\lambda_2^V = \lambda_2^{(I)} = \lambda_2^{(A)} = \infty$ can be a useful benchmark for comparisons. Such a model belongs to the family of RCOP models, meaning the two random vectors X_L, X_R are exchangeable and the concentration matrix Θ is invariant under the action of $J = \begin{pmatrix} O_q & I_q \\ I_q & O_q \end{pmatrix}$, i.e. $J\Theta J = \Theta$. Interestingly, Højsgaard & Lauritzen (2008) proved that in this case the estimate of Θ can simply be computed as the MLE of the standard GGM without symmetry restrictions by considering the average sample covariance matrix $\overline{S} = (S + JSJ)/2$.

2.7.5 ADMM algorithm

The restrictions defined for a pdRCON model are once again linear in the concentration matrix, it is a linear exponential model, the log-likelihood function is concave and the graphical lasso for paired data represents a convex optimization problem. For the solution the Authors propose an Alternating Directions Method of Multipliers (ADMM) algorithm (Boyd et al., 2011), which breaks down the problem into smaller, more manageable subproblems that can be solved efficiently. It alternates updates on each variable while keeping all others fixed, and iterating until convergence is reached. The algorithm here extends the one proposed for the symmetric graphical lasso (Ranciati et al., 2021).

In this case, the procedure consists of two nested optimization problems, each solved with an ADMM algorithm. The optimization problem (2.3) is equivalent to the minimization with respect to Θ, Z of the quantity

$$-\log\det(\Theta) + \mathrm{tr}(S\Theta) + \lambda_1||Z||_1 + \lambda_2^{(V)}||\mathrm{diag}(Z_{LL}) - \mathrm{diag}(Z_{RR})||_1 + \\ \lambda_2^{(I)}||Z^*_{LL} - Z^*_{RR}||_1 + \lambda_2^{(A)}||Z_{LR} - Z_{RL}||_1 \ ,$$

where $\Theta, Z \in \mathcal{S}_p^+$ and subject to the linear constraint $Z = \Theta$. The scaled form (Boyd et al. (2011), Section 3.1.1) of the augmented Lagrangian is then

$$\begin{aligned} L_{\rho_1}(\Theta, Z, U) = & -\log\det(\Theta) + \mathrm{tr}(S\Theta) + \lambda_1||Z||_1 + \lambda_2^{(V)}||\mathrm{diag}(Z_{LL}) - \mathrm{diag}(Z_{RR})||_1 + \\ & \lambda_2^{(I)}||Z^*_{LL} - Z^*_{RR}||_1 + \lambda_2^{(A)}||Z_{LR} - Z_{RL}||_1 + \\ & \frac{\rho_1}{2}||\Theta - Z + U||_F^2 - \frac{\rho_1}{2}||U||_F^2 \ , \end{aligned} \tag{2.5}$$

where U is the scaled dual variable, $||\cdot||_F$ is the Frobenius norm (see Section 1.3.3 for the definition), and ρ_1 is the augmented Lagrangian parameter defining the step size, controlling the penalty for violating the linear constraint $Z = \Theta$. When the algorithm is at convergence, the constraint $Z = \Theta$ cancels out the last two terms of Equation (2.5) and the solution is obtained. Equation (2.5) can be minimized by initializing Z^1 and U^1 equal to the zero matrix and alternating updates for the quantities Θ, Z, U iteratively through these three following steps:

1. $\Theta^{l+1} = \underset{\Theta \in \mathcal{S}_p^+}{\arg\min} \left\{ -\log\det(\Theta) + \mathrm{tr}(S\Theta) + \frac{\rho_1}{2}||\Theta - Z^l + U^l||_F^2 \right\}$;

2. $$\begin{aligned} Z^{l+1} = \underset{Z \in \mathcal{S}_p^+}{\arg\min} \big\{ & \lambda_1||Z||_1 + \lambda_2^{(V)}||\mathrm{diag}(Z_{LL}) - \mathrm{diag}(Z_{RR})||_1 + \lambda_2^{(I)}||Z^*_{LL} - Z^*_{RR}||_1 + \\ & \lambda_2^{(A)}||Z_{LR} - Z_{RL}||_1 + \frac{\rho_1}{2}||\Theta^{l+1} - Z + U^l||_F^2 \big\} \ ; \end{aligned}$$

3. $U^{l+1} = U^l + \Theta^{l+1} - Z^{l+1}$,

for $l = 1, 2, ...$ until convergence.

Step 1. coincides with the first step of the ADMM algorithm for graphical lasso detailed in Boyd et al. (2011) and can be solved analytically. Letting QDQ^T be the eigendecomposition of the matrix $\rho_1(Z^l - U^l) - S$, with d_{ii} being the i-th diagonal entry of D, the analytical solution is given by $\Theta^{l+1} = Q\tilde{D}Q^T$, where D is a diagonal matrix with i-th diagonal entry $\tilde{d}_{ii} = \frac{d_{ii} + \sqrt{d_{ii}^2 + 4\rho_1}}{2\rho_1}$. Each $\tilde{d}_{ii}$ will always be positive because $\rho_1 > 0$ so the condition $\Theta^{l+1} \in \mathcal{S}_p^+$ will always be satisfied.

Step 2. needs to be solved with an ADMM algorithm of its own. Let vech$(\cdot)$ denote the *half vectorization* operator, i.e. an operator converting a symmetric matrix into a vector by stacking the elements from its lower triangolar part, diagonal included, vd$(\cdot)$ denote the *diagonal extraction* operator, i.e. the operator extracting the diagonal entries of a square matrix and stacking them into a vector, and A a generic matrix having rows and columns also indexed by $V = L \cup R$. The vector $v(A)$ is defined as

$$v(A) = \Big[\mathrm{vd}(A_{LL})^T \mathrm{vd}(A_{RR})^T \mathrm{vech}(A_{LL})^T \mathrm{vech}(A_{RR})^T \\ \mathrm{vech}(A_{LR})^T \mathrm{vech}(A_{RL})^T \mathrm{vd}(A_{LR})^T \Big] ,$$

and the vectors z^l, b^l are defined as $z^l = v(Z^l)$ and $b^l = v(\Theta^l) + v(U^l)$. Letting $s = \frac{q(q-1)}{2}$, the equality constraints associated with parametric symmetries can be encoded in the matrix

$$F = \begin{pmatrix} F_1 \\ F_2 \\ F_3 \end{pmatrix} ,$$

where

$$F_1 = \begin{pmatrix} I_q & -I_q & O_{q,4s+q} \end{pmatrix} , \quad F_2 = \begin{pmatrix} O_{s,2q} & I_s & -I_s & O_{s,2s+q} \end{pmatrix} \text{ and}$$
$$F_3 = \begin{pmatrix} O_{s,2q+2s} & I_s & -I_s & O_{qq} \end{pmatrix} . \tag{2.6}$$

Step (2) can then be re-stated as the optimization problem

$$\arg\min_{z} \left\{ \frac{1}{2} ||z - b||_2^2 + \lambda_1' ||z||_1 + ||F(\lambda_2' \circ z)||_1 \right\} , \tag{2.7}$$

where $\circ$ denotes the Hadamard product (see Section 1.3.1 for a definition), $\lambda_1' = \lambda_1/\rho_1$ and λ_2' is the vector $\lambda_2'^T = \left(\frac{\lambda_2^{(V)}}{rho_1} 1_{2q}^T \quad \frac{\lambda_2^{(I)}}{\rho_1} 1_{2s}^T \quad \frac{\lambda_2^{(A)}}{\rho_1} 1_{2s+q}^T \right)$, with 1_a being the unit vector of length a.

Friedman et al. (2007) proved that for fused-type lasso optimization problems such as Equation (2.7), once the solution for $\lambda_1' = 0$, $\hat{z}(0, \lambda_2')$, is known, the general solution for $\lambda_1 > 0$ is given by a soft-tresholding operation, namely,

$$\hat{z}(\lambda_1', \lambda_2') = S_{\lambda_1'}(\hat{z}(0, \lambda_2'))\mathrm{sign}(\hat{z}(0, \lambda_2'))(|\hat{z}(0, \lambda_2')| - \lambda_1')_+ .$$

The optimization problem that needs to be solved is then

$$\arg\min_{z} \left\{ \frac{1}{2}||z-b||_2^2 + ||F(\lambda_2' \circ z)||_1 \right\} ,$$

which requires an inner ADMM procedure. The vectors v and t are initialized with zero entries, and the quantities z, v, t are updated with the following three steps:

i $z^{m+1} = (I + \rho_2 F^T F)^{-1}[b + \rho_2 F^T(v^m - t^m)]$;

ii $v^{m+1} = S_{\lambda_2'/\rho_2}(Fz^{m+1} + t^m)$;

iii $t^{m+1} = t^m + Fz^{m+1} - v^{m+1}$.

The algorithm stops when the primal and dual residuals are both below the given numerical precision treshold (Boyd et al., 2011).

The implementation given in the R package `pdglasso` (Ranciati & Roverato, 2024b) achieves better efficiency for the algorithm by making the step sizes ρ_1 and ρ_2 adaptive, i.e. their value is updated at every step to speed up convergence, by reducing the dimensions of the matrix F depending on the pdRCON submodel class of interest, and by exploiting the sparsity of F when encoding the results of the products involving it.

Chapter 3

Simulation study

This chapter presents the results of a simulation study regarding the role played by the across-graph association in the context of pdRCON models, and how it affects inference on the left and right blocks. The study was performed in R and made use of the `pdglasso` (Ranciati & Roverato, 2024a) package, and the `tidyverse` (Wickham et al., 2019), `ggplot2` (Wickham, 2016), `see` (Lüdecke et al., 2021) packages were used to analyse and visualise the results. The `R` code is available in the Appendix C. The scenarios considered here will have $p = 20$ variables; the results of a similar simulation study but in a higher dimensionality setting, having 50 variables, can be found in Appendix D.

3.1 Objective of the study

The objective of the study is to determine whether a reduction of the number of the parameters in the model to be estimated obtained by making the assumption of no across-block edges or the assumption of full across-block symmetry in the graph $\mathcal{G}$ leads to a substantial improvement in the quality of the inference on the left and right sub-graphs, $\mathcal{G}_L$ and $\mathcal{G}_R$.

If that is the case, such assumptions could then be useful in situations where the across-graph association structure is of no interest, the entries of the concentration matrix defining it are regarded as nuisance parameters, only the left and right blocks of Θ, Θ_{LL} and Θ_{RR} are the parameters of interested so disregarding across-graph edges to focus on recovering the structure of the two sub-graphs might be attractive.

There are two aspects to *coloured graph recovery*: the overall edge structure of the underlying skeleton graph, i.e. whether an edge is present or missing, and the symmetric structure, i.e. whether an edge is coloured or not. The only symmetries investigated here are the ones involving pairs of non-missing edges; it could be of interest to evaluate also the presence of symmetries due to pairs of missing edges, but since pdRCON models are often deployed in high-dimensionality, relatively low edge-density frameworks, the first type of symmetries are considered.

3.2 The pdRCON without across-block edges submodel class

The additional pdRCON submodel class considered in the simulation study is obtained by redefining the penalization term $\mathcal{P}_{\lambda_1}(\Theta)$ in Equation (2.3), which is the penalization term of the standard graphical lasso and thus penalizes equally every entry of the concentration matrix Θ, in such a way that the left and right block Θ_{LL} and Θ_{RR} are penalized differently from the block defining the across-group structure, Θ_{LR}. The new penalization term is

$$\mathcal{P}_{\lambda_1} = \lambda_1(||\Theta_{LL}||_1 + ||\Theta_{RR}||_1) + \lambda_1^{across}||\Theta_{LR}||_1 \ , \tag{3.1}$$

and the *pdRCON without across-block edges* submodel class is given by imposing $\lambda_1^{across} = \infty$. In this way, all entries of Θ_{LR}, and of $\Theta_{RL} = \Theta_{LR}^T$, are forced to be equal to zero in the solution; Θ_{LL} and Θ_{RR}, on the other hand, have their ℓ_1 norm penalized with the amount of penalization implied by $\lambda_1 < \infty$, regulating for the amount of sparsity of the edges in the two subgraphs $\mathcal{G}_L$ and $\mathcal{G}_R$. The model selection procedure for this new submodel class works in the same way as described in Section 2.7.3, selecting first the optimal value of λ_1 in the grid, and then the optimal value of λ_2 conditioned on the value of λ_1 selected in the first step.

3.3 Framework of the simulation study

The true model the data was generated from was a pdRCON model using the function `pdRCON.simulate()`, having parameters

- `p`, the number of variables;
- `dens`, the density of uncoloured edges throughout the whole graph;
- `dens.vertex`, the proportion of coloured vertices among the `p` vertices;
- `dens.inside` the density of coloured inside-block vertices;
- `dens.across` the density of coloured across-block vertices.

Three coloured graph scenarios were considered, in order to determine whether the effects of different across-group association structure hypotheses on the inference over the two groups differ depending on the edge density and the number of symmetries in the graph. These were:

- a high density, 50% symmetric setting, having `p` = 20, `dens` = 0.5, `dens.vertex` = 0.5, `dens.inside` = 0.25, `dens.across` = 0.25;
- a high density, fully symmetric setting, having `p` = 20, `dens` = 0, `dens.vertex` = 1, `dens.inside` = 0.5, `dens.across` = 0.5;
- a low density, low symmetry setting, having `p` = 20, `dens` = 0.2, `dens.vertex` = 0.1, `dens.inside` = 0.1, `dens.across` = 0.1.

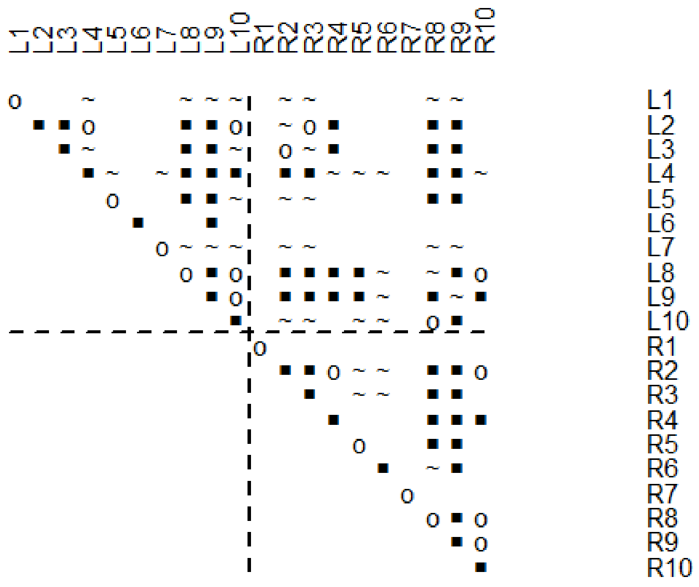


FIGURE 3.1: Example of a true coloured graph the data was generated from in the high density, 50% symmetric setting. ■ denotes a parametric symmetry, ○ a structural (but not parametric) symmetry, $\sim$ a structural asymmetry and an empty cell a missing edge.

An example of a true coloured graph in the high density, 50% symmetric setting is given in Figure 3.1. At each step of the simulation, for each true pdRCON model, three models, having decreasing number of parameters to be estimated, are selected based on the simulated dataset:

1. a pdRCON model allowing, but not forcing, vertex, inside-block and across block symmetries;

2. a pdRCON model allowing vertex and inside-block symmetries and forcing all across-block edges to be symmetric;

3. a pdRCON model allowing vertex and inside-block symmetries and having no across-block edges.

The three selected coloured graphs based on the dataset generated from the example of a coloured graph of Figure 3.1, with sample size $N = 100$, are given in Figure 3.2, 3.3, 3.4 respectively. It can be observed that the models selected have lower densities than the true model, and different assumptions on the across-group structure lead to selected coloured graphs that differ also in the left and right blocks.

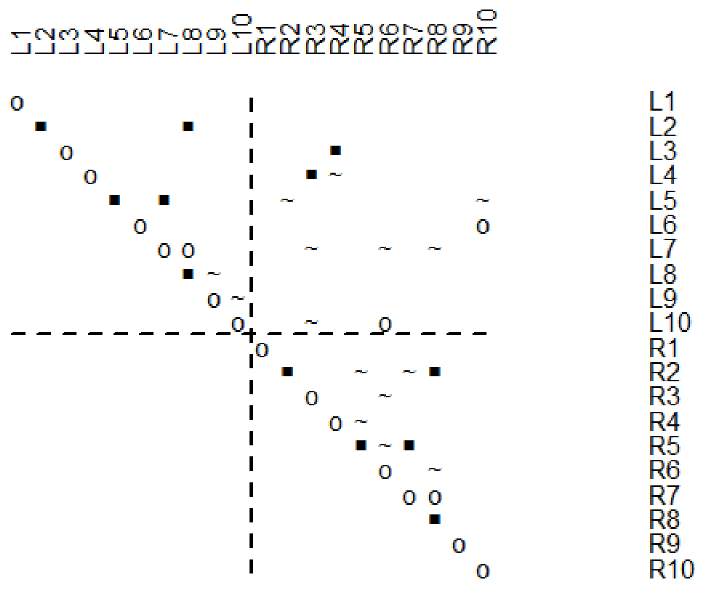


FIGURE 3.2: pdRCON model allowing for vertex, inside-block and across-block symmetries selected by the BIC on a simulated dataset from the coloured graph of Figure 3.1.

The selection procedure for each model uses a grid of 20 equally spaced on the logarithmic scale values of λ_1 from $\frac{\lambda_1^{\text{diag}}}{20}$ to λ_1^{diag} and a grid of 20 equally spaced on the logarithmic scale values of λ_2 from $\frac{\lambda_2^{\text{sym}}}{20}$ to λ_2^{sym}, and BIC is used to compare the models. For each sample size value, ten true pdRCON models are randomly generated, and for each true pdRCON model the measures defined in Section 3.4 are computed on the three selected models from the aforementioned pdRCON submodel classes, and then the scores obtained for the measures over the ten iterations are averaged, and their standard deviations are computed to account for variability.

A number of iterations larger than ten would be preferable, as the results exhibit large variability, but the computational burden of the model selection procedure is an

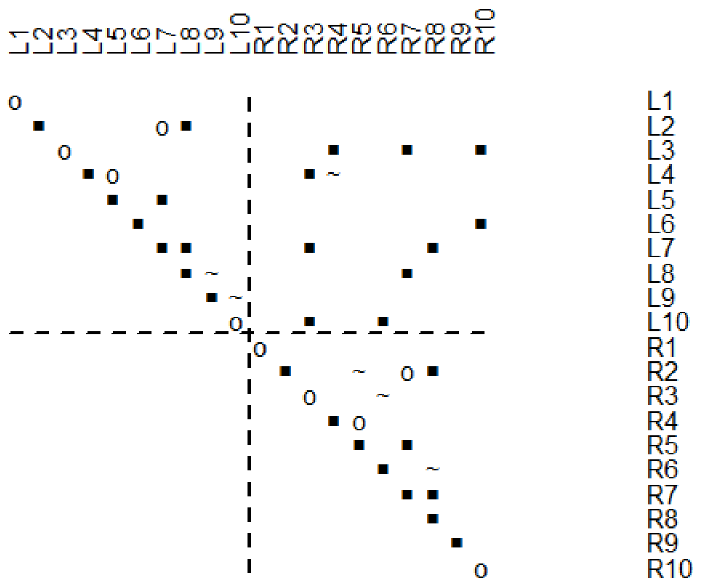


FIGURE 3.3: pdRCON model allowing for vertex and inside-block symmetries, and forcing across-block symmetries selected by the BIC on a simulated dataset from the coloured graph of Figure 3.1.

obstacle even when working in a relatively low dimensionality setting, such as in this case with $\mathtt{p} = 20$.

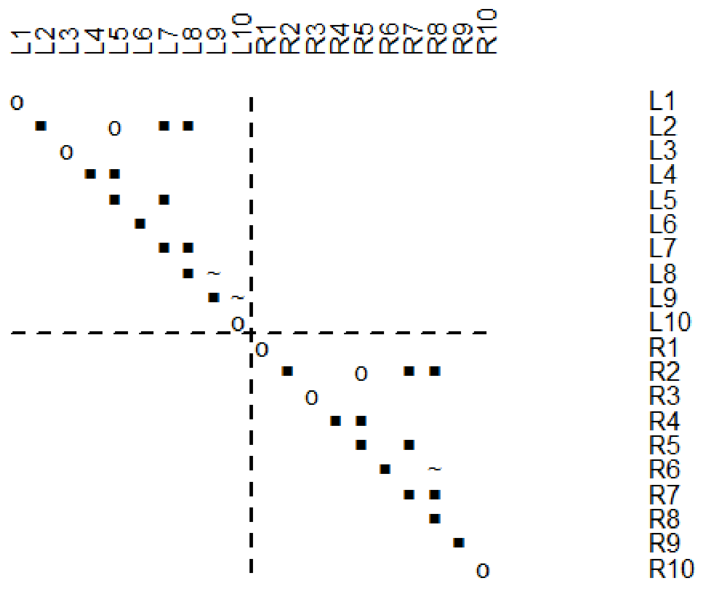


FIGURE 3.4: pdRCON model allowing for vertex and inside-block symmetries, and not allowing across-block edges selected by the BIC on a simulated dataset from the coloured graph of Figure 3.1.

3.4 Performance measures

3.4.1 Graph recovery measures

In order to compare the performance of the models in recovering the structure of the subgraphs $\mathcal{G}_L$ and $\mathcal{G}_R$ some well-established measures were considered:

$$\text{ePPV} = \frac{\text{eTP}}{\#\text{edges}}\,, \quad \text{eTPR} = \frac{\text{eTP}}{\text{eP}}\,, \quad \text{eTNR} = \frac{\text{eTN}}{\text{eN}}\,,$$

$$\text{eF1} = 2 \times \frac{\text{ePPV} \times \text{eTPR}}{\text{ePPV} + \text{eTPR}} = 2 \times \frac{\text{eTP}}{2 \times \text{eTP} + \text{eFP} + \text{eFN}}\,,$$

$$\text{eMCC} = \frac{\text{eTP} \times \text{eTN} - \text{eFP} \times \text{eFN}}{\sqrt{(\text{eTP}+\text{eFP})(\text{eTP}+\text{eFN})(\text{eTN}+\text{eFP})(\text{eTN}+\text{eFN})}}\,,$$

where

- ePPV is the edge Positive Predicted Value, or precision, defined as the ratio between the number of true, correctly identified edges, eTP, i.e. the edges in the selected left and right subgraphs that correspond to edges in the true subgraphs, and the number of edges in the selected subgraphs, #edges. In the second reformulation, eFP is the number of false edges, i.e. the edges in the selected left and right subgraphs not corresponding to edges in the true subgraphs, and eFN is the number of false missing edges, i.e. edges in the true subgraphs not recovered by the selected subgraphs;

- eTPR is the edge True-Positive Rate, defined as the ratio between eTP and the number of edges in the true subgraphs, eP;
- eTNR is the edge true-negative rate, defined as the ratio between the true, correctly identified missing edges (eTN), i.e. the missing edges in the selected subgraphs that correspond to missing edges in the true subgraphs, and the number of missing edges in the true subgraphs (eN);
- eF1 is the edge F1 score, which is a summary measure defined as the harmonic mean of ePPV and eTPR, evaluating the overall accuracy of the selected subgraphs, particularly used in sparse graph scenarios;
- eMCC is the Matthews Correlation Coefficient, function of the true and false edges and of the true and false missing edges, ranging in value from -1 to 1, where -1 is associated with a completely inverse identification of edges and non-edges, 0 with a performance equal to guessing at random, and 1 with perfect graph recovery.

3.4.2 Symmetry recovery measures

Analogously, the measures considered to assess the performance in recovering the symmetric structure of $\mathcal{G}_L$ and $\mathcal{G}_R$ were

$$\mathrm{sTPR} = \frac{\mathrm{sTP}}{\mathrm{sP}}\ , \quad \mathrm{sTNR} = \frac{\mathrm{sTN}}{\mathrm{sN}}\ , \tag{3.2}$$

where

- sTPR is the symmetry True-Positive Rate, defined as the ratio between the number of true, correctly identified, symmetries, sTP, i.e. the symmetries in the selected left and right subgraphs that correspond to symmetries in the true subgraphs, and the number of symmetries in the true subgraphs, sP;
- sTNR is the symmetry true-negative rate, defined as the ratio between the true parametric asymmetries, sTN, i.e. the parametric asymmetries in the selected subgraphs that correspond to parametric asymmetries in the true subgraphs, and the number of parametric asymmetries in the true subgraphs, sN.

3.4.3 Measures of the accuracy of the estimated concentration matrix

Other than the recovery of the coloured graph, interest is also in the accuracy of the estimates of the left and right block of the concentration matrix, Θ_{LL} and Θ_{RR}.

The two measures considered were the Frobenius norm error and the entropy loss (Dey & Srinivasan, 1985), summed for left and right block respectively:

$$\text{Frobenius norm error}(\hat{\Theta}) = ||\Theta_{LL} - \hat{\Theta}_{LL}||_F + ||\Theta_{RR} - \hat{\Theta}_{RR}||_F \ ,$$

$$\begin{aligned}\text{Entropy loss}(\hat{\Theta}) =& (\text{tr}(\Sigma_{LL}\hat{\Theta}_{LL}) - \log\det(\Sigma_{LL}\hat{\Theta}_{LL}) - q) + \\ & (\text{tr}(\Sigma_{RR}\hat{\Theta}_{RR}) - \log\det(\Sigma_{RR}\hat{\Theta}_{RR}) - q) \ ,\end{aligned}$$

where $\Sigma = \Theta^{-1}$ is the true covariance matrix. The Frobenius norm error is a distance, an element-by-element discrepancy measure with a geometric interpretation, while the entropy loss is a scale-invariant loss function, thus focusing on structural inaccuracies in the estimation of the concentration matrix, rather than the absolute values of the singular concentrations, and has a probabilistic interpretation, it measures how well $\hat{\Theta}$ estimates the inverse of the covariance matrix of a multivariate normal distribution. The more accurate the estimate, the closer $S\hat{\Theta}$ will be to the identity matrix.

3.5 Results

3.5.1 High density, 50% symmetric setting

The sample size values considered were $N = \{20, 40, 60, 80, 100, 150, 200\}$. Figure 3.5 shows that no model has a clear edge in terms of structure recovery of $\mathcal{G}_L$ and $\mathcal{G}_R$, and the variability of the results, visualized here through vertical bars depicting ± 1 standard deviation from the mean across the ten iterations, is quite high, although the pdRCON model forcing across-block edges to be symmetric seems to perform consistently slightly better on average than the other two in the F1 score. Similarly, Figure 3.6 shows that there are not large differences in recovering the symmetric structure of $\mathcal{G}_L$ and $\mathcal{G}_R$ either, although the model forcing across-block edges to be symmetric estimates a larger amount of symmetries in the two subgraphs (Figure 3.8). The two estimation accuracy measures in Figure 3.7 show different results: allowing across-block symmetries leads

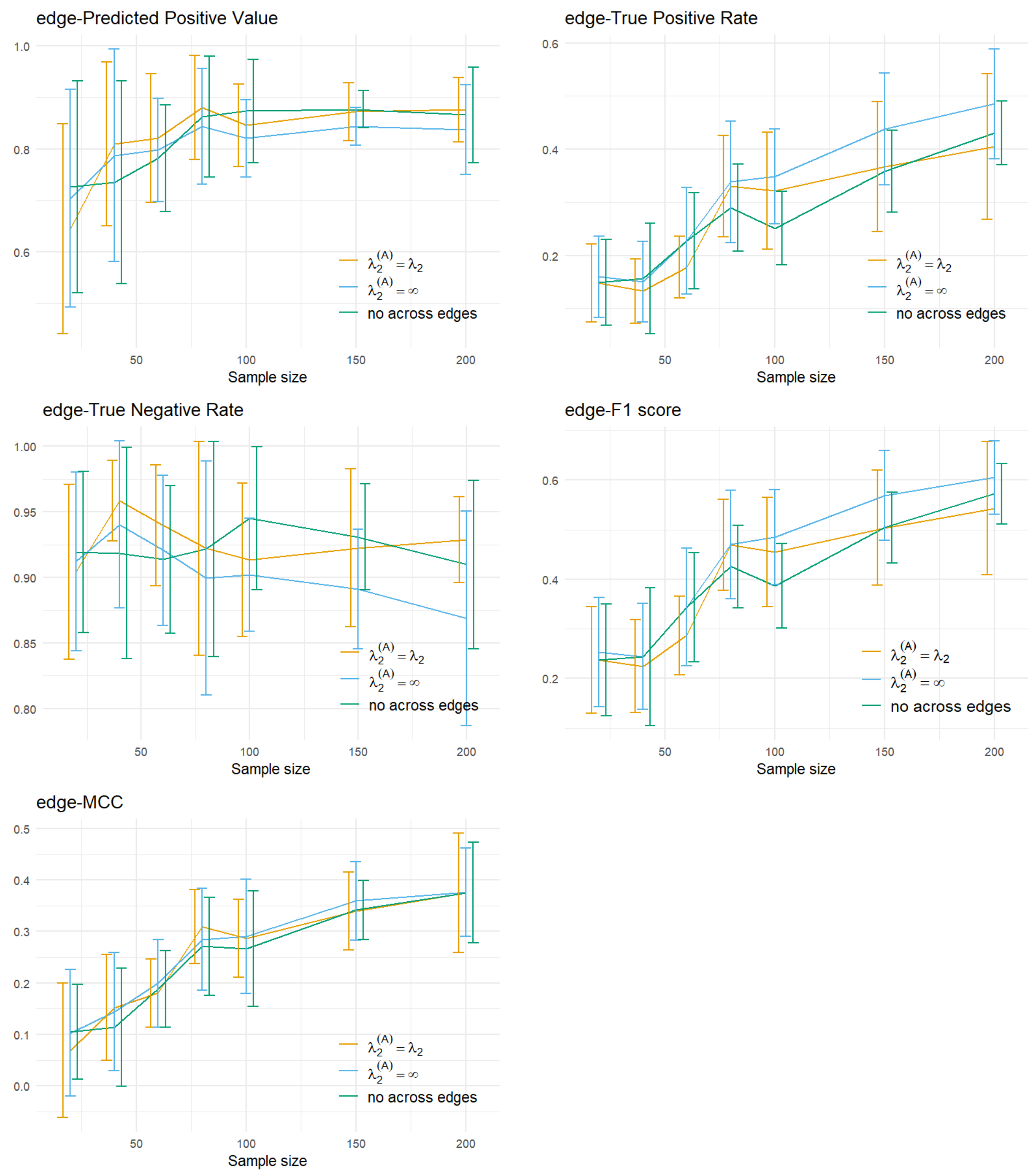


FIGURE 3.5: Structure recovery measures of $\mathcal{G}_L$ and $\mathcal{G}_R$ for the high density, 50% symmetric setting. Vertical bars denote ± 1 standard deviation across the 10 iterations.

to a better estimate in terms of Frobenius norm error, while the no across-block edges hypothesis leads to a smaller entropy loss.

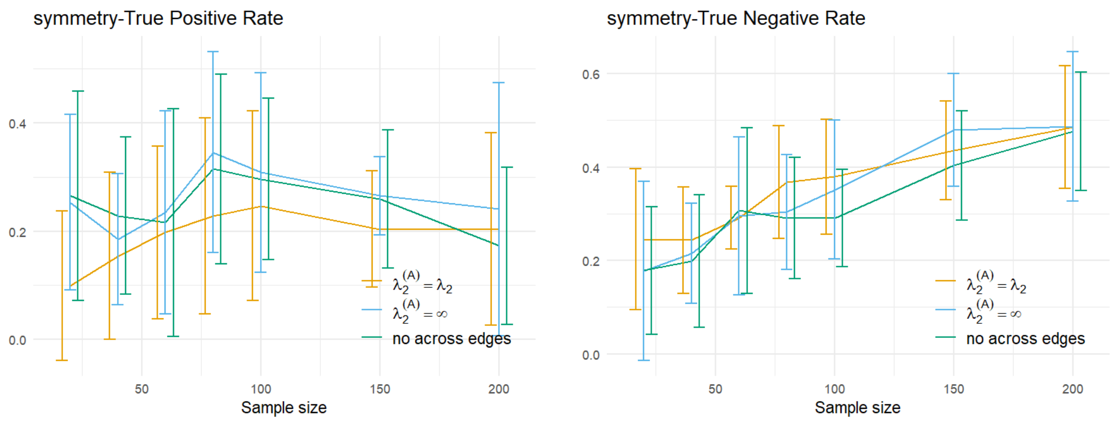


FIGURE 3.6: Symmetric structure recovery measures of $\mathcal{G}_L$ and $\mathcal{G}_R$ for the high density, 50% symmetric setting.

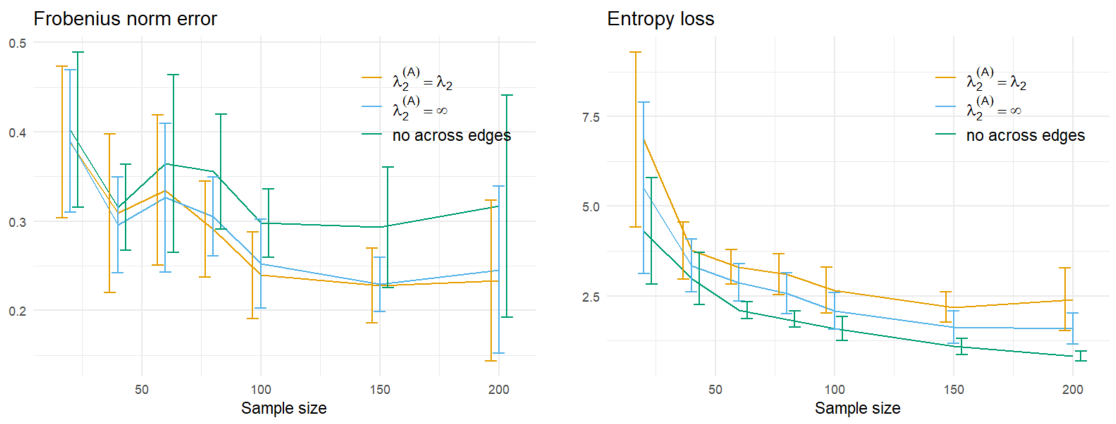


FIGURE 3.7: Estimation accuracy measures of Θ_{LL} and Θ_{RR} for the high density, 50% symmetric setting.

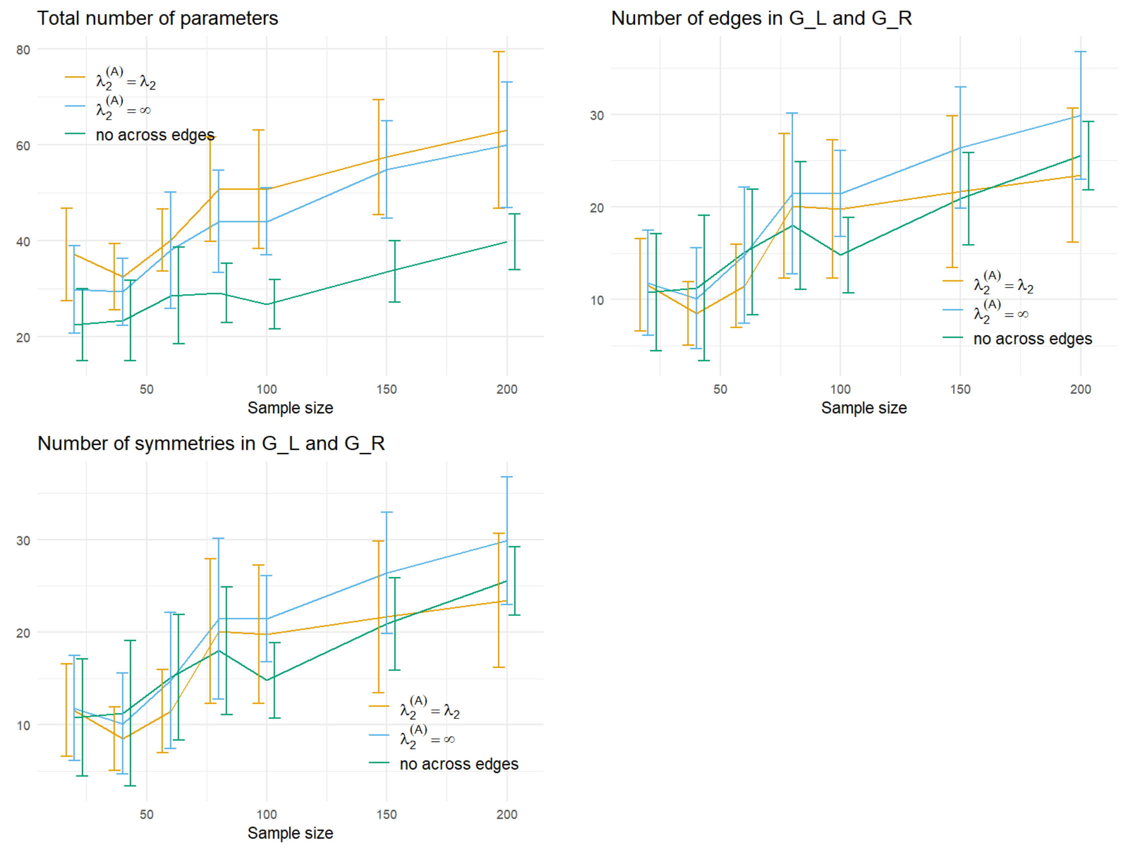


FIGURE 3.8: Total number of parameters estimated and numbers of edges and non-zero parametric symmetries in $\mathcal{G}_L$ and $\mathcal{G}_R$ for the high density, 50% symmetric setting.

3.5.2 High density, fully symmetric setting

Another scenario considered is with a true coloured graph being an instance of a fully symmetric model, having every edge and vertex coloured. An example of a model the data was generated from is given in Figure 3.9.

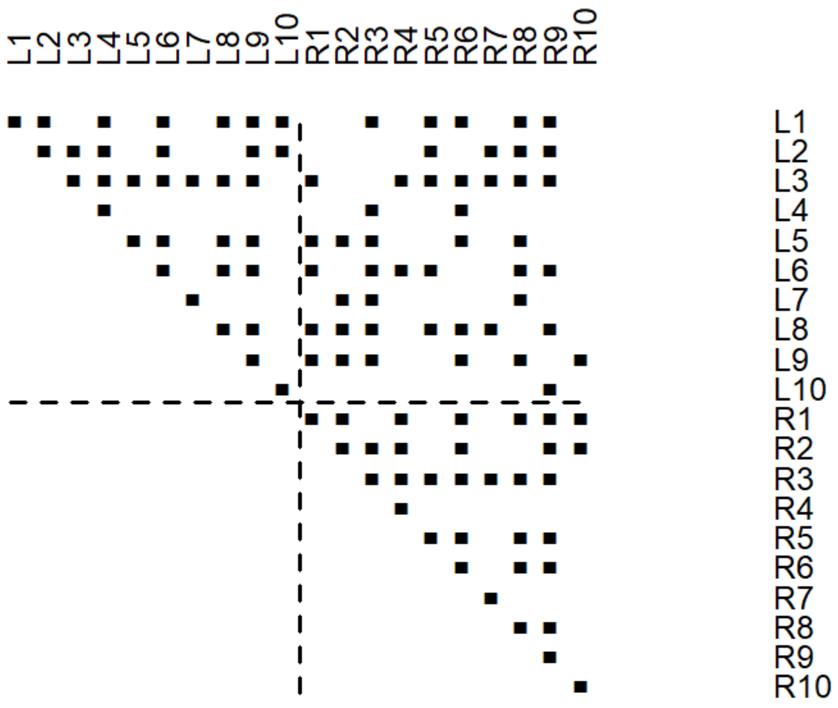


FIGURE 3.9: Example of a true coloured graph the data was generated from, in the high density, fully symmetric setting.

Compared to the prior scenario, with similar graph density but lower symmetry, the pdRCON model forcing across-group symmetries has a more distinct edge in recovering the graph structure of $\mathcal{G}_L$ and $\mathcal{G}_R$, performing better in both the F1 score and Matthews Correlation Coefficient (Figure 3.10). Some values for the ePPV and the eMCC in the low sample sizes are missing because it sometimes happens that the model selected in an iteration has no edges, resulting in a `NaN` value for those measures. Figure 3.11 shows that the symmetry-wise True Positive Rate is also higher for the model forcing across-group symmetries, although all three models are comparable in terms of estimation accuracy (Figure 3.12). Remarkably, it can be observed from Figure 3.13 that the total number of parameters estimated is actually higher when forcing across-group symmetries: the reduction of the model in the across-group structure is more than compensated

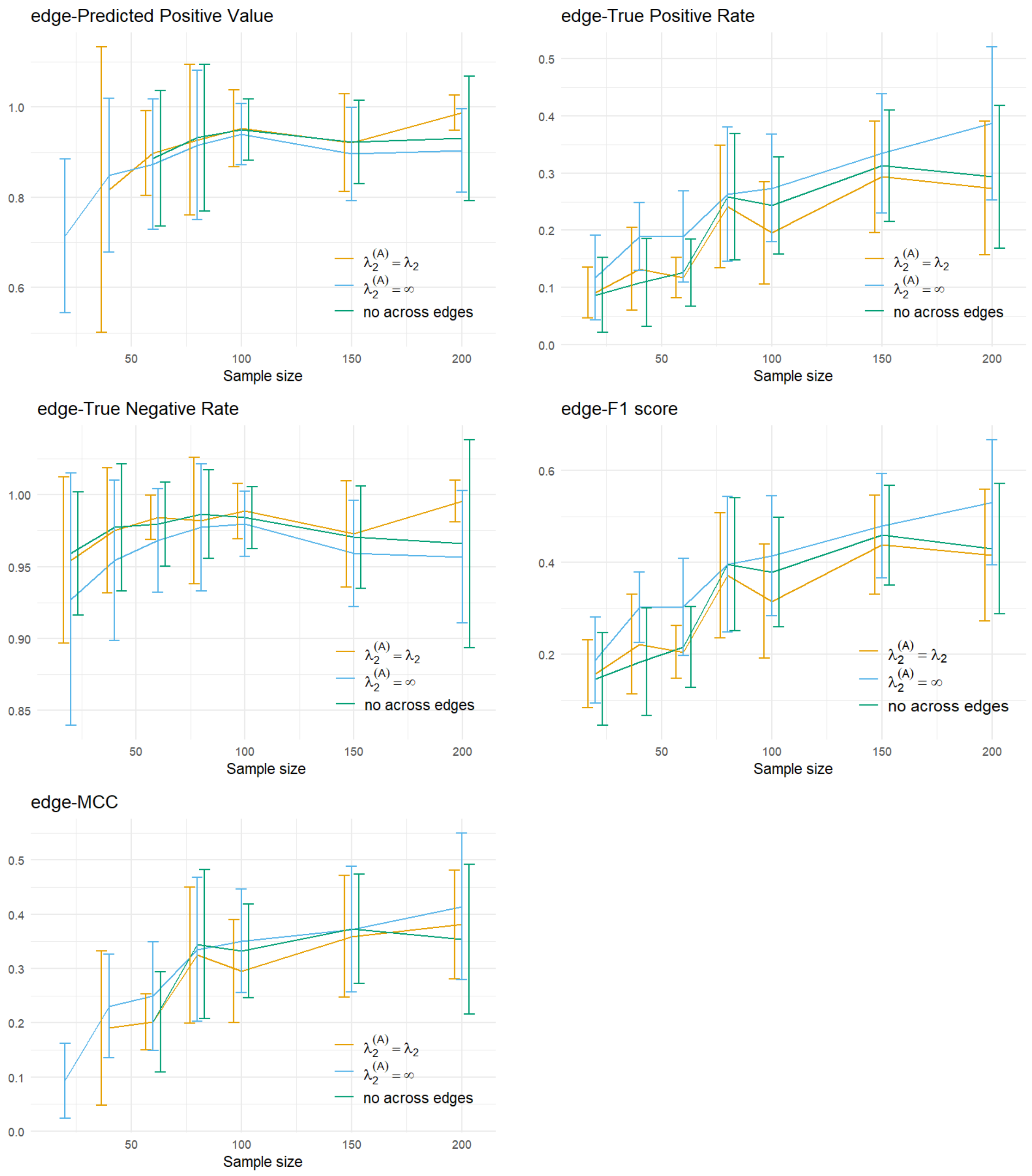


FIGURE 3.10: Structure recovery measures of $\mathcal{G}_L$ and $\mathcal{G}_R$ for the high density, fully symmetric setting

by an increase of the number of both edges and non-zero parametric symmetries estimated in the left and right groups. The coloured subgraphs $\mathcal{G}_L$ and $\mathcal{G}_R$ obtained are both denser and have more information about the symmetric structure.

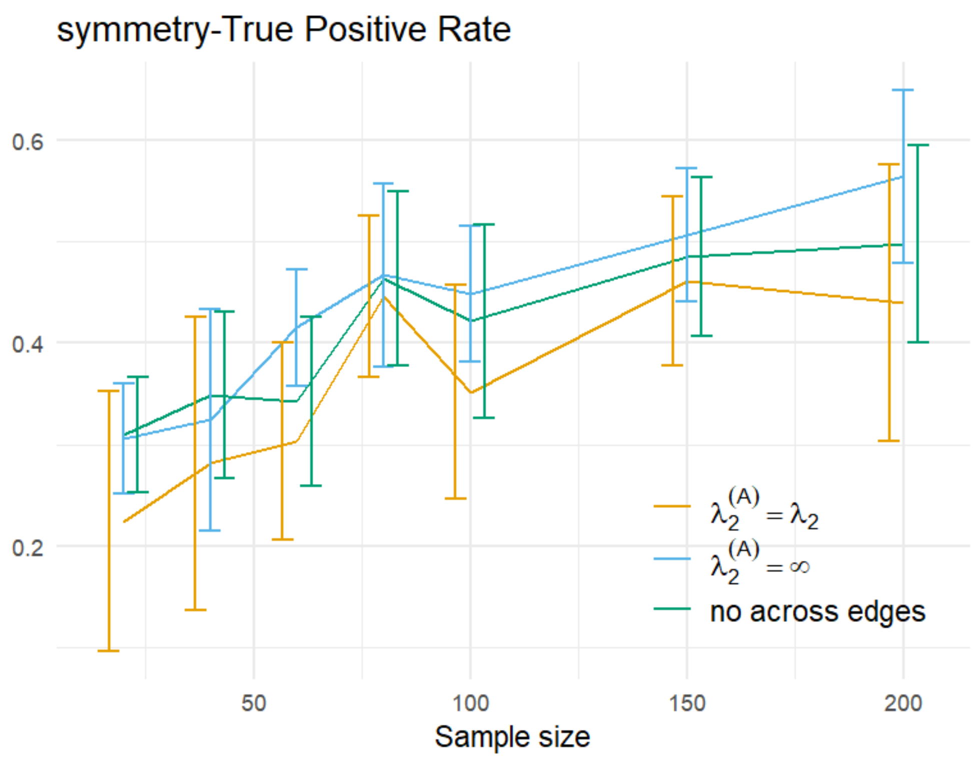


FIGURE 3.11: Symmetry-True Positive Rate of $\mathcal{G}_L$ and $\mathcal{G}_R$ for the high density, fully symmetric setting

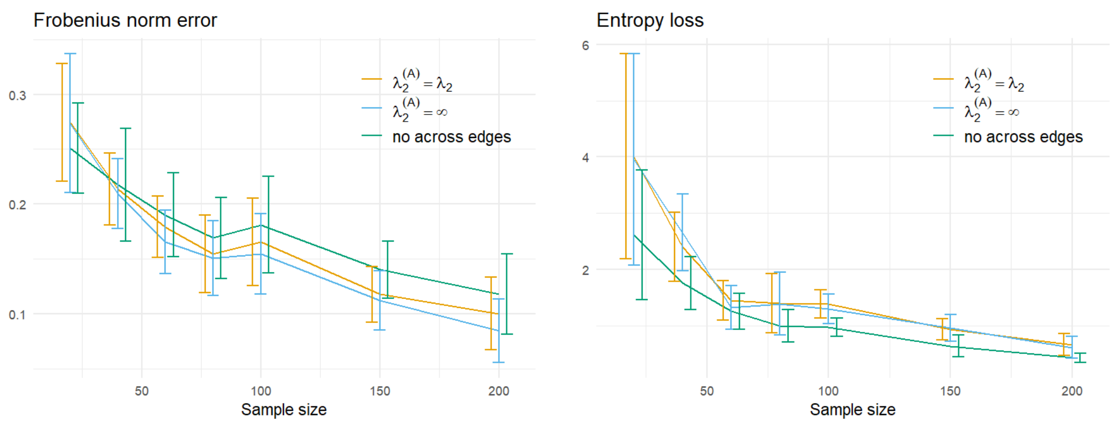


FIGURE 3.12: Estimation accuracy measures of Θ_{LL} and Θ_{RR} for the high density, fully symmetric setting

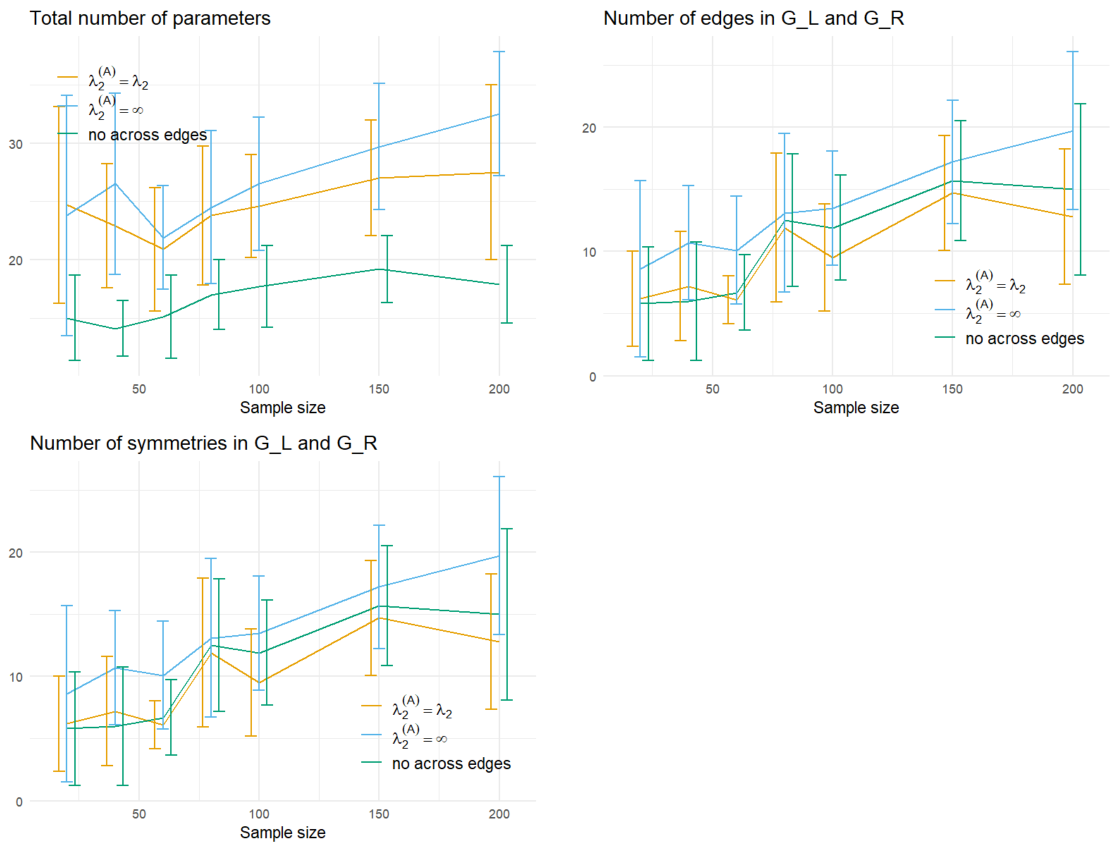


FIGURE 3.13: Total number of parameters estimated in the models and numbers of edges and non-zero parametric symmetries in $\mathcal{G}_L$ and $\mathcal{G}_R$ for the high density, fully symmetric setting

3.5.3 Low density, low symmetry setting

Lastly, the lower density scenario was considered. An example of a model the data was generated from is given in Figure 3.14.

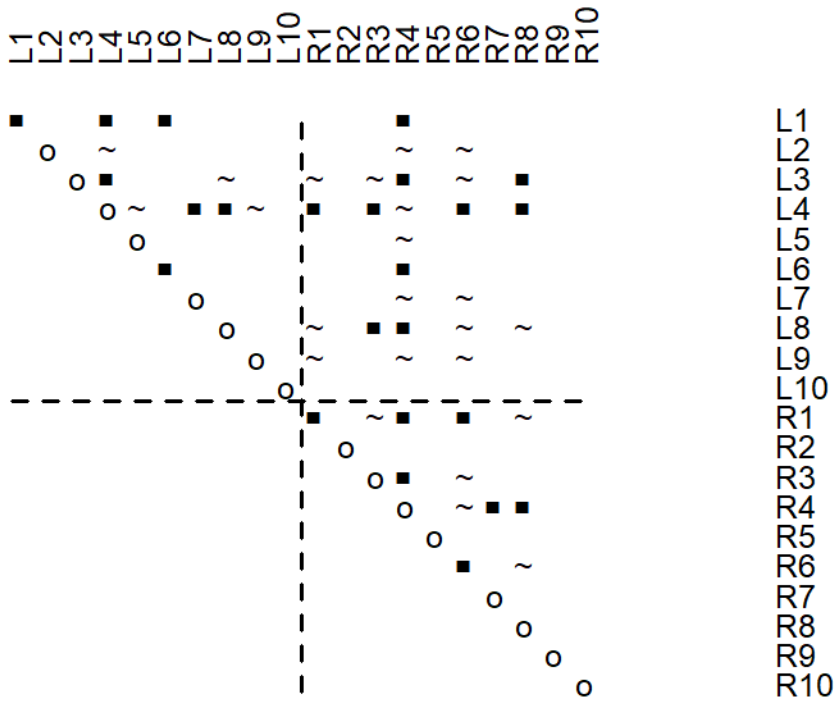


FIGURE 3.14: Example of a true coloured graph the data was generated from, in the low density, low symmetry setting.

In this scenario differences between the three submodel classes are more opaque. Once again, some values of ePPV and eMCC are missing when the sample size is low. The models with $\lambda_2^{(A)} = \infty$ and $\lambda_2^{(A)} = \lambda_2$ have very similar Frobenius norm error (Figure 3.17), but the F1 score, which is a more useful metric in low density settings since predicting well non-missing edges is more important than predicting missing edges, is generally slightly higher when across-group edges are forced to be symmetric (Figure 3.15). Moreover, it also has an advantage in terms of edge and symmetry-True Positive Rate (Figure 3.16) when for larger sample sizes. The model with $\lambda_2^{(A)} = \infty$ is also the one estimating the most amount of edges and symmetries in the two subgraphs (Figure 3.18).

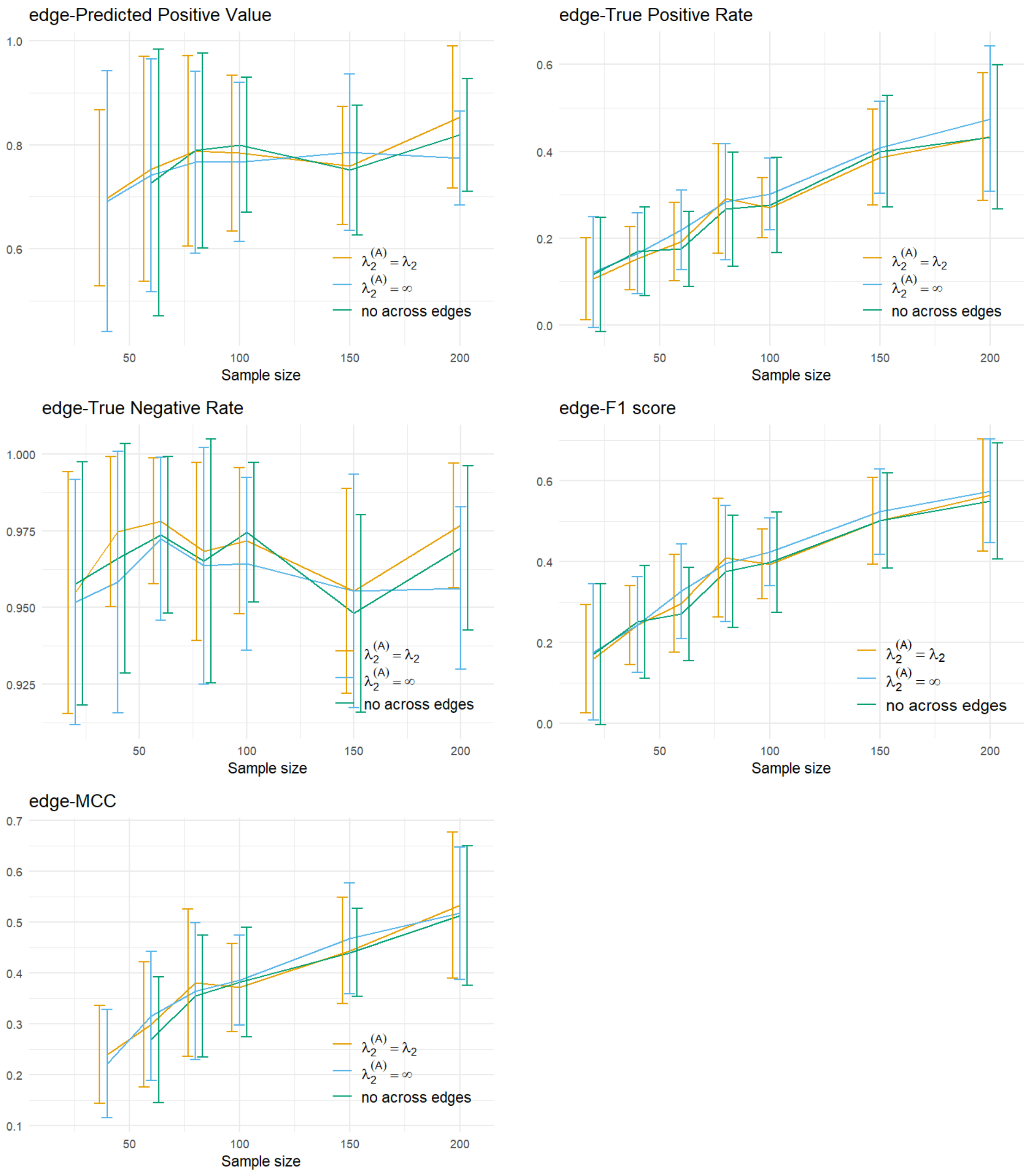


FIGURE 3.15: Structure recovery measures of $\mathcal{G}_L$ and $\mathcal{G}_R$ for the low density, low symmetry setting.

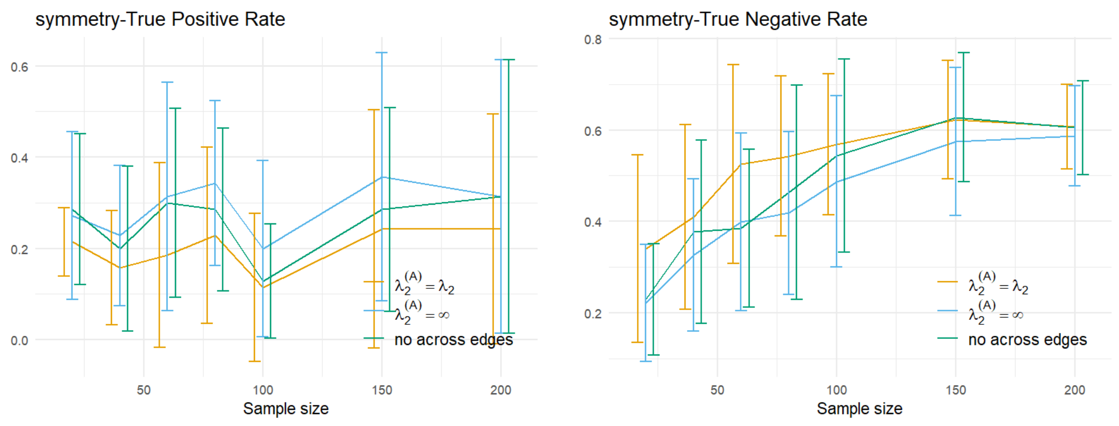


FIGURE 3.16: Symmetric structure recovery measures of $\mathcal{G}_L$ and $\mathcal{G}_R$ for the low density, low symmetry setting.

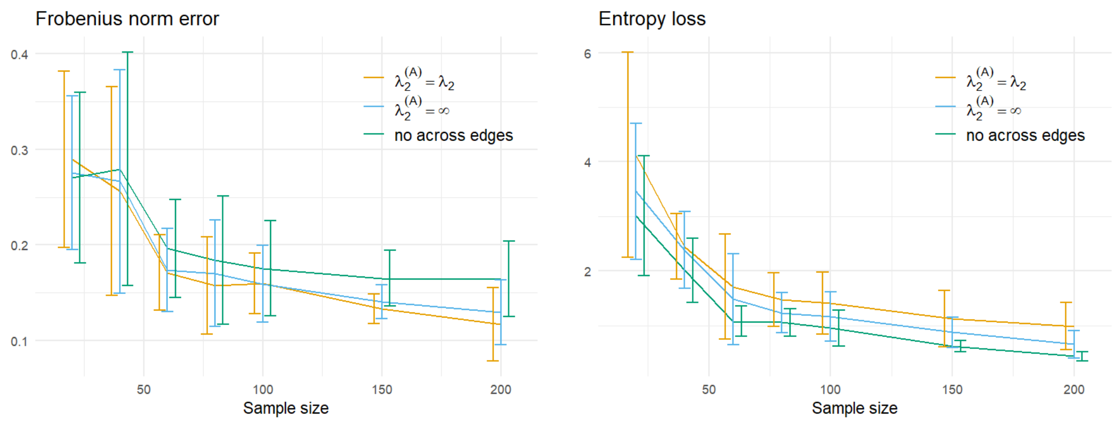


FIGURE 3.17: Estimation accuracy measures of Θ_{LL} and Θ_{RR} for the low density, low symmetry setting.

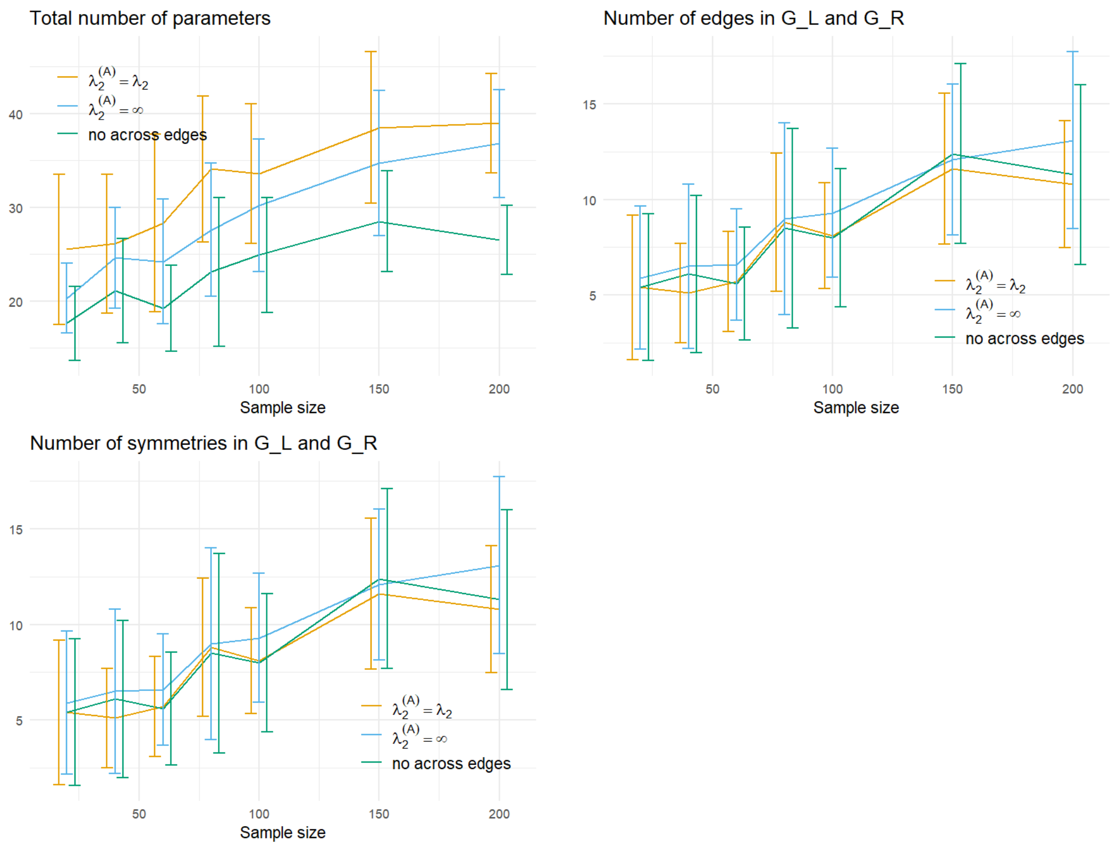


FIGURE 3.18: Total number of parameters estimated in the models and numbers of edges and non-zero parametric symmetries in $\mathcal{G}_L$ and $\mathcal{G}_R$ for the low density, low symmetry setting.

Chapter 4

Breast cancer data analysis

This chapter presents an analysis of the breast cancer gene expression data considered in Ranciati & Roverato (2024a) with the three pdRCON submodel classes considered in Chapter 3, and how the models obtained differ based on the assumption about the across-group structure.

4.1 Problem description

The data is the gene-level transcription estimates through a $\log_2(Y+1)$ transformation of the normalized counts Y, quantifying how much of a particular gene is transcribed into RNA. $N = 114$ individuals took part in the study, and for each individual a sample of breast cancer tissue was examined, along with a sample of healthy adjacent tissue. This is an instance of a paired data problem, where the $p = 178$ genes can be naturally split into two groups each one comprised of $q = 89$ genes, describing the breast cancer sample and the control one respectively. Interest is in the estimation of a graphical model that can represent the conditional dependence structure between the transcription estimates of the genes; a pdRCON model can also indentify symmetries and asymmetries involving the association network of the breast cancer tissue and the healthy tissue, which can provide particularly useful insights.

Moreover, it is plausible that the interest lies in the two groups' subgraphs and how they differ, while the across-group structure is a nuisance element. The three submodel classes of Chapter 3, having different assumptions about the across-group association structure, will be considered and the models obtained compared.

4.2 Model selection with eBIC

The model selection procedure considered is the same grid search with 20 equally spaced on the logarithmic scale values for both parameters described in Section 3.3.

Figure 4.1 shows the coloured graph selected within the subclass of pdRCON models allowing, but not forcing, vertex, inside-block and across-block symmetries. It can be oberserved how the subgraph associated with the tissue affected by cancer, which is the left block of the coloured graph, has a denser network, having 104 edges compared to 56 for the right block, corresponding respectively to a subgraph density of 2.7% and 1.4%, suggesting a higher number of interactions between genes in cancer cells compared to healthy tissue, which is a well established fact (see e.g. Goh et al., 2007). The graph has low density (1.07%) and no parametric symmetries are estimated, although there are 17 structural inside-block symmetries due to pairs of non-missing edges. Moreover, 12 out of the 17 structural inside-block symmetries involve the gene LRP2.

Figure 4.2 and 4.3 show that in this case, the left and right coloured subgraphs selected are exactly the same for all three subclasses, and the left and right blocks of the estimates of the concentration matrix have Frobenius norm differences close to zero. When dimensionality is high and the number of across-block edges is small, the assumptions on the across-group structure seem to have little effect on the inference over the two groups.

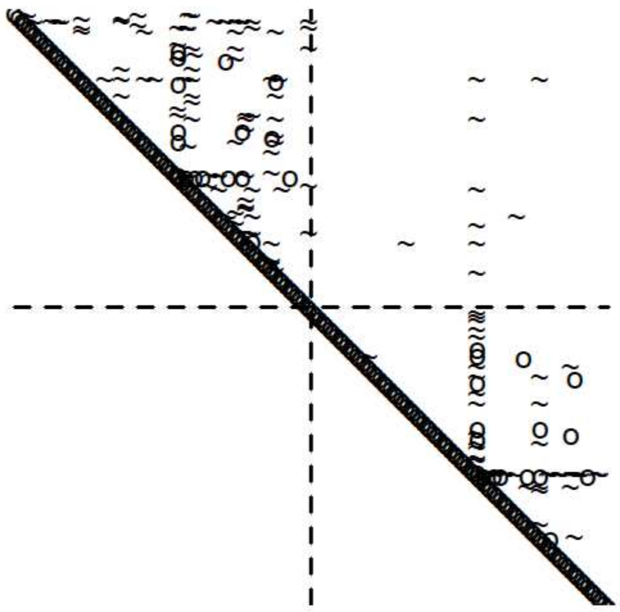


Figure 4.1: pdRCON allowing vertex, inside-block, and across-block symmetries selected by the eBIC having $\gamma = 0.5$.

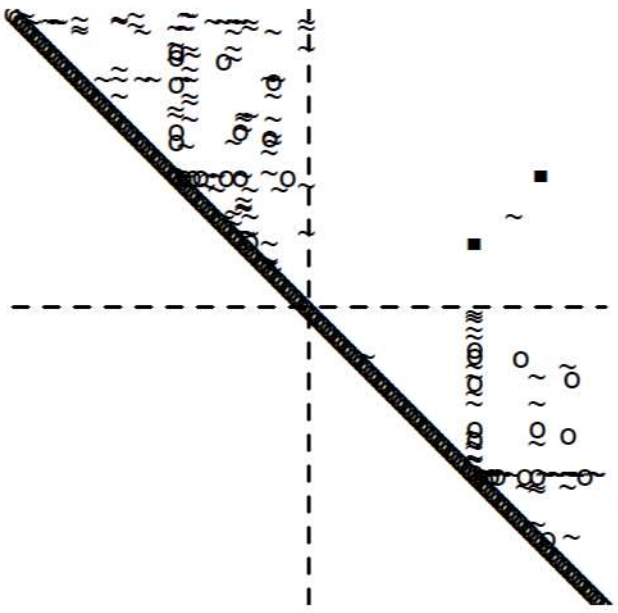


FIGURE 4.2: pdRCON allowing vertex and inside-block symmetries and forcing across-block symmetries selected by the eBIC having $\gamma = 0.5$.

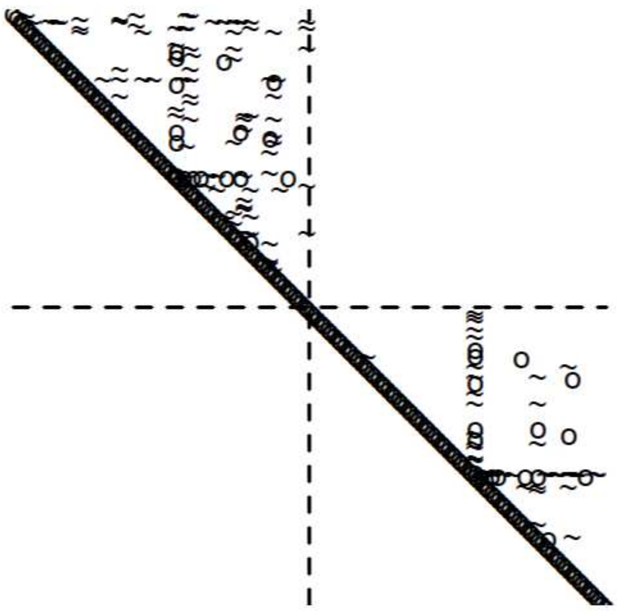


FIGURE 4.3: pdRCON allowing vertex and inside-block symmetries and forcing across-block edges to be missing selected by the eBIC having $\gamma = 0.5$.

4.3 Model selection with BIC

When using BIC in the model selection procedure, the pdRCON models obtained are considerably less sparse, and some differences between submodel classes arise even in

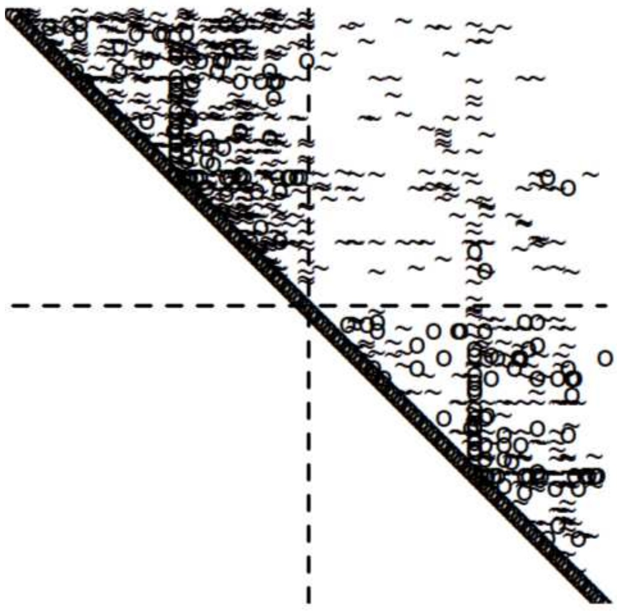


FIGURE 4.4: pdRCON allowing vertex, inside-block, and across-block symmetries selected by the BIC.

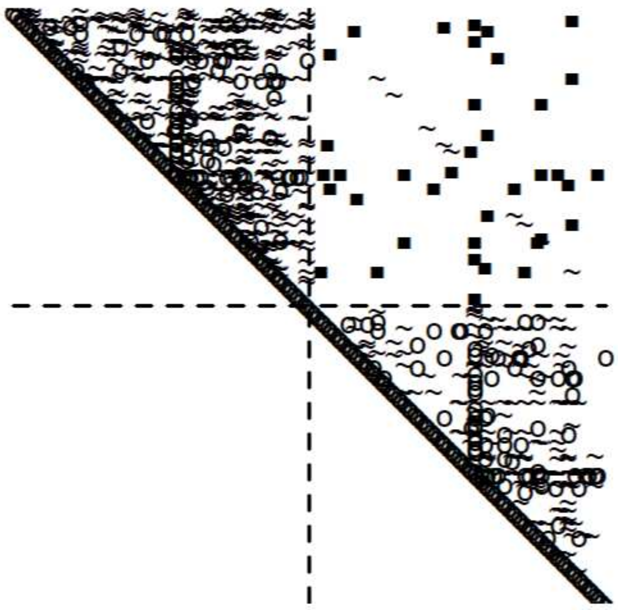


FIGURE 4.5: pdRCON allowing vertex and inside-block symmetries and forcing across-block symmetries selected by the BIC.

the left and right blocks. The model forcing across-block symmetries has less across-block edges than the model simply allowing across-block symmetries, 47 compared to 109; differences between these two models in the left and right blocks are however very small, with $\approx 99\%$ of inside-block edges being shared. The model forcing across-block edges to be missing is more parsimonious, with an overall graph density of 2.58% compared to $\approx 5\%$ for the previous two coloured graphs, and an inside-block density of

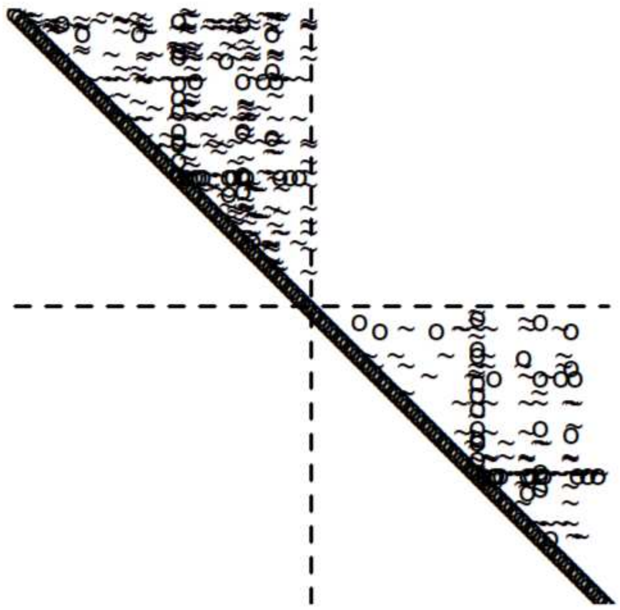


FIGURE 4.6: pdRCON allowing vertex and inside-block symmetries and forcing across-block edges to be missing selected by the BIC.

5.2% compared to 9.4%. The results are summarized in Table 4.1.

It can be concluded that the conditional dependence structure of the gene-expression levels of the two tissues does present some structural similarities, involving in particular the LRP2 gene, which could be a starting point for further investigation. To recover symmetries better, however, a larger sample size would likely be needed.

	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6
Overall edges	169	163	160	847	781	407
Graph density	1.07%	1.03%	1.02%	5.38%	4.96%	2.58%
Inside-block edges	160	160	160	738	734	407
Graph density of $\mathcal{G}_L$	2.66%	2.66%	2.66%	13.15%	13.07%	7.20%
Graph density of $\mathcal{G}_R$	1.43%	1.43%	1.43%	5.69%	5.67%	3.19%
Inside-block density	2.04%	2.04%	2.04%	9.42%	9.37%	5.20%
Inside-block structural symmetries	17	17	17	79	80	39

Table 4.1: Comparison of selected pdRCON models. Model 1 is the model having $\lambda_2^{(A)} = \lambda_2$ selected by eBIC; model 2 is the model having $\lambda_2^{(A)} = \infty$ selected by eBIC; model 3 is the model with no across-block edges selected by eBIC; model 4 is the model having $\lambda_2^{(A)} = \lambda_2$ selected by BIC; model 5 is the model having $\lambda_2^{(A)} = \infty$ selected by BIC; model 6 is the model with no across-block edges selected by BIC.

Conclusion

This work addressed the role played by the across-group association structure in paired data problems. A simulation study was set to investigate whether different hypotheses on the across-group association structure can alter the quality of the recovery of the group-level subgraphs and their symmetries, as well as the accuracy of the estimation of the left and right blocks of the concentration matrix. A pdRCON model forcing the across-block edges to be symmetric, thus reducing the number of parameters in the across block of the concentration matrix to be estimated, leads to improved inference on the two groups, with more edges and symmetries of their relative subgraphs recovered and a better performance in terms of Frobenius norm error of the estimate of the two left and right blocks of the concentration matrix. Lastly, the model subclasses considered in the simulation study were deployed to deal with the breast cancer gene expression data, and the models obtained were compared; however differences in the selected subgraphs and the inside-group parameters were small, especially when model selection was performed using eBIC with $\gamma = 0.5$, so that sparse models with few across-block edges are selected. It is worth remarking that computational aspects are particularly relevant with pdRCON models, as the model selection procedure, even when the grid search for the optimal parameters is over a relatively small number of values, can take hours on a regular laptop in high dimensionality contexts such as in the breast cancer case, having $p = 178$.

Appendix A

Pathwise coordinate descent algorithm

Friedman et al. (2007) propose the pathwise coordinate descent algorithm as follows.

One can observe that the solution of a lasso regression problem with a single predictor,

$$\hat{\beta} = \arg\max_{\beta} \frac{1}{2} \sum_{i=1}^{N} (y_i - x_i \beta)^2 + \lambda \beta \ , \tag{A.1}$$

is very simple, and is the soft-tresholded version of the least square estimate $\tilde{\beta}$:

$$\hat{\beta}^{\text{lasso}} = S_\lambda(\tilde{\beta}) = \text{sign}(\tilde{\beta})(|\tilde{\beta}| - \lambda)_+ \ . \tag{A.2}$$

With multiple, uncorrelated, predictors the lasso solution is once again the soft tresholded version of the $\tilde{\beta}$ vector containing the least squares estimates. In general, with correlated predictors, this is no longer the case. One can instead consider a simple iterative algorithm that applies univariate soft-tresholding for every $j = 1, ..., p$ using $x_{ij}, i = 1, ..., N$ as the predictor and the *partial residuals* $r_i^{(j)}, i = 1, ..., N$ as the response variable, iterating over $j = 1, ..., p, 1, ..., p, 1, ...$ until convergence. $r_i^{(j)} = y_i - \sum_{k \neq j} x_{ik} \hat{\beta}_k = y_i - \hat{y}_i{}_i^{(j)}$ represent the residuals of the regression model without the j-th predictor, and $\hat{\beta}_k$ is the current estimate of β_k in this iterative process.

The algorithm allows to solve the lasso regression problem over a grid of values of λ in a very computationally efficient manner, as each solution can be used as a warm start for the next value of λ in the grid.

Moreover, the pathwise coordinate descent algorithm performs well in the whole class of problems for which the univariate solution can be computed quickly, such as the Least Absolute Deviation lasso (LAD-lasso), the elastic net, the group lasso and many more.

However, the pathwise coordinate descent algorithm does not work in problems where the penalty term is *non-separable*, and therefore cannot be represented as a univariate problem; an example of such instance is with the *fused lasso*.

Appendix B

Conditional distribution of a random variable given the rest of the graph

Let $X = (X_1, ..., X_p)^T$ be a p-dimensional multivariate normal random vector, $X \sim N_p(\mu, \Sigma)$, with $\Theta = \Sigma^{-1}$ concentration matrix. Interest is in the conditional distribution of the j-th component given all other components $X_{-j} = (X_1, ..., X_{j-1}, X_{j+1}, X_p)^T$.

Partition X and Θ, isolating the variable X_j from all others:

$$X = \begin{pmatrix} X_j \\ X_{-j} \end{pmatrix} , \quad \Theta = \begin{pmatrix} \theta_{jj} & \theta_{j,-j} \\ \theta_{-j,j} & \Theta_{-j,-j} \end{pmatrix} . \tag{B.1}$$

The joint probability density function of X is

$$f(X) = \frac{1}{\sqrt{(2\pi)^p \det(\Sigma)}} \exp\left\{ -\frac{1}{2} X^T \Theta X \right\} . \tag{B.2}$$

The term $X^T \Theta X$ in the exponential can be re-written as

$$X^T \Theta X = \begin{pmatrix} X_j & X_{-j}^T \end{pmatrix} \begin{pmatrix} \theta_{jj} & \theta_{j,-j} \\ \theta_{-j,j} & \Theta_{-j,-j} \end{pmatrix} \begin{pmatrix} X_j \\ X_{-j} \end{pmatrix} , \tag{B.3}$$

resulting in the a quadratic form in X_j

$$\begin{aligned} X^T \Theta X &= \theta_{jj} X_j^2 + 2 X_j \theta_{j,-j} X_{-j} + X_{-j}^T \Theta_{-j,-j} X_{-j} \\ &= \theta_{jj} \left(X_j^2 + 2 \frac{\theta_{j,-j} X_{-j}}{\theta_{jj}} X_j \right) + X_{-j}^T \Theta_{-j,-j} X_{-j} , \end{aligned} \tag{B.4}$$

observing that $X_{-j}^T\theta_{-j,j} = \theta_{j,-j}X_{-j}$. Completing the square for X_j the argument of the exponential term in (B.2) becomes

$$-\frac{1}{2}\theta_{jj}\left(\left(X_j + \frac{\theta_{j,-j}X_{-j}}{\theta_{jj}}\right)^2 - \left(\frac{\theta_{j,-j}X_{-j}}{\theta_{jj}}\right)^2\right) - \frac{1}{2}X_{-j}^T\Theta_{-j,-j}X_{-j}\ . \tag{B.5}$$

Looking only at the terms involving X_j and considering X_{-j} as known, a normal distribution for X_j given X_{-j} can be identified, where the conditional mean and variance are

$$\mathbb{E}(X_j|X_{-j}) = -\frac{\theta_{j,-j}X_{-j}}{\theta_{jj}}\ ,\quad \mathrm{Var}(X_j|X_{-j}) = 1/\theta_{jj}\ . \tag{B.6}$$

Observing that $\theta_{j,-j}X_{-j} = \sum_{i\neq j} X_i\theta_{ij}$, the conditional distribution is

$$X_{j|V\setminus\{j\}} \sim N\left(-\sum_{i\neq j} X_i\frac{\theta_{ij}}{\theta_{jj}}, \frac{1}{\theta_{jj}}\right)\ . \tag{B.7}$$

Appendix C

Code

C.1 Simulation study

```
rm(list=ls())
library(pdglasso)
library(tidyverse)
library(ggplot2)
library(see)

#auxiliary functions:
half.vec <- function(M){
        return(M[upper.tri(M, diag=FALSE)])
}

mat2vec <- function(M){
        p  <- dim(M)[1]
        q  <- p/2
        return(c(diag(M[1:q,1:q]),
        diag(M[(q+1):p,(q+1):p]),
        half.vec(M[1:q,1:q]),
        half.vec(M[(q+1):p,(q+1):p]),
        half.vec(M[1:q,(q+1):p]),
        half.vec(M[(q+1):p,1:q]),
        diag(M[1:q,(q+1):p]))
        )
}

#function to perform model selection in the subclass of pdRCON models having
#no across-block edges.
pdRCON.fit.noacrossedges <- function(S,
n,
lams        = NULL,
gamma.eBIC  = 0.5,
type        = c("vertex",
"inside.block.edge",
"across.block.edge"),
force.symm  = NULL,
X.init      = NULL,
rho1        = 1,
rho2        = 1,
varying.rho1= TRUE,
varying.rho2= TRUE,
max_iter    = 5000,
eps.abs     = 1e-08,
eps.rel     = 1e-08,
verbose     = FALSE,
progress    = TRUE,
print.type  = TRUE){
        start.time <- Sys.time()

        if(is.null(lams)){
                lams <- matrix(0,2,4)
                lams[,2] <- lams.max(S)
                lams[,1] <- lams[,2]/20
                lams[,3] <- c(20,20)
                lams[,4] <- c(TRUE,TRUE)
        }
        rownames(lams) <- c("l1","l2")
        colnames(lams) <- c("min","max","n.pts","log.spacing")

        ## Prepare temp objects
        eBIC.l1 <-  matrix(0,lams[1,3],4)
        eBIC.l2 <-  matrix(0,lams[2,3],4)

        ### First grid search for lambda_1, with lambda_2=0
        if(lams[1,4]==1) l1.vec <- exp(seq(log(lams[1,1]),log(lams[1,2]),
        length.out=lams[1,3])) else l1.vec <-
        seq(lams[1,1], lams[1,2], length.out=lams[1,3])
        l1.vec <- sort(l1.vec, decreasing=TRUE)
        p <- NCOL(S)
        q <- p/2
        cat("Selecting a pdRCON model without across-block edges.\n")
        for(i in 1:lams[1,3]){
                if(progress==TRUE) cat(
                "Searching over lambda1 grid (",i,"/",lams[1,3],").\n", sep="")
                lam1 <- matrix(l1.vec[i], nrow = p, ncol = p)
                lam1[1:q,(q+1):p] <- Inf
                lam1[(q+1):p,1:q] <- Inf
                lam1 <- mat2vec(lam1)
                mod.out <- admm.pdglasso(S,
                lambda1=lam1,
                lambda2=0,
                type=type,
                force.symm=force.symm,
                X.init=X.init,
                rho1=rho1,
                rho2=rho2,
                varying.rho1=varying.rho1,
                varying.rho2=varying.rho2,
                max_iter=max_iter,
                eps.abs=eps.abs,
                eps.rel=eps.rel,
                verbose=FALSE,
                print.type=FALSE)
                eBIC.l1[i,1:3] <- compute.eBIC(S=S,
                mod=mod.out,
                n=n,
                gamma.eBIC=gamma.eBIC,
                max_iter=max_iter)
                eBIC.l1[i,4] <- mod.out$internal.par$converged+0
                if(eBIC.l1[i,4]==0) cat
                ("Convergence not achieved for this value of lambda1! \n")
        }
        best.l1 <- l1.vec[which.min(eBIC.l1[,1])]
        if(length(best.l1)==0) stop("Grid search of lambda1 failed!")
        best.l1.mat <- matrix(best.l1, nrow = p, ncol = p)
        best.l1.mat[1:q,(q+1):p] <- Inf
        best.l1.mat[(q+1):p,1:q] <- Inf
        best.l1.mat <- mat2vec(best.l1.mat)

        if(progress==TRUE) cat("--- \n", sep="")

        ### Second grid search for lambda_2, with lambda_1=best.l1.mat
        if(lams[2,4]==1) l2.vec <- exp(seq(log(lams[2,1]),log(lams[2,2]),
        length.out=lams[2,3])) else l2.vec <-
        seq(lams[2,1], lams[2,2], length.out=lams[2,3])
        l2.vec <- sort(l2.vec, decreasing=TRUE)
        for(i in 1:lams[2,3]){
                if(progress==TRUE)
                cat("Searching over lambda2 grid (",i,"/",lams[2,3],").\n", sep="")
                mod.out <- admm.pdglasso(S,
                lambda1=best.l1.mat,
                lambda2=l2.vec[i],
                type=type,
                force.symm=force.symm,
                X.init=X.init,
                rho1=rho1,
                rho2=rho2,
                varying.rho1=varying.rho1,
                varying.rho2=varying.rho2,
                max_iter=max_iter,
                eps.abs=eps.abs,
                eps.rel=eps.rel,
                verbose=FALSE,
                print.type=FALSE)
                eBIC.l2[i,1:3] <- compute.eBIC(S=S,
                mod=mod.out,
                n=n,
                gamma.eBIC=gamma.eBIC,
                max_iter=max_iter)
                eBIC.l2[i,4] <- mod.out$internal.par$converged+0
                if(eBIC.l2[i,4]==0)
                cat("Convergence not achieved for this value of lambda2! \n")
        }
        ### adding eBIC value/results and l2 value to path
        ### already estimated from the first grid.search where lam2=0
        eBIC.l2 <- rbind(eBIC.l2, eBIC.l1[which.min(eBIC.l1[,1]),])
        l2.vec <- c(l2.vec,0)
```

```
        best.l2 <- l2.vec[which.min(eBIC.l2[,1])]
        if(length(best.l2)==0) stop("Grid search of lambda2 failed!")

        ## Fit final model
        mod.out <- admm.pdglasso(S,
        lambda1=best.l1.mat,
        lambda2=best.l2,
        type=type,
        force.symm=force.symm,
        X.init=X.init,
        rho1=rho1,
        rho2=rho2,
        varying.rho1=varying.rho1,
        varying.rho2=varying.rho2,
        max_iter=max_iter,
        eps.abs=eps.abs,
        eps.rel=eps.rel,
        verbose=verbose,
        print.type=print.type)

        l1.path=cbind(l1.vec,eBIC.l1)
        l2.path=cbind(l2.vec,eBIC.l2)
        colnames(l1.path) <- c("lambda1.grid", "eBIC      ",
        "  log-Likelihood  ","num. of params",
        "converged (1=TRUE)")
        colnames(l2.path) <- c("lambda2.grid", "eBIC      ",
        "  log-Likelihood  ","num. of params",
        "converged (1=TRUE)")

        time.exec <- Sys.time()-start.time
        return(list(model=mod.out,
        best.lambdas=c(best.l1,best.l2),
        lambda.grid=lams,
        l1.path=l1.path,
        l2.path=l2.path,
        time.exec=time.exec))
}

#SIMULATION STUDY
#How do different assumptions on the across-block structure of the pdRCON model
#influence inference on the left and right blocks of the concentration matrix?

performance.measures <- function(G_estimated, G_true){
        p <- dim(G_estimated)[1] #number of variables
        n.edges <- p*(p-1)/2 #number of edges in the complete graph
        GL_estimated <- G_estimated != 0 #position of the edges in the estimated graph
        GL_true  <- G_true != 0 #position of the edges in the true graph
        P.G_true <- sum(GL_true[upper.tri(GL_true, diag = FALSE)]) #number of edges
        #in the true graph
        N.G_true <- n.edges - P.G_true #number of missing edges in the true graph
        P.G_estimated <- sum(GL_estimated[upper.tri(GL_estimated, diag = FALSE)])
        #number of edges in the estimated graph
        N.G_estimated <- n.edges - P.G_estimated #number of missing edges in the
        #estimated graph
        TP <- sum(GL_estimated[upper.tri(GL_estimated, diag = FALSE)] &
        GL_true[upper.tri(GL_true, diag = FALSE)]) #number of true
        #edges
        FP <- P.G_estimated - TP #number of false edges
        FN <- P.G_true - TP #number of false missing edges
        TN <- N.G_estimated - FN #number of true missing edges

        G_estimated_lowertri <- G_estimated[lower.tri(G_estimated, diag = TRUE)]
        G_true_lowertri <- G_true[lower.tri(G_true, diag = TRUE)]

        sP.G_estimated <- sum(G_estimated_lowertri == 2) #number of non-zero symmetric
        #concentrations in the estimated concentration matrix
        sP.G_true <- sum(G_true_lowertri == 2) #number of non-zero symmetric
        #concentrations in the true concentration matrix

        sN.G_estimated <- sum(G_estimated_lowertri == 1) #number of parametric
        #asymmetries in the estimated concentration matrix
        sN.G_true <- sum(G_true_lowertri == 1) #number of parametric asymmetries
        #in the true concentration matrix

        sTP <- sum(G_estimated_lowertri == G_true_lowertri & G_true_lowertri == 2)
        #number of true symmetries, non-zero symmetries in the estimated concentration
        #matrix corresponding to non-symmetries in the true concentration matrix
        sTN <- sum(G_estimated_lowertri == G_true_lowertri & G_true_lowertri == 1)
        #number of true parametric asymmetries, parametric asymmetries in the
        #estimated concentration matrix corresponding to parametric asymmetries in
        #the true concentration matrix

        return(list(n.edges=n.edges, P.G_true=P.G_true, N.G_true=N.G_true,
        P.G_estimated=P.G_estimated, N.G_estimated=N.G_estimated,
        TP=TP, FP=FP, TN=TN, FN=FN,
        sP.G_estimated=sP.G_estimated, sP.G_true=sP.G_true,
        sN.G_estimated=sN.G_estimated, sN.G_true=sN.G_true,
        sTP=sTP, sTN=sTN))
}

measures_subgraphs <- function(G, Gtrue) {
        #computes the relevant quantities over the left and right subgraphs, and
        #returns the list with the sum of the quantities
        p <- dim(G)[1]
        q <- p/2
        G_L <- G[1:q,1:q]
        G_R <- G[(q+1):p,(q+1):p]
        Gtrue_L <- Gtrue[1:q,1:q]
        Gtrue_R <- Gtrue[(q+1):p,(q+1):p]
        measures_L <- performance.measures(G_L, Gtrue_L)
        measures_R <- performance.measures(G_R, Gtrue_R)
        totals <- Map('+', measures_L, measures_R)
        return(totals)
}

PPV <- function(A){return(A$TP/A$P.G_estimated)}# Positive Predicted Values or
#Precision
TPR <- function(A){return(A$TP/A$P.G_true)}# True Positive Rate or Recall
TNR <- function(A){return(A$TN/A$N.G_true)} #True Negative Rate
F1  <- function(A){return(2*A$TP/(2*A$TP+A$FP+A$FN))} #F1 score, harmonic mean
```

```
#mean of precision and recall
MCC <- function(A){return((A$TP*A$TN-A$FP*A$FN)/
        sqrt((A$TP+A$FP)*(A$TP+A$FN)*
        (A$TN+A$FP)*(A$TN+A$FN)))} #Matthews
#Correlation Coefficient
sTPR <- function(A){return(A$sTP/A$sP.G_true)} #symmetry - True Positive Rate
sTNR <- function(A){return(A$sTN/A$sN.G_true)} #symmetry - True Negative Rate

simulation.study <- function(N, n.iterations, p, d, dcv, dci, dca) {
        #N is the vector containing the sample sizes
        #n.iterations the number of iterations to be averaged over to get the results
        #measures
        #p is the number of variables in the graph
        #d is the density of non-coloured edges
        #dcv is the density of coloured vertices
        #dci is the density of coloured inside-block edges
        #dca is the density of coloured across-block edges
        layers.labels <- paste("N=", N, sep = "")
        final.results <- array(0, dim=c(3, 24, 7))
        dimnames(final.results) <- list(
        rows = c("lambda2^(A)=lambda2","lambda2^(A)=Inf", "no across-block edges"),
        cols = rep(c("ePPV", "eTPR", "eTNR", "eF1", "eMCC", "sTPR", "sTNR",
        "Frob", "EL", "Npar", "Nedges", "Nsymmetries"),2),
        layers = layers.labels
        )
        times.mle.fails.to.be.computed <- 0 #to keep track of how many times the
        #function pdRCON.mle() fails to compute the MLE of the selected model
        if (p %% 2 != 0) {
                stop("Error: The number of variables 'p' must be an even number.")
        }
        q <- p/2

        start.time <- Sys.time() #to keep track of how much time the simulation takes
        for (i in 1:length(N)) {
                n <- N[i]
                res.1 <- matrix(0, nrow = n.iterations, ncol = 12)
                res.2 <- matrix(0, nrow = n.iterations, ncol = 12)
                res.3 <- matrix(0, nrow = n.iterations, ncol = 12)
                #matrices where the scores of each performance measure for every
                #iteration with the current sample size value (outer for loop)
                #are saved
                for (j in 1:n.iterations) {
                        sim <- pdRCON.simulate(p=p, dens=d,
                        dens.vertex=dcv, dens.inside = dci,
                        dens.across = dca, sample.size = n) #true model
                        #the data is generated from

                        G0 <- sim$pdColG #true coloured graph
                        K0 <- sim$K #true concentration matrix
                        K0_L <- K0[1:q,1:q] #left block of the true concentration matrix
                        K0_R <- K0[(q+1):p,(q+1):p] #right block of the true
                        #concentration matrix
                        Sigma <- solve(K0) #true covariance matrix
                        Sigma_L <- Sigma[1:q,1:q] #left block of the true covariance
                        #matrix
```

```
Sigma_R <- Sigma[(q+1):p,(q+1):p] #right block of the true
#covariance matrix
S <- cov(sim$sample.data) #sample covariance matrix

mod.1 <- pdRCON.fit(S, n, gamma.eBIC = 0)
#selection of a pdRCON model allowing vertex, inside-block and
#across-block symmetries
mod.2 <- pdRCON.fit(S, n,
force.symm = "across.block.edge",
gamma.eBIC = 0)
#selection of a pdRCON model allowing vertex and inside-block
#symmetries and forcing across-block symmetries
mod.3 <- pdRCON.fit.noacrossedges(S, n, gamma.eBIC = 0)
#selection of a pdRCON model allowing vertex and inside-block
#symmetries and not allowing across-block edges

estG.1 <- pdColG.get(mod.1$model)$pdColG
estG.2 <- pdColG.get(mod.2$model)$pdColG
estG.3 <- pdColG.get(mod.3$model)$pdColG
#matrices encoding the estimated graphs for each of the three
#selected models

measures.1 <- measures_subgraphs(estG.1, G0)
measures.2 <- measures_subgraphs(estG.2, G0)
measures.3 <- measures_subgraphs(estG.3, G0)
#relevant quantities to compute the measures related to graph
#structure recovery and symmetry recovery

PPV.1 <- PPV(measures.1)
TPR.1 <- TPR(measures.1)
TNR.1 <- TNR(measures.1)
F1.1 <- F1(measures.1)
MCC.1 <- MCC(measures.1)
sTPR.1 <- sTPR(measures.1)
sTNR.1 <- sTNR(measures.1)
#computation of the measures related to graph structure recovery
#andsymmetry recovery
mle.1 <- pdRCON.mle(S, estG.1)
if (is.null(mle.1)) {
        #if pdRCON.mle fails to compute the MLE, use the solution
        #of the penalized loglikelihood optimization problem as a
        #fallback estimate
        mle.1 <- mod.1$model$X
        mle.1_L <- mle.1[1:q,1:q]
        mle.1_R <- mle.1[(q+1):p,(q+1):p]
        Frob.1_L <- norm(K0_L - mle.1_L, type = "F")
        #Frobenius norm error of the left block
        Frob.1_R <- norm(K0_R - mle.1_R, type = "F")
        #Frobenius norm error of the right block
        Frob.1 <- Frob.1_L + Frob.1_R
        #sum of the Frobenius norm error over the two left and
        #right blocks
        prod.mle.1_L <- mle.1_L %*% Sigma_L
        prod.mle.1_R <- mle.1_R %*% Sigma_R
        EL.1_L <- sum(diag(prod.mle.1_L)) -
        log(det(prod.mle.1_L)) - q
        #entropy loss of the left block
        EL.1_R <- sum(diag(prod.mle.1_R)) -
        log(det(prod.mle.1_R)) - q
        #entropy loss of the right block
        EL.1 <- EL.1_L + EL.1_R
        #sum of the entropy loss over the two left and right blocks
        times.mle.fails.to.be.computed <-
        times.mle.fails.to.be.computed + 1
        #add 1 to the count of the times pdRCON.mle() fails to
        #compute the mle
}
else {
        mle.1_L <- mle.1[1:q,1:q]
        mle.1_R <- mle.1[(q+1):p,(q+1):p]
        Frob.1_L <- norm(K0_L - mle.1_L, type = "F")
        #Frobenius norm error of the left block
        Frob.1_R <- norm(K0_R - mle.1_R, type = "F")
        #Frobenius norm error of the right block
        Frob.1 <- Frob.1_L + Frob.1_R
        #sum of the Frobenius norm error over the two left and
        #right blocks
        prod.mle.1_L <- mle.1_L %*% Sigma_L
        prod.mle.1_R <- mle.1_R %*% Sigma_R
        EL.1_L <- sum(diag(prod.mle.1_L)) -
        log(det(prod.mle.1_L)) - q
        #entropy loss of the left block
        EL.1_R <- sum(diag(prod.mle.1_R)) -
        log(det(prod.mle.1_R)) - q
        #entropy loss of the right block
        EL.1 <- EL.1_L + EL.1_R
        #sum of the entropy loss over the two left and right
        #blocks
}
Npar.1 <- pdColG.get(mod.1$model)$n.par
#total number of parameters estimated by the selected model
#(across-block included)
Nedges.1 <- measures.1$P.G_estimated
Nsymm.1 <- measures.1$sP.G_estimated

PPV.2 <- PPV(measures.2)
TPR.2 <- TPR(measures.2)
TNR.2 <- TNR(measures.2)
F1.2 <- F1(measures.2)
MCC.2 <- MCC(measures.2)
sTPR.2 <- sTPR(measures.2)
sTNR.2 <- sTNR(measures.2)
mle.2 <- pdRCON.mle(S, estG.2)
if (is.null(mle.2)) {
        mle.2 <- mod.2$model$X
        mle.2_L <- mle.2[1:q,1:q]
        mle.2_R <- mle.2[(q+1):p,(q+1):p]
        Frob.2_L <- norm(K0_L - mle.2_L, type = "F")
        Frob.2_R <- norm(K0_R - mle.2_R, type = "F")
        Frob.2 <- Frob.2_L + Frob.2_R
```

```
                prod.mle.2_L <- mle.2_L %*% Sigma_L
                prod.mle.2_R <- mle.2_R %*% Sigma_R
                EL.2_L <- sum(diag(prod.mle.2_L)) -
                log(det(prod.mle.2_L)) - q
                EL.2_R <- sum(diag(prod.mle.2_R)) -
                log(det(prod.mle.2_R)) - q
                EL.2 <- EL.2_L + EL.2_R
                times.mle.fails.to.be.computed <-
                times.mle.fails.to.be.computed + 1
        }
        else {
                mle.2_L <- mle.2[1:q,1:q]
                mle.2_R <- mle.2[(q+1):p,(q+1):p]
                Frob.2_L <- norm(K0_L - mle.2_L, type = "F")
                Frob.2_R <- norm(K0_R - mle.2_R, type = "F")
                Frob.2 <- Frob.2_L + Frob.2_R
                prod.mle.2_L <- mle.2_L %*% Sigma_L
                prod.mle.2_R <- mle.2_R %*% Sigma_R
                EL.2_L <- sum(diag(prod.mle.2_L)) -
                log(det(prod.mle.2_L)) - q
                EL.2_R <- sum(diag(prod.mle.2_R)) -
                log(det(prod.mle.2_R)) - q
                EL.2 <- EL.2_L + EL.2_R
        }
        Npar.2 <- pdColG.get(mod.2$model)$n.par
        Nedges.2 <- measures.2$P.G_estimated
        Nsymm.2 <- measures.2$sP.G_estimated

        PPV.3 <- PPV(measures.3)
        TPR.3 <- TPR(measures.3)
        TNR.3 <- TNR(measures.3)
        F1.3 <- F1(measures.3)
        MCC.3 <- MCC(measures.3)
        sTPR.3 <- sTPR(measures.3)
        sTNR.3 <- sTNR(measures.3)
        mle.3 <- pdRCON.mle(S, estG.3)
        if (is.null(mle.3)) {
                mle.3 <- mod.3$model$X
                mle.3_L <- mle.3[1:q,1:q]
                mle.3_R <- mle.3[(q+1):p,(q+1):p]
                Frob.3_L <- norm(K0_L - mle.3_L, type = "F")
                Frob.3_R <- norm(K0_R - mle.3_R, type = "F")
                Frob.3 <- Frob.3_L + Frob.3_R
                prod.mle.3_L <- mle.3_L %*% Sigma_L
                prod.mle.3_R <- mle.3_R %*% Sigma_R
                EL.3_L <- sum(diag(prod.mle.3_L)) -
                log(det(prod.mle.3_L)) - q
                EL.3_R <- sum(diag(prod.mle.3_R)) -
                log(det(prod.mle.3_R)) - q
                EL.3 <- EL.3_L + EL.3_R
                times.mle.fails.to.be.computed <-
                times.mle.fails.to.be.computed + 1
        }
        else {
                mle.3_L <- mle.3[1:q,1:q]
```

```
                                mle.3_R <- mle.3[(q+1):p,(q+1):p]
                                Frob.3_L <- norm(K0_L - mle.3_L, type = "F")
                                Frob.3_R <- norm(K0_R - mle.3_R, type = "F")
                                Frob.3 <- Frob.3_L + Frob.3_R
                                prod.mle.3_L <- mle.3_L %*% Sigma_L
                                prod.mle.3_R <- mle.3_R %*% Sigma_R
                                EL.3_L <- sum(diag(prod.mle.3_L)) -
                                log(det(prod.mle.3_L)) - q
                                EL.3_R <- sum(diag(prod.mle.3_R)) -
                                log(det(prod.mle.3_R)) - q
                                EL.3 <- EL.3_L + EL.3_R
                        }
                        Npar.3 <- pdColG.get(mod.3$model)$n.par
                        Nedges.3 <- measures.3$P.G_estimated
                        Nsymm.3 <- measures.3$sP.G_estimated

                        res.1[j,] <- c(PPV.1, TPR.1, TNR.1, F1.1, MCC.1,
                        sTPR.1, sTNR.1,
                        Frob.1, EL.1, Npar.1, Nedges.1, Nsymm.1)
                        res.2[j,] <- c(PPV.2, TPR.2, TNR.2, F1.2, MCC.2,
                        sTPR.2, sTNR.2,
                        Frob.2, EL.2, Npar.2, Nedges.2, Nsymm.2)
                        res.3[j,] <- c(PPV.3, TPR.3, TNR.3, F1.3, MCC.3,
                        sTPR.3, sTNR.3,
                        Frob.3, EL.3, Npar.3, Nedges.3, Nsymm.3)
                        #save each measure in the corresponding row of the matrices
                        cat("i = ", i, ", j = ", j, "\n") #to trace progress
                }
                #at the end of the inner for cycle, compute the averages and save them in
                #the results object
                final.results[1,,i] <- c(colMeans(res.1), apply(res.1, 2, sd))
                final.results[2,,i] <- c(colMeans(res.2), apply(res.2, 2, sd))
                final.results[3,,i] <- c(colMeans(res.3), apply(res.3, 2, sd))
        }
        time.exec <- Sys.time()-start.time
        return(list(results = final.results,
        time.exec=time.exec,
        times.mle.fails.to.be.computed = times.mle.fails.to.be.computed))
}

################################################################################
#Scenario 1: p = 20, dens = 0.5, dens.vertex = 0.5, dens.inside = 0.25,
#dens.across = 0.25 (high density, 50% symmetric)
################################################################################

#Example of a true coloured graph:
set.seed(555)
sim <- pdRCON.simulate(p = 20, dens = 0.5, dens.vertex = 0.5,
dens.inside = 0.25, dens.across = 0.25,
sample.size = 100)
G0 <- sim$pdColG
png("examplep20d50dcv50dci25dca25.png", width = 800, height = 600, res = 150)
pdColG.plot(G0)
dev.off()
```

```
N <- c(20, 40, 60, 80, 100, 150, 200)
sim.p20d50dcv50dci25dca25 <- simulation.study(N, n.iterations = 10, p = 20,
d = 0.5, dcv = 0.5, dci = 0.25,
dca = 0.25)
results <- sim.p20d50dcv50dci25dca25$results
sim.p20d50dcv50dci25dca25$times.mle.fails.to.compute
sim.p20d50dcv50dci25dca25$time.exec
save.image("p20d50dcv50dci25dca25.simulation.RData")
#load("p20d50dcv50dci25dca25.simulation.RData")

ePPVdf.means <- data.frame(sample_size = N,
across.symm = results[1,1,],
forcedacross.symm = results[2,1,],
noacross.edges = results[3,1,])
ePPVdf.sds <- data.frame(sample_size = N,
across.symm = results[1,13,],
forcedacross.symm = results[2,13,],
noacross.edges = results[3,13,])
ePPVdf.means <- ePPVdf.means %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
ePPVdf.sds <- ePPVdf.sds %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
merged_ePPVdf <- left_join(ePPVdf.means, ePPVdf.sds,
by = c("sample_size", "group"))
ePPVplot <- ggplot(merged_ePPVdf,
mapping = aes(x=sample_size, y=y.x,
color = group)) +
geom_line() +
geom_errorbar(aes(ymin = y.x - y.y, ymax = y.x + y.y), width = 15,
position = position_dodge(width = 10), size = 0.5) +
labs(title = "edge-Predicted Positive Value",
x="Sample size", y = "", color = NULL) +
scale_color_okabeito(labels = c(expression(lambda[2]^{(A)} == lambda[2]),
expression(lambda[2]^{(A)} == infinity),
"no across edges")) +
theme_minimal() +
theme(legend.position = c(0.8, 0.2),
legend.text = element_text(size = 11))
png("p20d50dcv50dci25dca25.ePPV.png", width = 800, height = 600, res = 150)
ePPVplot
dev.off()

eTPRdf.means <- data.frame(sample_size = N,
across.symm = results[1,2,],
forcedacross.symm = results[2,2,],
noacross.edges = results[3,2,])
eTPRdf.sds <- data.frame(sample_size = N,
across.symm = results[1,14,],
forcedacross.symm = results[2,14,],
noacross.edges = results[3,14,])
eTPRdf.means <- eTPRdf.means %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
eTPRdf.sds <- eTPRdf.sds %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
merged_eTPRdf <- left_join(eTPRdf.means, eTPRdf.sds,
by = c("sample_size", "group"))
eTPRplot <- ggplot(merged_eTPRdf,
mapping = aes(x=sample_size, y=y.x,
color = group)) +
geom_line() +
geom_errorbar(aes(ymin = y.x - y.y, ymax = y.x + y.y), width = 15,
position = position_dodge(width = 10), size = 0.5) +
labs(title = "edge-True Positive Rate",
x="Sample size", y = "", color = NULL) +
scale_color_okabeito(labels = c(expression(lambda[2]^{(A)} == lambda[2]),
expression(lambda[2]^{(A)} == infinity),
"no across edges")) +
theme_minimal() +
theme(legend.position = c(0.8, 0.2),
legend.text = element_text(size = 11))
png("p20d50dcv50dci25dca25.eTPR.png", width = 800, height = 600, res = 150)
eTPRplot
dev.off()

eTNRdf.means <- data.frame(sample_size = N,
across.symm = results[1,3,],
forcedacross.symm = results[2,3,],
noacross.edges = results[3,3,])
eTNRdf.sds <- data.frame(sample_size = N,
across.symm = results[1,15,],
forcedacross.symm = results[2,15,],
noacross.edges = results[3,15,])
eTNRdf.means <- eTNRdf.means %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
eTNRdf.sds <- eTNRdf.sds %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
merged_eTNRdf <- left_join(eTNRdf.means, eTNRdf.sds,
by = c("sample_size", "group"))
eTNRplot <- ggplot(merged_eTNRdf,
mapping = aes(x=sample_size, y=y.x,
color = group)) +
geom_line() +
geom_errorbar(aes(ymin = y.x - y.y, ymax = y.x + y.y), width = 15,
position = position_dodge(width = 10), size = 0.5) +
labs(title = "edge-True Negative Rate",
x="Sample size", y = "", color = NULL) +
scale_color_okabeito(labels = c(expression(lambda[2]^{(A)} == lambda[2]),
expression(lambda[2]^{(A)} == infinity),
"no across edges")) +
theme_minimal() +
theme(legend.position = c(0.8, 0.2),
legend.text = element_text(size = 11))
png("p20d50dcv50dci25dca25.eTNR.png", width = 800, height = 600, res = 150)
eTNRplot
dev.off()

eF1df.means <- data.frame(sample_size = N,
across.symm = results[1,4,],
forcedacross.symm = results[2,4,],
noacross.edges = results[3,4,])
eF1df.sds <- data.frame(sample_size = N,
across.symm = results[1,16,],
forcedacross.symm = results[2,16,],
noacross.edges = results[3,16,])
eF1df.means <- eF1df.means %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
eF1df.sds <- eF1df.sds %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
merged_eF1df <- left_join(eF1df.means, eF1df.sds,
by = c("sample_size", "group"))
eF1plot <- ggplot(merged_eF1df,
mapping = aes(x=sample_size, y=y.x,
color = group)) +
geom_line() +
geom_errorbar(aes(ymin = y.x - y.y, ymax = y.x + y.y), width = 15,
position = position_dodge(width = 10), size = 0.5) +
labs(title = "edge-F1 score",
x="Sample size", y = "", color = NULL) +
scale_color_okabeito(labels = c(expression(lambda[2]^{(A)} == lambda[2]),
expression(lambda[2]^{(A)} == infinity),
"no across edges")) +
theme_minimal() +
theme(legend.position = c(0.8, 0.2),
legend.text = element_text(size = 12))
png("p20d50dcv50dci25dca25.eF1.png", width = 800, height = 600, res = 150)
eF1plot
dev.off()

eMCCdf.means <- data.frame(sample_size = N,
across.symm = results[1,5,],
forcedacross.symm = results[2,5,],
noacross.edges = results[3,5,])
eMCCdf.sds <- data.frame(sample_size = N,
across.symm = results[1,17,],
forcedacross.symm = results[2,17,],
noacross.edges = results[3,17,])
eMCCdf.means <- eMCCdf.means %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
eMCCdf.sds <- eMCCdf.sds %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
merged_eMCCdf <- left_join(eMCCdf.means, eMCCdf.sds,
by = c("sample_size", "group"))
eMCCplot <- ggplot(merged_eMCCdf,
mapping = aes(x=sample_size, y=y.x,
color = group)) +
geom_line() +
geom_errorbar(aes(ymin = y.x - y.y, ymax = y.x + y.y), width = 15,
position = position_dodge(width = 10), size = 0.5) +
labs(title = "edge-MCC",
x="Sample size", y = "", color = NULL) +
scale_color_okabeito(labels = c(expression(lambda[2]^{(A)} == lambda[2]),
expression(lambda[2]^{(A)} == infinity),
"no across edges")) +
theme_minimal() +
theme(legend.position = c(0.8, 0.2),
legend.text = element_text(size = 11))
png("p20d50dcv50dci25dca25.eMCC.png", width = 800, height = 600, res = 150)
eMCCplot
dev.off()

sTPRdf.means <- data.frame(sample_size = N,
across.symm = results[1,6,],
forcedacross.symm = results[2,6,],
noacross.edges = results[3,6,])
sTPRdf.sds <- data.frame(sample_size = N,
across.symm = results[1,18,],
forcedacross.symm = results[2,18,],
noacross.edges = results[3,18,])
sTPRdf.means <- sTPRdf.means %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
sTPRdf.sds <- sTPRdf.sds %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
merged_sTPRdf <- left_join(sTPRdf.means, sTPRdf.sds,
by = c("sample_size", "group"))
sTPRplot <- ggplot(merged_sTPRdf,
mapping = aes(x=sample_size, y=y.x,
color = group)) +
geom_line() +
geom_errorbar(aes(ymin = y.x - y.y, ymax = y.x + y.y), width = 15,
position = position_dodge(width = 10), size = 0.5) +
labs(title = "symmetry-True Positive Rate",
x="Sample size", y = "", color = NULL) +
scale_color_okabeito(labels = c(expression(lambda[2]^{(A)} == lambda[2]),
expression(lambda[2]^{(A)} == infinity),
"no across edges")) +
theme_minimal() +
theme(legend.position = c(0.8, 0.2),
legend.text = element_text(size = 11))
png("p20d50dcv50dci25dca25.sTPR.png", width = 800, height = 600, res = 150)
sTPRplot
dev.off()

sTNRdf.means <- data.frame(sample_size = N,
across.symm = results[1,7,],
forcedacross.symm = results[2,7,],
noacross.edges = results[3,7,])
sTNRdf.sds <- data.frame(sample_size = N,
across.symm = results[1,19,],
forcedacross.symm = results[2,19,],
noacross.edges = results[3,19,])
sTNRdf.means <- sTNRdf.means %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
sTNRdf.sds <- sTNRdf.sds %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
merged_sTNRdf <- left_join(sTNRdf.means, sTNRdf.sds,
by = c("sample_size", "group"))
sTNRplot <- ggplot(merged_sTNRdf,
mapping = aes(x=sample_size, y=y.x,
color = group)) +
geom_line() +
geom_errorbar(aes(ymin = y.x - y.y, ymax = y.x + y.y), width = 15,
position = position_dodge(width = 10), size = 0.5) +
labs(title = "symmetry-True Negative Rate",
x="Sample size", y = "", color = NULL) +
scale_color_okabeito(labels = c(expression(lambda[2]^{(A)} == lambda[2]),
expression(lambda[2]^{(A)} == infinity),
"no across edges")) +
theme_minimal() +
theme(legend.position = c(0.8, 0.2),
legend.text = element_text(size = 11))
png("p20d50dcv50dci25dca25.sTNR.png", width = 800, height = 600, res = 150)
sTNRplot
dev.off()

Frobdf.means <- data.frame(sample_size = N,
across.symm = results[1,8,],
forcedacross.symm = results[2,8,],
noacross.edges = results[3,8,])
Frobdf.sds <- data.frame(sample_size = N,
across.symm = results[1,20,],
forcedacross.symm = results[2,20,],
noacross.edges = results[3,20,])
Frobdf.means <- Frobdf.means %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
Frobdf.sds <- Frobdf.sds %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
merged_Frobdf <- left_join(Frobdf.means, Frobdf.sds,
by = c("sample_size", "group"))
Frobplot <- ggplot(merged_Frobdf,
mapping = aes(x=sample_size, y=y.x,
color = group)) +
geom_line() +
geom_errorbar(aes(ymin = y.x - y.y, ymax = y.x + y.y), width = 15,
position = position_dodge(width = 10), size = 0.5) +
labs(title = "Frobenius norm error",
x="Sample size", y = "", color = NULL) +
scale_color_okabeito(labels = c(expression(lambda[2]^{(A)} == lambda[2]),
expression(lambda[2]^{(A)} == infinity),
"no across edges")) +
theme_minimal() +
theme(legend.position = c(0.8, 0.8),
legend.text = element_text(size = 11))
png("p20d50dcv50dci25dca25.Frob.png", width = 800, height = 600, res = 150)
Frobplot
dev.off()

ELdf.means <- data.frame(sample_size = N,
across.symm = results[1,9,],
forcedacross.symm = results[2,9,],
noacross.edges = results[3,9,])
ELdf.sds <- data.frame(sample_size = N,
across.symm = results[1,21,],
forcedacross.symm = results[2,21,],
noacross.edges = results[3,21,])
ELdf.means <- ELdf.means %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
ELdf.sds <- ELdf.sds %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
merged_ELdf <- left_join(ELdf.means, ELdf.sds,
by = c("sample_size", "group"))
ELplot <- ggplot(merged_ELdf,
mapping = aes(x=sample_size, y=y.x,
color = group)) +
geom_line() +
geom_errorbar(aes(ymin = y.x - y.y, ymax = y.x + y.y), width = 15,
position = position_dodge(width = 10), size = 0.5) +
labs(title = "Entropy loss",
x="Sample size", y = "", color = NULL) +
scale_color_okabeito(labels = c(expression(lambda[2]^{(A)} == lambda[2]),
expression(lambda[2]^{(A)} == infinity),
"no across edges")) +
theme_minimal() +
theme(legend.position = c(0.8, 0.8),
legend.text = element_text(size = 11))
```

```
png("p20d50dcv50dci25dca25.EL.png", width = 800, height = 600, res = 150)
ELplot
dev.off()

Npardf.means <- data.frame(sample_size = N,
across.symm = results[1,10,],
forcedacross.symm = results[2,10,],
noacross.edges = results[3,10,])
Npardf.sds <- data.frame(sample_size = N,
across.symm = results[1,22,],
forcedacross.symm = results[2,22,],
noacross.edges = results[3,22,])
Npardf.means <- Npardf.means %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
Npardf.sds <- Npardf.sds %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
merged_Npardf <- left_join(Npardf.means, Npardf.sds,
by = c("sample_size", "group"))
Nparplot <- ggplot(merged_Npardf,
mapping = aes(x=sample_size, y=y.x,
color = group)) +
geom_line() +
geom_errorbar(aes(ymin = y.x - y.y, ymax = y.x + y.y), width = 15,
position = position_dodge(width = 10), size = 0.5) +
labs(title = "Total number of parameters",
x="Sample size", y = "", color = NULL) +
scale_color_okabeito(label = c(expression(lambda[2]^{(A)} == lambda[2]),
expression(lambda[2]^{(A)} == infinity),
"no across edges")) +
theme_minimal() +
theme(legend.position = c(0.2, 0.8),
legend.text = element_text(size = 11))
png("p20d50dcv50dci25dca25.Npar.png", width = 800, height = 600, res = 150)
Nparplot
dev.off()

Nedgesdf.means <- data.frame(sample_size = N,
across.symm = results[1,11,],
forcedacross.symm = results[2,11,],
noacross.edges = results[3,11,])
Nedgesdf.sds <- data.frame(sample_size = N,
across.symm = results[1,23,],
forcedacross.symm = results[2,23,],
noacross.edges = results[3,23,])
Nedgesdf.means <- Nedgesdf.means %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
Nedgesdf.sds <- Nedgesdf.sds %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
merged_Nedgesdf <- left_join(Nedgesdf.means, Nedgesdf.sds,
by = c("sample_size", "group"))
Nedgesplot <- ggplot(merged_Nedgesdf,
mapping = aes(x=sample_size, y=y.x,
color = group)) +
geom_line() +
geom_errorbar(aes(ymin = y.x - y.y, ymax = y.x + y.y), width = 15,
position = position_dodge(width = 10), size = 0.5) +
labs(title = "Number of edges in G_L and G_R",
x="Sample size", y = "", color = NULL) +
scale_color_okabeito(label = c(expression(lambda[2]^{(A)} == lambda[2]),
expression(lambda[2]^{(A)} == infinity),
"no across edges")) +
theme_minimal() +
theme(legend.position = c(0.8, 0.2),
legend.text = element_text(size = 11))
png("p20d50dcv50dci25dca25.Nedges.png", width = 800, height = 600, res = 150)
Nedgesplot
dev.off()

Nsymmdf.means <- data.frame(sample_size = N,
across.symm = results[1,11,],
forcedacross.symm = results[2,11,],
noacross.edges = results[3,11,])
Nsymmdf.sds <- data.frame(sample_size = N,
across.symm = results[1,23,],
forcedacross.symm = results[2,23,],
noacross.edges = results[3,23,])
Nsymmdf.means <- Nsymmdf.means %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
Nsymmdf.sds <- Nsymmdf.sds %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
merged_Nsymmdf <- left_join(Nsymmdf.means, Nsymmdf.sds,
by = c("sample_size", "group"))
Nsymmplot <- ggplot(merged_Nsymmdf,
mapping = aes(x=sample_size, y=y.x,
color = group)) +
geom_line() +
geom_errorbar(aes(ymin = y.x - y.y, ymax = y.x + y.y), width = 15,
position = position_dodge(width = 10), size = 0.5) +
labs(title = "Number of symmetries in G_L and G_R",
x="Sample size", y = "", color = NULL) +
scale_color_okabeito(label = c(expression(lambda[2]^{(A)} == lambda[2]),
expression(lambda[2]^{(A)} == infinity),
"no across edges")) +
theme_minimal() +
theme(legend.position = c(0.8, 0.2),
legend.text = element_text(size = 11))
png("p20d50dcv50dci25dca25.Nsymm.png", width = 800, height = 600, res = 150)
Nsymmplot
```

```
dev.off()

###############################################################################
#scenario 2: p = 20, dens = 0, dens.vertex = 1, dens.inside = 0.5,
#dens.across = 0.5 (full symmetry)
###############################################################################

#example of a true coloured graph:
set.seed(777)
sim <- pdRCON.simulate(p = 20, dens = 0, dens.vertex = 1,
dens.inside = 0.5, dens.across = 0.5,
sample.size = 100)
G0 <- sim$pdColG

png("examplep20d0dcv100dci50dca50.png", width = 800, height = 600, res = 150)
pdColG.plot(G0)
dev.off()

N <- c(20, 40, 60, 80, 100, 150, 200)
sim.p20d0dcv100dci50dca50 <- simulation.study(N, n.iterations = 10, p = 20,
d = 0, dcv = 1, dci = 0.5,
dca = 0.5)
results <- sim.p20d0dcv100dci50dca50$results
sim.p20d0dcv100dci50dca50$times.mle.fails.to.compute
sim.p20d0dcv100dci50dca50$time.exec
save.image("p20d0dcv100dci50dca50.simulation.RData")
#load("p20d0dcv100dci50dca50.simulation.RData")

ePPVdf.means <- data.frame(sample_size = N,
across.symm = results[1,1,],
forcedacross.symm = results[2,1,],
noacross.edges = results[3,1,])
ePPVdf.sds <- data.frame(sample_size = N,
across.symm = results[1,13,],
forcedacross.symm = results[2,13,],
noacross.edges = results[3,13,])
ePPVdf.means <- ePPVdf.means %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
ePPVdf.sds <- ePPVdf.sds %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
merged_ePPVdf <- left_join(ePPVdf.means, ePPVdf.sds,
by = c("sample_size", "group"))
ePPVplot <- ggplot(merged_ePPVdf,
mapping = aes(x=sample_size, y=y.x,
color = group)) +
geom_line() +
geom_errorbar(aes(ymin = y.x - y.y, ymax = y.x + y.y), width = 15,
position = position_dodge(width = 10), size = 0.5) +
labs(title = "edge-Predicted Positive Value",
x="Sample size", y = "", color = NULL) +
scale_color_okabeito(labels = c(expression(lambda[2]^{(A)} == lambda[2]),
expression(lambda[2]^{(A)} == infinity),
"no across edges")) +
theme_minimal() +
theme(legend.position = c(0.8, 0.2),
legend.text = element_text(size = 11))
png("p20d0dcv100dci50dca50.ePPV.png", width = 800, height = 600, res = 150)
ePPVplot
dev.off()

eTPRdf.means <- data.frame(sample_size = N,
across.symm = results[1,2,],
forcedacross.symm = results[2,2,],
noacross.edges = results[3,2,])
eTPRdf.sds <- data.frame(sample_size = N,
across.symm = results[1,14,],
forcedacross.symm = results[2,14,],
noacross.edges = results[3,14,])
eTPRdf.means <- eTPRdf.means %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
eTPRdf.sds <- eTPRdf.sds %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
merged_eTPRdf <- left_join(eTPRdf.means, eTPRdf.sds,
by = c("sample_size", "group"))
eTPRplot <- ggplot(merged_eTPRdf,
mapping = aes(x=sample_size, y=y.x,
color = group)) +
geom_line() +
geom_errorbar(aes(ymin = y.x - y.y, ymax = y.x + y.y), width = 15,
position = position_dodge(width = 10), size = 0.5) +
labs(title = "edge-True Positive Rate",
x="Sample size", y = "", color = NULL) +
scale_color_okabeito(labels = c(expression(lambda[2]^{(A)} == lambda[2]),
expression(lambda[2]^{(A)} == infinity),
"no across edges")) +
theme_minimal() +
theme(legend.position = c(0.8, 0.2),
legend.text = element_text(size = 11))
png("p20d0dcv100dci50dca50.eTPR.png", width = 800, height = 600, res = 150)
eTPRplot
dev.off()

eTNRdf.means <- data.frame(sample_size = N,
across.symm = results[1,3,],
forcedacross.symm = results[2,3,],
noacross.edges = results[3,3,])
eTNRdf.sds <- data.frame(sample_size = N,
across.symm = results[1,15,],
forcedacross.symm = results[2,15,],
noacross.edges = results[3,15,])
eTNRdf.means <- eTNRdf.means %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
eTNRdf.sds <- eTNRdf.sds %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
merged_eTNRdf <- left_join(eTNRdf.means, eTNRdf.sds,
by = c("sample_size", "group"))
eTNRplot <- ggplot(merged_eTNRdf,
mapping = aes(x=sample_size, y=y.x,
color = group)) +
geom_line() +
geom_errorbar(aes(ymin = y.x - y.y, ymax = y.x + y.y), width = 15,
position = position_dodge(width = 10), size = 0.5) +
labs(title = "edge-True Negative Rate",
x="Sample size", y = "", color = NULL) +
scale_color_okabeito(labels = c(expression(lambda[2]^{(A)} == lambda[2]),
expression(lambda[2]^{(A)} == infinity),
"no across edges")) +
theme_minimal() +
theme(legend.position = c(0.8, 0.2),
legend.text = element_text(size = 11))
png("p20d0dcv100dci50dca50.eTNR.png", width = 800, height = 600, res = 150)
eTNRplot
dev.off()

eF1df.means <- data.frame(sample_size = N,
across.symm = results[1,4,],
forcedacross.symm = results[2,4,],
noacross.edges = results[3,4,])
eF1df.sds <- data.frame(sample_size = N,
across.symm = results[1,16,],
forcedacross.symm = results[2,16,],
noacross.edges = results[3,16,])
eF1df.means <- eF1df.means %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
eF1df.sds <- eF1df.sds %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
merged_eF1df <- left_join(eF1df.means, eF1df.sds,
by = c("sample_size", "group"))
eF1plot <- ggplot(merged_eF1df,
mapping = aes(x=sample_size, y=y.x,
color = group)) +
geom_line() +
geom_errorbar(aes(ymin = y.x - y.y, ymax = y.x + y.y), width = 15,
position = position_dodge(width = 10), size = 0.5) +
labs(title = "edge-F1 score",
x="Sample size", y = "", color = NULL) +
scale_color_okabeito(labels = c(expression(lambda[2]^{(A)} == lambda[2]),
expression(lambda[2]^{(A)} == infinity),
"no across edges")) +
theme_minimal() +
theme(legend.position = c(0.8, 0.2),
legend.text = element_text(size = 12))
png("p20d0dcv100dci50dca50.eF1.png", width = 800, height = 600, res = 150)
eF1plot
dev.off()

eMCCdf.means <- data.frame(sample_size = N,
across.symm = results[1,5,],
forcedacross.symm = results[2,5,],
noacross.edges = results[3,5,])
eMCCdf.sds <- data.frame(sample_size = N,
across.symm = results[1,17,],
forcedacross.symm = results[2,17,],
noacross.edges = results[3,17,])
eMCCdf.means <- eMCCdf.means %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
eMCCdf.sds <- eMCCdf.sds %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
merged_eMCCdf <- left_join(eMCCdf.means, eMCCdf.sds,
by = c("sample_size", "group"))
eMCCplot <- ggplot(merged_eMCCdf,
mapping = aes(x=sample_size, y=y.x,
color = group)) +
geom_line() +
geom_errorbar(aes(ymin = y.x - y.y, ymax = y.x + y.y), width = 15,
position = position_dodge(width = 10), size = 0.5) +
labs(title = "edge-MCC",
x="Sample size", y = "", color = NULL) +
scale_color_okabeito(labels = c(expression(lambda[2]^{(A)} == lambda[2]),
expression(lambda[2]^{(A)} == infinity),
"no across edges")) +
theme_minimal() +
theme(legend.position = c(0.8, 0.2),
legend.text = element_text(size = 11))
png("p20d0dcv100dci50dca50.eMCC.png", width = 800, height = 600, res = 150)
eMCCplot
dev.off()

sTPRdf.means <- data.frame(sample_size = N,
across.symm = results[1,6,],
forcedacross.symm = results[2,6,],
noacross.edges = results[3,6,])
sTPRdf.sds <- data.frame(sample_size = N,
across.symm = results[1,18,],
forcedacross.symm = results[2,18,],
noacross.edges = results[3,18,])
sTPRdf.means <- sTPRdf.means %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
```

```
sTPRdf.sds <- sTPRdf.sds %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
merged_sTPRdf <- left_join(sTPRdf.means, sTPRdf.sds,
by = c("sample_size", "group"))
sTPRplot <- ggplot(merged_sTPRdf,
mapping = aes(x=sample_size, y=y.x,
color = group)) +
geom_line() +
geom_errorbar(aes(ymin = y.x - y.y, ymax = y.x + y.y), width = 15,
position = position_dodge(width = 10), size = 0.5) +
labs(title = "symmetry-True Positive Rate",
x="Sample size", y = "", color = NULL) +
scale_color_okabeito(labels = c(expression(lambda[2]^{(A)} == lambda[2]),
expression(lambda[2]^{(A)} == infinity),
"no across edges")) +
theme_minimal() +
theme(legend.position = c(0.8, 0.2),
legend.text = element_text(size = 11))
png("p20d0dcv100dci50dca50.sTPR.png", width = 800, height = 600, res = 150)
sTPRplot
dev.off()

sTNRdf.means <- data.frame(sample_size = N,
across.symm = results[1,7,],
forcedacross.symm = results[2,7,],
noacross.edges = results[3,7,])
sTNRdf.sds <- data.frame(sample_size = N,
across.symm = results[1,19,],
forcedacross.symm = results[2,19,],
noacross.edges = results[3,19,])
sTNRdf.means <- sTNRdf.means %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
sTNRdf.sds <- sTNRdf.sds %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
merged_sTNRdf <- left_join(sTNRdf.means, sTNRdf.sds,
by = c("sample_size", "group"))
sTNRplot <- ggplot(merged_sTNRdf,
mapping = aes(x=sample_size, y=y.x,
color = group)) +
geom_line() +
geom_errorbar(aes(ymin = y.x - y.y, ymax = y.x + y.y), width = 15,
position = position_dodge(width = 10), size = 0.5) +
labs(title = "symmetry-True Negative Rate",
x="Sample size", y = "", color = NULL) +
scale_color_okabeito(labels = c(expression(lambda[2]^{(A)} == lambda[2]),
expression(lambda[2]^{(A)} == infinity),
"no across edges")) +
theme_minimal() +
theme(legend.position = c(0.8, 0.2),
legend.text = element_text(size = 11))
png("p20d0dcv100dci50dca50.sTNR.png", width = 800, height = 600, res = 150)
sTNRplot
dev.off()

Frobdf.means <- data.frame(sample_size = N,
across.symm = results[1,8,],
forcedacross.symm = results[2,8,],
noacross.edges = results[3,8,])
Frobdf.sds <- data.frame(sample_size = N,
across.symm = results[1,20,],
forcedacross.symm = results[2,20,],
noacross.edges = results[3,20,])
Frobdf.means <- Frobdf.means %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
Frobdf.sds <- Frobdf.sds %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
merged_Frobdf <- left_join(Frobdf.means, Frobdf.sds,
by = c("sample_size", "group"))
Frobplot <- ggplot(merged_Frobdf,
mapping = aes(x=sample_size, y=y.x,
color = group)) +
geom_line() +
geom_errorbar(aes(ymin = y.x - y.y, ymax = y.x + y.y), width = 15,
position = position_dodge(width = 10), size = 0.5) +
labs(title = "Frobenius norm error",
x="Sample size", y = "", color = NULL) +
scale_color_okabeito(labels = c(expression(lambda[2]^{(A)} == lambda[2]),
expression(lambda[2]^{(A)} == infinity),
"no across edges")) +
theme_minimal() +
theme(legend.position = c(0.8, 0.8),
legend.text = element_text(size = 11))
png("p20d0dcv100dci50dca50.Frob.png", width = 800, height = 600, res = 150)
Frobplot
dev.off()

ELdf.means <- data.frame(sample_size = N,
across.symm = results[1,9,],
forcedacross.symm = results[2,9,],
noacross.edges = results[3,9,])
ELdf.sds <- data.frame(sample_size = N,
across.symm = results[1,21,],
forcedacross.symm = results[2,21,],
noacross.edges = results[3,21,])
ELdf.means <- ELdf.means %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
ELdf.sds <- ELdf.sds %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
merged_ELdf <- left_join(ELdf.means, ELdf.sds,
by = c("sample_size", "group"))
ELplot <- ggplot(merged_ELdf,
mapping = aes(x=sample_size, y=y.x,
color = group)) +
geom_line() +
geom_errorbar(aes(ymin = y.x - y.y, ymax = y.x + y.y), width = 15,
position = position_dodge(width = 10), size = 0.5) +
labs(title = "Entropy loss",
x="Sample size", y = "", color = NULL) +
scale_color_okabeito(labels = c(expression(lambda[2]^{(A)} == lambda[2]),
expression(lambda[2]^{(A)} == infinity),
"no across edges")) +
theme_minimal() +
theme(legend.position = c(0.8, 0.8),
legend.text = element_text(size = 11))
png("p20d0dcv100dci50dca50.EL.png", width = 800, height = 600, res = 150)
ELplot
dev.off()

Npardf.means <- data.frame(sample_size = N,
across.symm = results[1,10,],
forcedacross.symm = results[2,10,],
noacross.edges = results[3,10,])
Npardf.sds <- data.frame(sample_size = N,
across.symm = results[1,22,],
forcedacross.symm = results[2,22,],
noacross.edges = results[3,22,])
Npardf.means <- Npardf.means %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
Npardf.sds <- Npardf.sds %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
merged_Npardf <- left_join(Npardf.means, Npardf.sds,
by = c("sample_size", "group"))
Nparplot <- ggplot(merged_Npardf,
mapping = aes(x=sample_size, y=y.x,
color = group)) +
geom_line() +
geom_errorbar(aes(ymin = y.x - y.y, ymax = y.x + y.y), width = 15,
position = position_dodge(width = 10), size = 0.5) +
labs(title = "Total number of parameters",
x="Sample size", y = "", color = NULL) +
scale_color_okabeito(label = c(expression(lambda[2]^{(A)} == lambda[2]),
expression(lambda[2]^{(A)} == infinity),
"no across edges")) +
theme_minimal() +
theme(legend.position = c(0.2, 0.8),
legend.text = element_text(size = 11))
png("p20d0dcv100dci50dca50.Npar.png", width = 800, height = 600, res = 150)
```

```
Nparplot
dev.off()

Nedgesdf.means <- data.frame(sample_size = N,
across.symm = results[1,11,],
forcedacross.symm = results[2,11,],
noacross.edges = results[3,11,])
Nedgesdf.sds <- data.frame(sample_size = N,
across.symm = results[1,23,],
forcedacross.symm = results[2,23,],
noacross.edges = results[3,23,])
Nedgesdf.means <- Nedgesdf.means %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
Nedgesdf.sds <- Nedgesdf.sds %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
merged_Nedgesdf <- left_join(Nedgesdf.means, Nedgesdf.sds,
by = c("sample_size", "group"))
Nedgesplot <- ggplot(merged_Nedgesdf,
mapping = aes(x=sample_size, y=y.x,
color = group)) +
geom_line() +
geom_errorbar(aes(ymin = y.x - y.y, ymax = y.x + y.y), width = 15,
position = position_dodge(width = 10), size = 0.5) +
labs(title = "Number of edges in G_L and G_R",
x="Sample size", y = "", color = NULL) +
scale_color_okabeito(label = c(expression(lambda[2]^{(A)} == lambda[2]),
expression(lambda[2]^{(A)} == infinity),
"no across edges")) +
theme_minimal() +
theme(legend.position = c(0.8, 0.2),
legend.text = element_text(size = 11))
png("p20d0dcv100dci50dca50.Nedges.png", width = 800, height = 600, res = 150)
Nedgesplot
dev.off()

Nsymmdf.means <- data.frame(sample_size = N,
across.symm = results[1,11,],
forcedacross.symm = results[2,11,],
noacross.edges = results[3,11,])
Nsymmdf.sds <- data.frame(sample_size = N,
across.symm = results[1,23,],
forcedacross.symm = results[2,23,],
noacross.edges = results[3,23,])
Nsymmdf.means <- Nsymmdf.means %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
Nsymmdf.sds <- Nsymmdf.sds %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
```

```
merged_Nsymmdf <- left_join(Nsymmdf.means, Nsymmdf.sds,
by = c("sample_size", "group"))
Nsymmplot <- ggplot(merged_Nsymmdf,
mapping = aes(x=sample_size, y=y.x,
color = group)) +
geom_line() +
geom_errorbar(aes(ymin = y.x - y.y, ymax = y.x + y.y), width = 15,
position = position_dodge(width = 10), size = 0.5) +
labs(title = "Number of symmetries in G_L and G_R",
x="Sample size", y = "", color = NULL) +
scale_color_okabeito(label = c(expression(lambda[2]^{(A)} == lambda[2]),
expression(lambda[2]^{(A)} == infinity),
"no across edges")) +
theme_minimal() +
theme(legend.position = c(0.8, 0.2),
legend.text = element_text(size = 11))
png("p20d0dcv100dci50dca50.Nsymm.png", width = 800, height = 600, res = 150)
Nsymmplot
dev.off()

#################################################################################
#scenario 3: p = 20, dens = 0.2, dens.vertex = 0.1, dens.inside = 0.1,
#dens.across = 0.1 (low density, low symmetry)
#################################################################################

#example of a true coloured graph:
set.seed(888)
sim <- pdRCON.simulate(p = 20, dens = 0.2, dens.vertex = 0.1,
dens.inside = 0.1, dens.across = 0.1,
sample.size = 100)
G0 <- sim$pdColG

png("examplep20d20dcv10dci10dca10.png", width = 800, height = 600, res = 150)
pdColG.plot(G0)
dev.off()

N <- c(20, 40, 60, 80, 100, 150, 200)
sim.p20d20dcv10dci10dca10 <- simulation.study(N, n.iterations = 10, p = 20,
d = 0.2, dcv = 0.1, dci = 0.1,
dca = 0.1)
results <- sim.p20d20dcv10dci10dca10$results
sim.p20d20dcv10dci10dca10$times.mle.fails.to.compute
sim.p20d20dcv10dci10dca10$time.exec
save.image("p20d20dcv10dci10dca10.simulation.RData")
#load("p20d20dcv10dci10dca10.simulation.RData")

ePPVdf.means <- data.frame(sample_size = N,
across.symm = results[1,1,],
forcedacross.symm = results[2,1,],
noacross.edges = results[3,1,])
ePPVdf.sds <- data.frame(sample_size = N,
across.symm = results[1,13,],
forcedacross.symm = results[2,13,],
noacross.edges = results[3,13,])
ePPVdf.means <- ePPVdf.means %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
ePPVdf.sds <- ePPVdf.sds %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
merged_ePPVdf <- left_join(ePPVdf.means, ePPVdf.sds,
by = c("sample_size", "group"))
ePPVplot <- ggplot(merged_ePPVdf,
mapping = aes(x=sample_size, y=y.x,
color = group)) +
geom_line() +
geom_errorbar(aes(ymin = y.x - y.y, ymax = y.x + y.y), width = 15,
position = position_dodge(width = 10), size = 0.5) +
labs(title = "edge-Predicted Positive Value",
x="Sample size", y = "", color = NULL) +
scale_color_okabeito(labels = c(expression(lambda[2]^{(A)} == lambda[2]),
expression(lambda[2]^{(A)} == infinity),
"no across edges")) +
theme_minimal() +
theme(legend.position = c(0.8, 0.2),
legend.text = element_text(size = 11))
png("p20d20dcv10dci10dca10.ePPV.png", width = 800, height = 600, res = 150)
ePPVplot
dev.off()

eTPRdf.means <- data.frame(sample_size = N,
across.symm = results[1,2,],
forcedacross.symm = results[2,2,],
noacross.edges = results[3,2,])
eTPRdf.sds <- data.frame(sample_size = N,
across.symm = results[1,14,],
forcedacross.symm = results[2,14,],
noacross.edges = results[3,14,])
eTPRdf.means <- eTPRdf.means %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
eTPRdf.sds <- eTPRdf.sds %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
merged_eTPRdf <- left_join(eTPRdf.means, eTPRdf.sds,
by = c("sample_size", "group"))
eTPRplot <- ggplot(merged_eTPRdf,
mapping = aes(x=sample_size, y=y.x,
color = group)) +
geom_line() +
geom_errorbar(aes(ymin = y.x - y.y, ymax = y.x + y.y), width = 15,
position = position_dodge(width = 10), size = 0.5) +
labs(title = "edge-True Positive Rate",
x="Sample size", y = "", color = NULL) +
scale_color_okabeito(labels = c(expression(lambda[2]^{(A)} == lambda[2]),
expression(lambda[2]^{(A)} == infinity),
"no across edges")) +
theme_minimal() +
theme(legend.position = c(0.8, 0.2),
legend.text = element_text(size = 11))
png("p20d20dcv10dci10dca10.eTPR.png", width = 800, height = 600, res = 150)
eTPRplot
dev.off()

eTNRdf.means <- data.frame(sample_size = N,
across.symm = results[1,3,],
forcedacross.symm = results[2,3,],
noacross.edges = results[3,3,])
eTNRdf.sds <- data.frame(sample_size = N,
across.symm = results[1,15,],
forcedacross.symm = results[2,15,],
noacross.edges = results[3,15,])
eTNRdf.means <- eTNRdf.means %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
eTNRdf.sds <- eTNRdf.sds %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
merged_eTNRdf <- left_join(eTNRdf.means, eTNRdf.sds,
by = c("sample_size", "group"))
eTNRplot <- ggplot(merged_eTNRdf,
mapping = aes(x=sample_size, y=y.x,
color = group)) +
geom_line() +
geom_errorbar(aes(ymin = y.x - y.y, ymax = y.x + y.y), width = 15,
position = position_dodge(width = 10), size = 0.5) +
labs(title = "edge-True Negative Rate",
x="Sample size", y = "", color = NULL) +
scale_color_okabeito(labels = c(expression(lambda[2]^{(A)} == lambda[2]),
expression(lambda[2]^{(A)} == infinity),
"no across edges")) +
theme_minimal() +
theme(legend.position = c(0.8, 0.2),
legend.text = element_text(size = 11))
png("p20d20dcv10dci10dca10.eTNR.png", width = 800, height = 600, res = 150)
eTNRplot
dev.off()

eF1df.means <- data.frame(sample_size = N,
across.symm = results[1,4,],
forcedacross.symm = results[2,4,],
noacross.edges = results[3,4,])
eF1df.sds <- data.frame(sample_size = N,
across.symm = results[1,16,],
forcedacross.symm = results[2,16,],
noacross.edges = results[3,16,])
eF1df.means <- eF1df.means %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
eF1df.sds <- eF1df.sds %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
merged_eF1df <- left_join(eF1df.means, eF1df.sds,
by = c("sample_size", "group"))
eF1plot <- ggplot(merged_eF1df,
mapping = aes(x=sample_size, y=y.x,
color = group)) +
geom_line() +
geom_errorbar(aes(ymin = y.x - y.y, ymax = y.x + y.y), width = 15,
position = position_dodge(width = 10), size = 0.5) +
labs(title = "edge-F1 score",
x="Sample size", y = "", color = NULL) +
scale_color_okabeito(labels = c(expression(lambda[2]^{(A)} == lambda[2]),
expression(lambda[2]^{(A)} == infinity),
"no across edges")) +
theme_minimal() +
theme(legend.position = c(0.8, 0.2),
legend.text = element_text(size = 12))
png("p20d20dcv10dci10dca10.eF1.png", width = 800, height = 600, res = 150)
eF1plot
dev.off()

eMCCdf.means <- data.frame(sample_size = N,
across.symm = results[1,5,],
forcedacross.symm = results[2,5,],
noacross.edges = results[3,5,])
eMCCdf.sds <- data.frame(sample_size = N,
across.symm = results[1,17,],
forcedacross.symm = results[2,17,],
noacross.edges = results[3,17,])
eMCCdf.means <- eMCCdf.means %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
eMCCdf.sds <- eMCCdf.sds %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
merged_eMCCdf <- left_join(eMCCdf.means, eMCCdf.sds,
by = c("sample_size", "group"))
eMCCplot <- ggplot(merged_eMCCdf,
mapping = aes(x=sample_size, y=y.x,
color = group)) +
geom_line() +
geom_errorbar(aes(ymin = y.x - y.y, ymax = y.x + y.y), width = 15,
position = position_dodge(width = 10), size = 0.5) +
labs(title = "edge-MCC",
x="Sample size", y = "", color = NULL) +
scale_color_okabeito(labels = c(expression(lambda[2]^{(A)} == lambda[2]),
expression(lambda[2]^{(A)} == infinity),
"no across edges")) +
theme_minimal() +
theme(legend.position = c(0.8, 0.2),
legend.text = element_text(size = 11))
png("p20d20dcv10dci10dca10.eMCC.png", width = 800, height = 600, res = 150)
eMCCplot
dev.off()

sTPRdf.means <- data.frame(sample_size = N,
across.symm = results[1,6,],
forcedacross.symm = results[2,6,],
noacross.edges = results[3,6,])
sTPRdf.sds <- data.frame(sample_size = N,
across.symm = results[1,18,],
forcedacross.symm = results[2,18,],
noacross.edges = results[3,18,])
sTPRdf.means <- sTPRdf.means %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
sTPRdf.sds <- sTPRdf.sds %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
merged_sTPRdf <- left_join(sTPRdf.means, sTPRdf.sds,
by = c("sample_size", "group"))
sTPRplot <- ggplot(merged_sTPRdf,
mapping = aes(x=sample_size, y=y.x,
color = group)) +
geom_line() +
geom_errorbar(aes(ymin = y.x - y.y, ymax = y.x + y.y), width = 15,
position = position_dodge(width = 10), size = 0.5) +
labs(title = "symmetry-True Positive Rate",
x="Sample size", y = "", color = NULL) +
scale_color_okabeito(labels = c(expression(lambda[2]^{(A)} == lambda[2]),
expression(lambda[2]^{(A)} == infinity),
"no across edges")) +
theme_minimal() +
theme(legend.position = c(0.8, 0.2),
legend.text = element_text(size = 11))
png("p20d20dcv10dci10dca10.sTPR.png", width = 800, height = 600, res = 150)
sTPRplot
dev.off()

sTNRdf.means <- data.frame(sample_size = N,
across.symm = results[1,7,],
forcedacross.symm = results[2,7,],
noacross.edges = results[3,7,])
sTNRdf.sds <- data.frame(sample_size = N,
across.symm = results[1,19,],
forcedacross.symm = results[2,19,],
noacross.edges = results[3,19,])
sTNRdf.means <- sTNRdf.means %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
sTNRdf.sds <- sTNRdf.sds %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
merged_sTNRdf <- left_join(sTNRdf.means, sTNRdf.sds,
by = c("sample_size", "group"))
sTNRplot <- ggplot(merged_sTNRdf,
mapping = aes(x=sample_size, y=y.x,
color = group)) +
geom_line() +
geom_errorbar(aes(ymin = y.x - y.y, ymax = y.x + y.y), width = 15,
position = position_dodge(width = 10), size = 0.5) +
labs(title = "symmetry-True Negative Rate",
x="Sample size", y = "", color = NULL) +
scale_color_okabeito(labels = c(expression(lambda[2]^{(A)} == lambda[2]),
expression(lambda[2]^{(A)} == infinity),
"no across edges")) +
theme_minimal() +
theme(legend.position = c(0.8, 0.2),
legend.text = element_text(size = 11))
png("p20d20dcv10dci10dca10.sTNR.png", width = 800, height = 600, res = 150)
sTNRplot
dev.off()

Frobdf.means <- data.frame(sample_size = N,
across.symm = results[1,8,],
forcedacross.symm = results[2,8,],
noacross.edges = results[3,8,])
Frobdf.sds <- data.frame(sample_size = N,
across.symm = results[1,20,],
forcedacross.symm = results[2,20,],
noacross.edges = results[3,20,])
Frobdf.means <- Frobdf.means %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
Frobdf.sds <- Frobdf.sds %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
merged_Frobdf <- left_join(Frobdf.means, Frobdf.sds,
by = c("sample_size", "group"))
Frobplot <- ggplot(merged_Frobdf,
mapping = aes(x=sample_size, y=y.x,
color = group)) +
geom_line() +
geom_errorbar(aes(ymin = y.x - y.y, ymax = y.x + y.y), width = 15,
position = position_dodge(width = 10), size = 0.5) +
labs(title = "Frobenius norm error",
x="Sample size", y = "", color = NULL) +
scale_color_okabeito(labels = c(expression(lambda[2]^{(A)} == lambda[2]),
expression(lambda[2]^{(A)} == infinity),
"no across edges")) +
theme_minimal() +
theme(legend.position = c(0.8, 0.8),
legend.text = element_text(size = 11))
```

```
png("p20d20dcv10dci10dca10.Frob.png", width = 800, height = 600, res = 150)
Frobplot
dev.off()

ELdf.means <- data.frame(sample_size = N,
across.symm = results[1,9,],
forcedacross.symm = results[2,9,],
noacross.edges = results[3,9,])
ELdf.sds <- data.frame(sample_size = N,
across.symm = results[1,21,],
forcedacross.symm = results[2,21,],
noacross.edges = results[3,21,])
ELdf.means <- ELdf.means %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
ELdf.sds <- ELdf.sds %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
merged_ELdf <- left_join(ELdf.means, ELdf.sds,
by = c("sample_size", "group"))
ELplot <- ggplot(merged_ELdf,
mapping = aes(x=sample_size, y=y.x,
color = group)) +
geom_line() +
geom_errorbar(aes(ymin = y.x - y.y, ymax = y.x + y.y), width = 15,
position = position_dodge(width = 10), size = 0.5) +
labs(title = "Entropy loss",
x="Sample size", y = "", color = NULL) +
scale_color_okabeito(labels = c(expression(lambda[2]^{(A)} == lambda[2]),
expression(lambda[2]^{(A)} == infinity),
"no across edges")) +
theme_minimal() +
theme(legend.position = c(0.8, 0.8),
legend.text = element_text(size = 11))
png("p20d20dcv10dci10dca10.EL.png", width = 800, height = 600, res = 150)
ELplot
dev.off()

Npardf.means <- data.frame(sample_size = N,
across.symm = results[1,10,],
forcedacross.symm = results[2,10,],
noacross.edges = results[3,10,])
Npardf.sds <- data.frame(sample_size = N,
across.symm = results[1,22,],
forcedacross.symm = results[2,22,],
noacross.edges = results[3,22,])
Npardf.means <- Npardf.means %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
Npardf.sds <- Npardf.sds %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
merged_Npardf <- left_join(Npardf.means, Npardf.sds,
by = c("sample_size", "group"))
Nparplot <- ggplot(merged_Npardf,
mapping = aes(x=sample_size, y=y.x,
color = group)) +
geom_line() +
geom_errorbar(aes(ymin = y.x - y.y, ymax = y.x + y.y), width = 15,
position = position_dodge(width = 10), size = 0.5) +
labs(title = "Total number of parameters",
x="Sample size", y = "", color = NULL) +
scale_color_okabeito(label = c(expression(lambda[2]^{(A)} == lambda[2]),
expression(lambda[2]^{(A)} == infinity),
"no across edges")) +
theme_minimal() +
theme(legend.position = c(0.2, 0.8),
legend.text = element_text(size = 11))
png("p20d20dcv10dci10dca10.Npar.png", width = 800, height = 600, res = 150)
Nparplot
dev.off()

Nedgesdf.means <- data.frame(sample_size = N,
across.symm = results[1,11,],
forcedacross.symm = results[2,11,],
noacross.edges = results[3,11,])
Nedgesdf.sds <- data.frame(sample_size = N,
across.symm = results[1,23,],
forcedacross.symm = results[2,23,],
noacross.edges = results[3,23,])
Nedgesdf.means <- Nedgesdf.means %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
Nedgesdf.sds <- Nedgesdf.sds %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
merged_Nedgesdf <- left_join(Nedgesdf.means, Nedgesdf.sds,
by = c("sample_size", "group"))
Nedgesplot <- ggplot(merged_Nedgesdf,
mapping = aes(x=sample_size, y=y.x,
color = group)) +
geom_line() +
geom_errorbar(aes(ymin = y.x - y.y, ymax = y.x + y.y), width = 15,
position = position_dodge(width = 10), size = 0.5) +
labs(title = "Number of edges in G_L and G_R",
x="Sample size", y = "", color = NULL) +
scale_color_okabeito(label = c(expression(lambda[2]^{(A)} == lambda[2]),
expression(lambda[2]^{(A)} == infinity),
"no across edges")) +
theme_minimal() +
theme(legend.position = c(0.8, 0.2),
legend.text = element_text(size = 11))
png("p20d20dcv10dci10dca10.Nedges.png", width = 800, height = 600, res = 150)
Nedgesplot
```

```
dev.off()

Nsymmdf.means <- data.frame(sample_size = N,
across.symm = results[1,11,],
forcedacross.symm = results[2,11,],
noacross.edges = results[3,11,])
Nsymmdf.sds <- data.frame(sample_size = N,
across.symm = results[1,23,],
forcedacross.symm = results[2,23,],
noacross.edges = results[3,23,])
Nsymmdf.means <- Nsymmdf.means %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
Nsymmdf.sds <- Nsymmdf.sds %>%
pivot_longer(cols = -sample_size,
names_to = "group",
values_to = "y")
merged_Nsymmdf <- left_join(Nsymmdf.means, Nsymmdf.sds,
by = c("sample_size", "group"))
Nsymmplot <- ggplot(merged_Nsymmdf,
mapping = aes(x=sample_size, y=y.x,
color = group)) +
geom_line() +
geom_errorbar(aes(ymin = y.x - y.y, ymax = y.x + y.y), width = 15,
position = position_dodge(width = 10), size = 0.5) +
labs(title = "Number of symmetries in G_L and G_R",
x="Sample size", y = "", color = NULL) +
scale_color_okabeito(label = c(expression(lambda[2]^{(A)} == lambda[2]),
expression(lambda[2]^{(A)} == infinity),
"no across edges")) +
theme_minimal() +
theme(legend.position = c(0.8, 0.2),
legend.text = element_text(size = 11))
png("p20d20dcv10dci10dca10.Nsymm.png", width = 800, height = 600, res = 150)
Nsymmplot
dev.off()
```

C.2 Breast cancer data analysis

```
rm(list=ls())
library(pdglasso)
load("breast_cancer.RData")
source("pdRCON.fit.noacrossedges.R")
N <- nrow(bc.data)

#Model 1: pdRCON allowing vertex, inside-block, and across-block symmetries
mod.1 <- pdRCON.fit(S, N)
estG.1 <- pdColG.get(mod.1$model)$pdColG
estK.1 <- mod.1$model$X
png("breastcancermod1.png", width = 800, height = 600, res = 150)
pdColG.plot(estG.1)
```

```
dev.off()

#Model 2: pdRCON allowing vertex and inside-block symmetries, and forcing
#across-block symmetries
mod.2 <- pdRCON.fit(S, N, force.symm = "across.block.edge")
estG.2 <- pdColG.get(mod.2$model)$pdColG
estK.2 <- mod.2$model$X
png("breastcancermod2.png", width = 800, height = 600, res = 150)
pdColG.plot(estG.2)
dev.off()

#Model 3: pdRCON allowing vertex and inside-block symmetries, and forcing every
#across-block edge to be missing
mod.3 <- pdRCON.fit.noacrossedges(S, N)
estG.3 <- pdColG.get(mod.3$model)$pdColG
estK.3 <- mod.3$model$X
png("breastcancermod3.png", width = 800, height = 600, res = 150)
pdColG.plot(estG.3)
dev.off()

save.image("breastcancermodels.RData")

pdColG.summarize(estG.1)
pdColG.summarize(estG.2)
pdColG.summarize(estG.3)

p <- dim(S)[1]
q <- p/2

estG.1_L <- estG.1[1:q,1:q]
estG.1_R <- estG.1[(q+1):p,(q+1):p]

estG.1_L_lowertri <- estG.1_L[lower.tri(estG.1_L)]
estG.1_R_lowertri <- estG.1_R[lower.tri(estG.1_L)]

sum(estG.1_L_lowertri != 0 & estG.1_R_lowertri != 0) #number of structural
#inside-block symmetries

n.edges.estG.1_L <- sum(estG.1_L_lowertri != 0)
n.edges.estG.1_R <- sum(estG.1_R_lowertri != 0)

n.edges.estG.1_L/((q)*(q-1)/2) #graph density of G_L
n.edges.estG.1_R/((q)*(q-1)/2) #graph density of G_R

(n.edges.estG.1_L + n.edges.estG.1_R) / (q * (q - 1)) #inside-block
#graph density

common_non_zero <- (estG.1_L != 0) & (estG.1_R != 0)

row_common_counts <- rowSums(common_non_zero)
col_common_counts <- colSums(common_non_zero)

max_row_index <- which(row_common_counts == max(row_common_counts))
max_col_index <- which(col_common_counts == max(col_common_counts))
```

```
gene.names[max_row_index]
gene.names[max_col_index]

estG.2_L <- estG.2[1:q,1:q]
estG.2_R <- estG.2[(q+1):p,(q+1):p]

estG.3_L <- estG.3[1:q,1:q]
estG.3_R <- estG.3[(q+1):p,(q+1):p]

estK.1_L <- estK.1[1:q,1:q]
estK.1_R <- estK.1[(q+1):p,(q+1):p]

estK.2_L <- estK.2[1:q,1:q]
estK.2_R <- estK.2[(q+1):p,(q+1):p]

estK.3_L <- estK.3[1:q,1:q]
estK.3_R <- estK.3[(q+1):p,(q+1):p]

identical(estG.1_L, estG.2_L) && identical(estG.1_L, estG.3_L)
identical(estG.1_R, estG.2_R) && identical(estG.1_R, estG.3_R)

norm(estK.1_L - estK.2_L, type = "f")
norm(estK.1_L - estK.3_L, type = "f")

norm(estK.1_R - estK.2_R, type = "f")
norm(estK.1_R - estK.3_R, type = "f")

#with BIC:

#Model 1: pdRCON allowing vertex, inside-block, and across-block symmetries
mod.1.BIC <- pdRCON.fit(S, N, gamma.eBIC = 0)
estG.1.BIC <- pdColG.get(mod.1.BIC$model)$pdColG
png("breastcancermod1BIC.png", width = 800, height = 600, res = 150)
pdColG.plot(estG.1.BIC)
dev.off()
save.image("breastcancermodels.RData")

#Model 2: pdRCON allowing vertex and inside-block symmetries, and forcing
#across-block symmetries
mod.2.BIC <- pdRCON.fit(S, N, force.symm = "across.block.edge", gamma.eBIC = 0)
estG.2.BIC <- pdColG.get(mod.2.BIC$model)$pdColG
png("breastcancermod2BIC.png", width = 800, height = 600, res = 150)
pdColG.plot(estG.2.BIC)
dev.off()
save.image("breastcancermodels.RData")

#Model 3: pdRCON allowing vertex and inside-block symmetries, and forcing every
#across-block edge to be missing
mod.3.BIC <- pdRCON.fit.noacrossedges(S, N, gamma.eBIC = 0)
estG.3.BIC <- pdColG.get(mod.3.BIC$model)$pdColG
png("breastcancermod3BIC.png", width = 800, height = 600, res = 150)
pdColG.plot(estG.3.BIC)
dev.off()
save.image("breastcancermodels.RData")
#load("breastcancermodels.RData")
```

```
pdColG.summarize(estG.1.BIC)
pdColG.summarize(estG.2.BIC)
pdColG.summarize(estG.3.BIC)

estG.1.BIC_L <- estG.1.BIC[1:q,1:q]
estG.1.BIC_R <- estG.1.BIC[(q+1):p,(q+1):p]

estG.1.BIC_L_lowertri <- estG.1.BIC_L[lower.tri(estG.1.BIC_L)]
estG.1.BIC_R_lowertri <- estG.1.BIC_R[lower.tri(estG.1.BIC_L)]

sum(estG.1.BIC_L_lowertri != 0 & estG.1.BIC_R_lowertri != 0)

n.edges.estG.1.BIC_L <- sum(estG.1.BIC_L_lowertri != 0)
n.edges.estG.1.BIC_R <- sum(estG.1.BIC_R_lowertri != 0)

n.edges.estG.1.BIC_L/((q)*(q-1)/2) #graph density of G_L
n.edges.estG.1.BIC_R/((q)*(q-1)/2) #graph density of G_R

(n.edges.estG.1.BIC_L + n.edges.estG.1.BIC_R) / (q * (q - 1)) #inside-block
#graph density

estG.2.BIC_L <- estG.2.BIC[1:q,1:q]
estG.2.BIC_R <- estG.2.BIC[(q+1):p,(q+1):p]

estG.2.BIC_L_lowertri <- estG.2.BIC_L[lower.tri(estG.2.BIC_L)]
estG.2.BIC_R_lowertri <- estG.2.BIC_R[lower.tri(estG.2.BIC_L)]

sum(estG.2.BIC_L_lowertri != 0 & estG.2.BIC_R_lowertri != 0)

n.edges.estG.2.BIC_L <- sum(estG.2.BIC_L_lowertri != 0)
n.edges.estG.2.BIC_R <- sum(estG.2.BIC_R_lowertri != 0)

(n.edges.estG.2.BIC_L + n.edges.estG.2.BIC_R) / (q * (q - 1))

n.edges.estG.2.BIC_L/((q)*(q-1)/2) #graph density of G_L
n.edges.estG.2.BIC_R/((q)*(q-1)/2) #graph density of G_R

sum(estG.1.BIC_L_lowertri != estG.2.BIC_L_lowertri)
estG.1.BIC_L[which(estG.1.BIC_L != estG.2.BIC_L)]
estG.2.BIC_L[which(estG.1.BIC_L != estG.2.BIC_L)]

sum(estG.1.BIC_R_lowertri != estG.2.BIC_R_lowertri)

1 - (sum(estG.1.BIC_L_lowertri != estG.2.BIC_L_lowertri) +
sum(estG.1.BIC_R_lowertri != estG.2.BIC_R_lowertri)) /
(n.edges.estG.1.BIC_L + n.edges.estG.1.BIC_R)


estG.3.BIC_L <- estG.3.BIC[1:q,1:q]
estG.3.BIC_R <- estG.3.BIC[(q+1):p,(q+1):p]

estG.3.BIC_L_lowertri <- estG.3.BIC_L[lower.tri(estG.3.BIC_L)]
estG.3.BIC_R_lowertri <- estG.3.BIC_R[lower.tri(estG.3.BIC_L)]
```

```
sum(estG.3.BIC_L_lowertri != 0 & estG.3.BIC_R_lowertri != 0)

n.edges.estG.3.BIC_L <- sum(estG.3.BIC_L_lowertri != 0)
n.edges.estG.3.BIC_R <- sum(estG.3.BIC_R_lowertri != 0)

n.edges.estG.3.BIC_L/((q)*(q-1)/2) #graph density of G_L
n.edges.estG.3.BIC_R/((q)*(q-1)/2) #graph density of G_R

(n.edges.estG.3.BIC_L + n.edges.estG.3.BIC_R) / (q * (q - 1))
```

Appendix D

Additional graphs

D.1 High density, 50% symmetric setting

When increasing the dimensionality to $\texttt{p} = 50$, the advantage given by assuming all across-block to edges to be symmetric in terms of structure recovery of $\mathcal{G}_L$ and $\mathcal{G}_R$ in the first scenario is made clearer, with the model performing consistently better in terms of F1 score and edge-True Positive Rate (Figure D.1), as well as symmetry-True Positive Rate (Figure D.2), throughout all sample and a higher absolute number of edges and symmetries is estimated in the two subgraphs (Figure D.4). The assumption $\lambda_2^{(V)} = \infty$ leads also to a more accurate estimate in terms of Frobenius norm error, while the no across-block edges assumption leads to the smallest entropy loss (Figure D.3.)

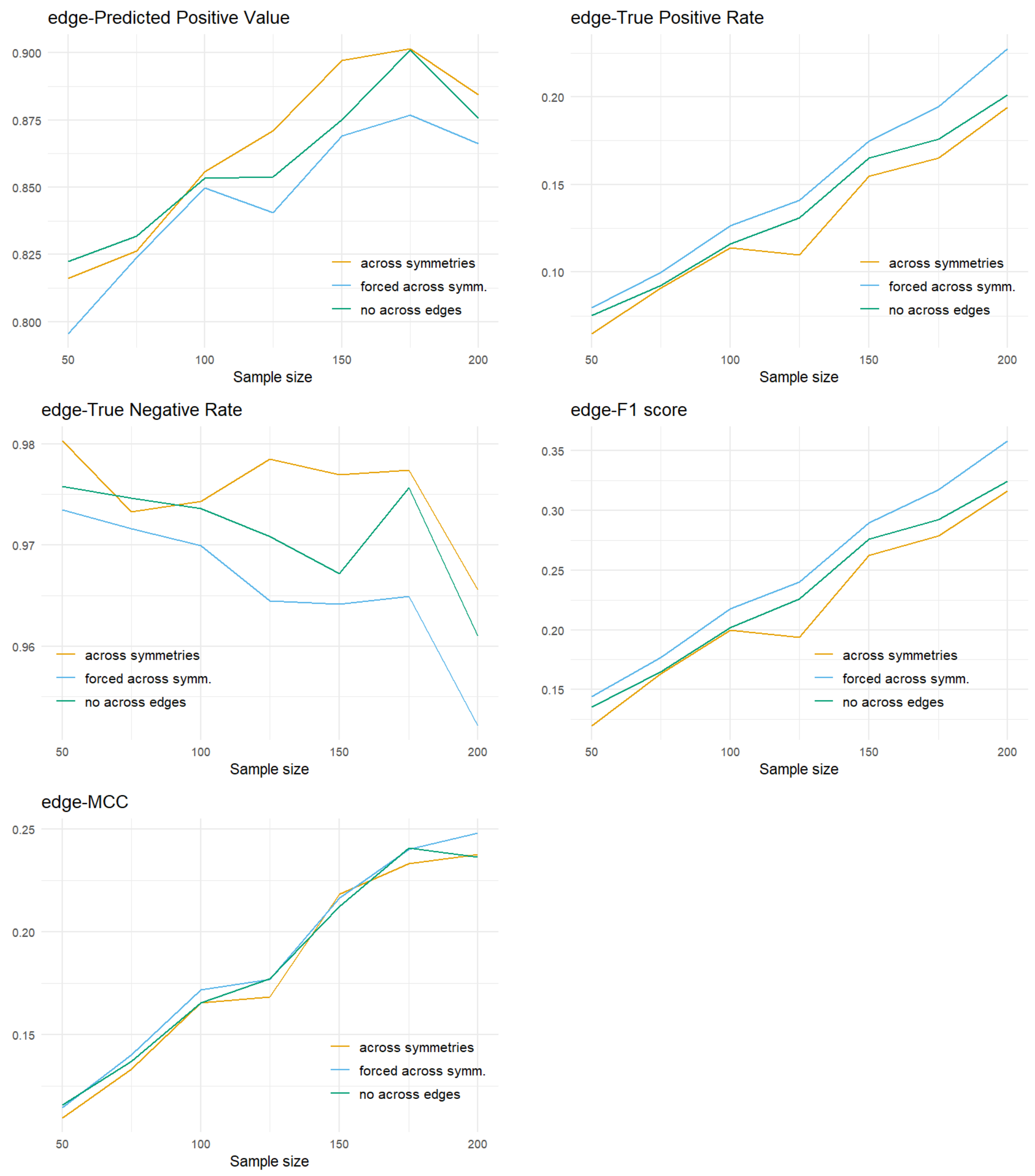


FIGURE D.1: Structure recovery measures of $\mathcal{G}_L$ and $\mathcal{G}_R$ for the high density, 50% symmetric setting, when $\mathtt{p} = 50$.

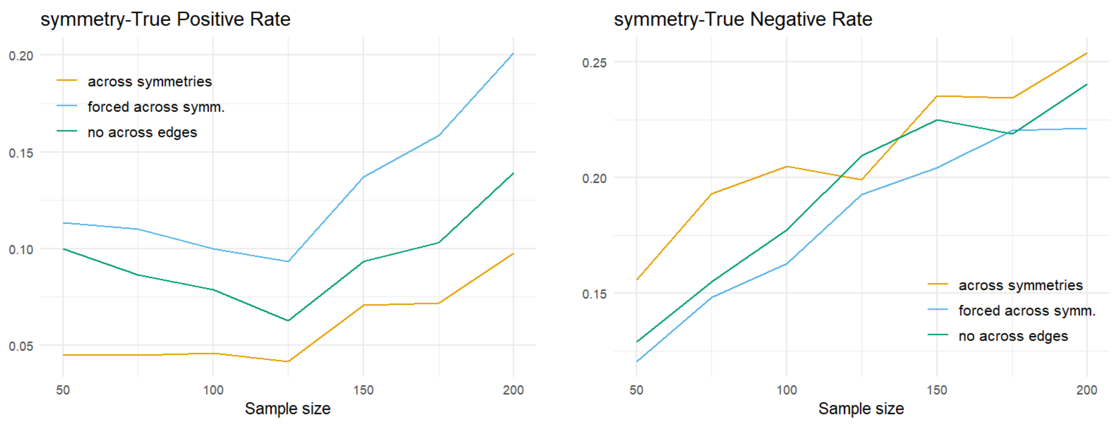


FIGURE D.2: Symmetric structure recovery measures of $\mathcal{G}_L$ and $\mathcal{G}_R$ for the high density, 50% symmetric setting, when $\mathtt{p} = 50$.

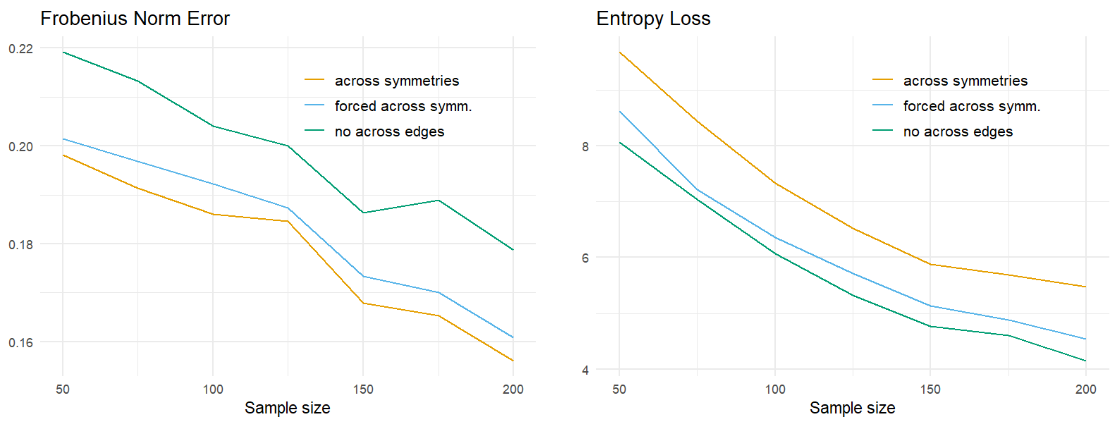
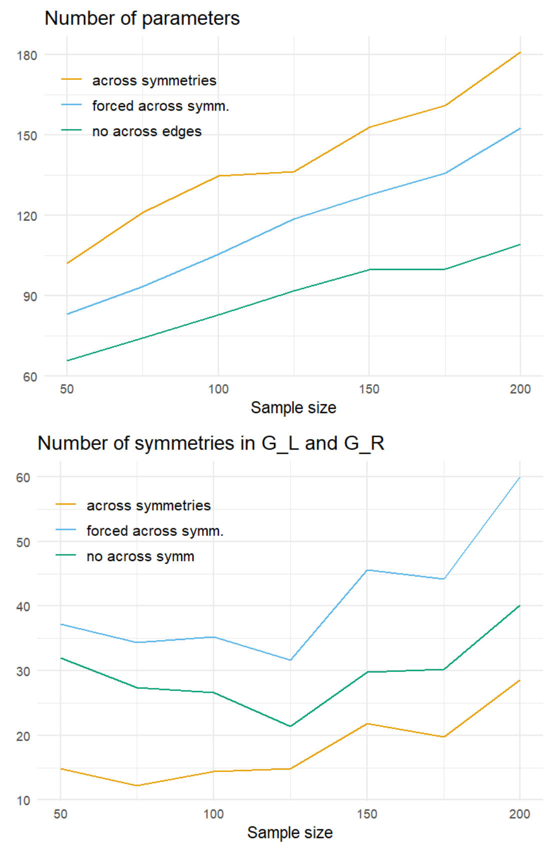


FIGURE D.3: Estimation accuracy measures of Θ_{LL} and Θ_{RR} for the high density, 50% symmetric setting, when $\mathtt{p} = 50$.



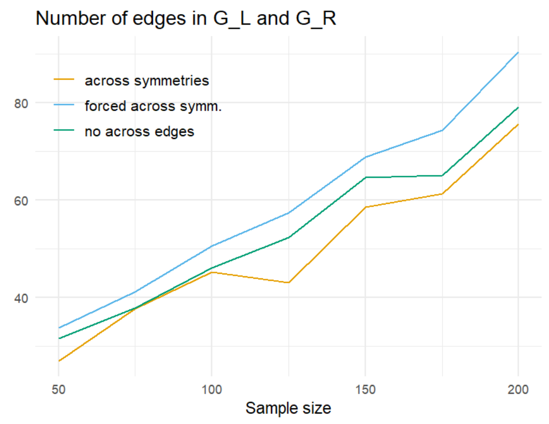


FIGURE D.4: Total number of parameters estimated and numbers of edges and non-zero parametric symmetries in $\mathcal{G}_L$ and $\mathcal{G}_R$ for the high density, 50% symmetric setting, when $\mathtt{p} = 50$.

D.2 High density, fully symmetric setting

Differences in terms of structure recovery are not clear (Figure D.5), but the model forcing across-block symmetries is distinctly better in terms of symmetry-True Positive Rate (Figure D.6), and a has a higher total number of parameters being estimated (Figure D.8). The no across-block edges hypothesis, on the other hand, leads to a substantial improvement in terms of entropy loss when compared to the other two models. (Figure D.7).

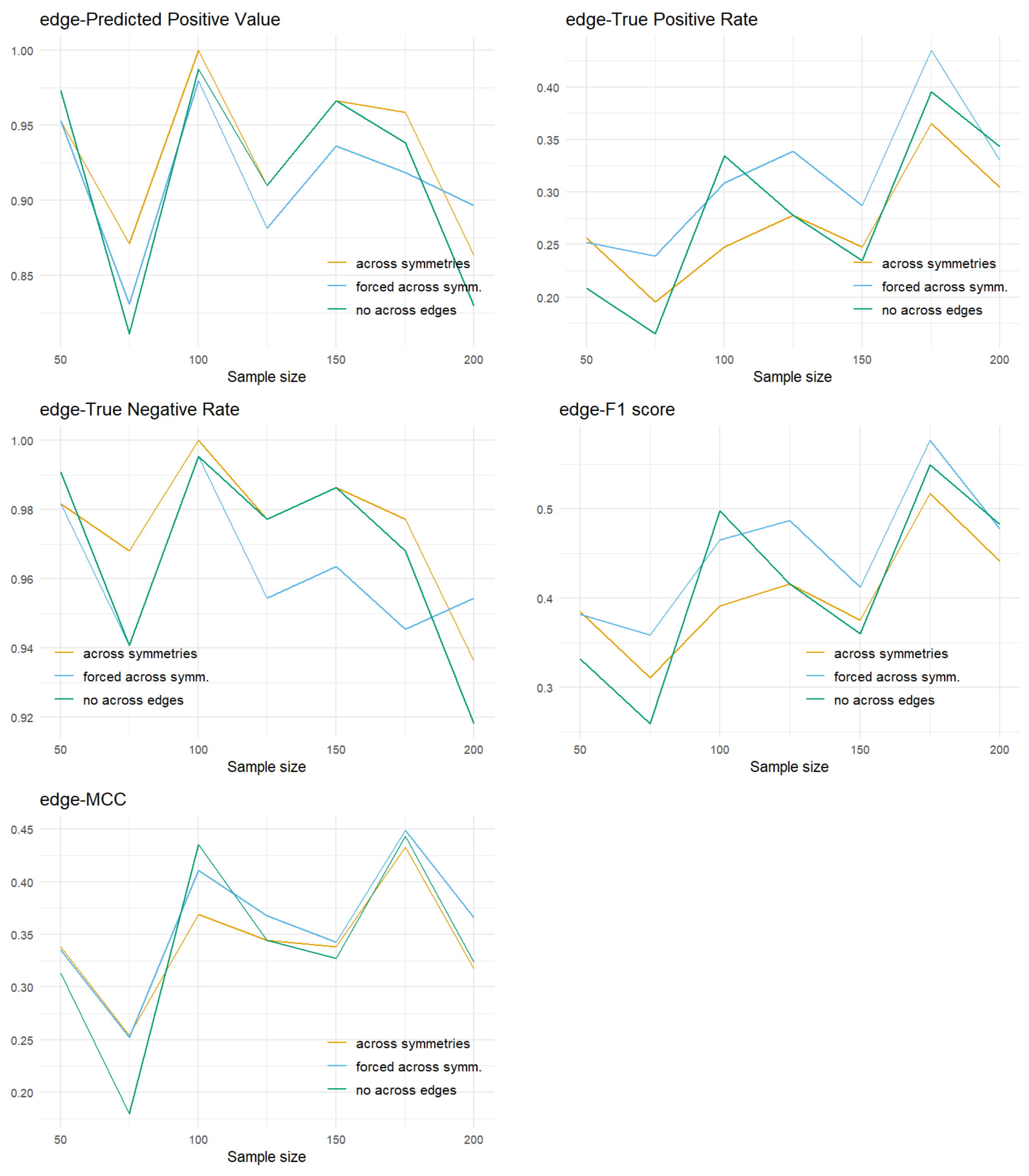


FIGURE D.5: Structure recovery measures of $\mathcal{G}_L$ and $\mathcal{G}_R$ for the high density, fully symmetric setting, when $\mathtt{p} = 50$.

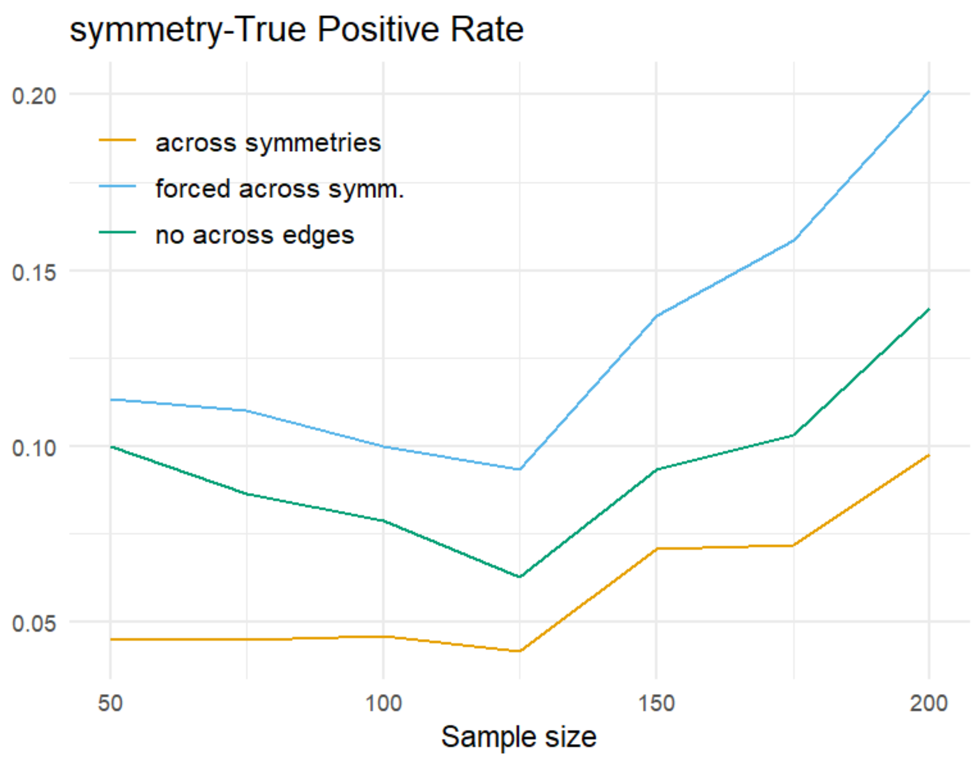


FIGURE D.6: Symmetry-True Positive rate of $\mathcal{G}_L$ and $\mathcal{G}_R$ for the high density, fully symmetric setting, when $\mathtt{p} = 50$.

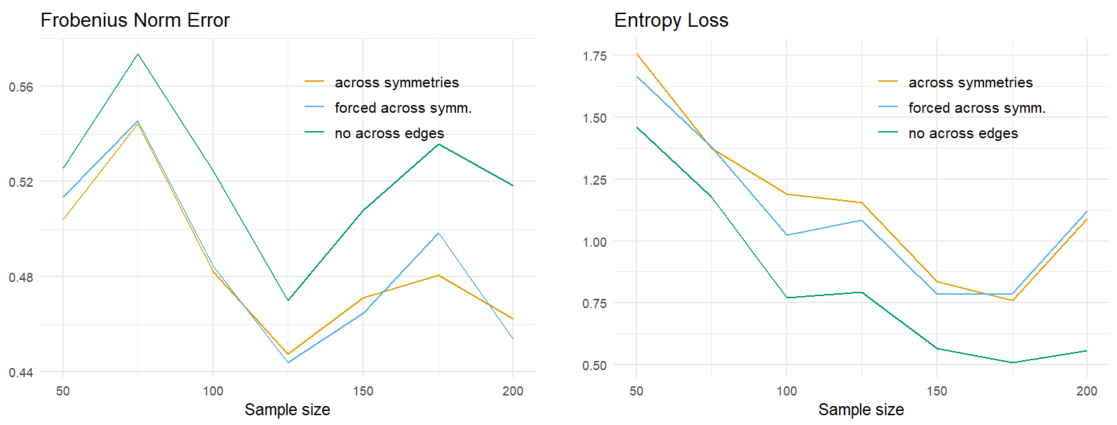


FIGURE D.7: Estimation accuracy measures of Θ_{LL} and Θ_{RR} for the high density, fully symmetric setting, when $\mathtt{p} = 50$.

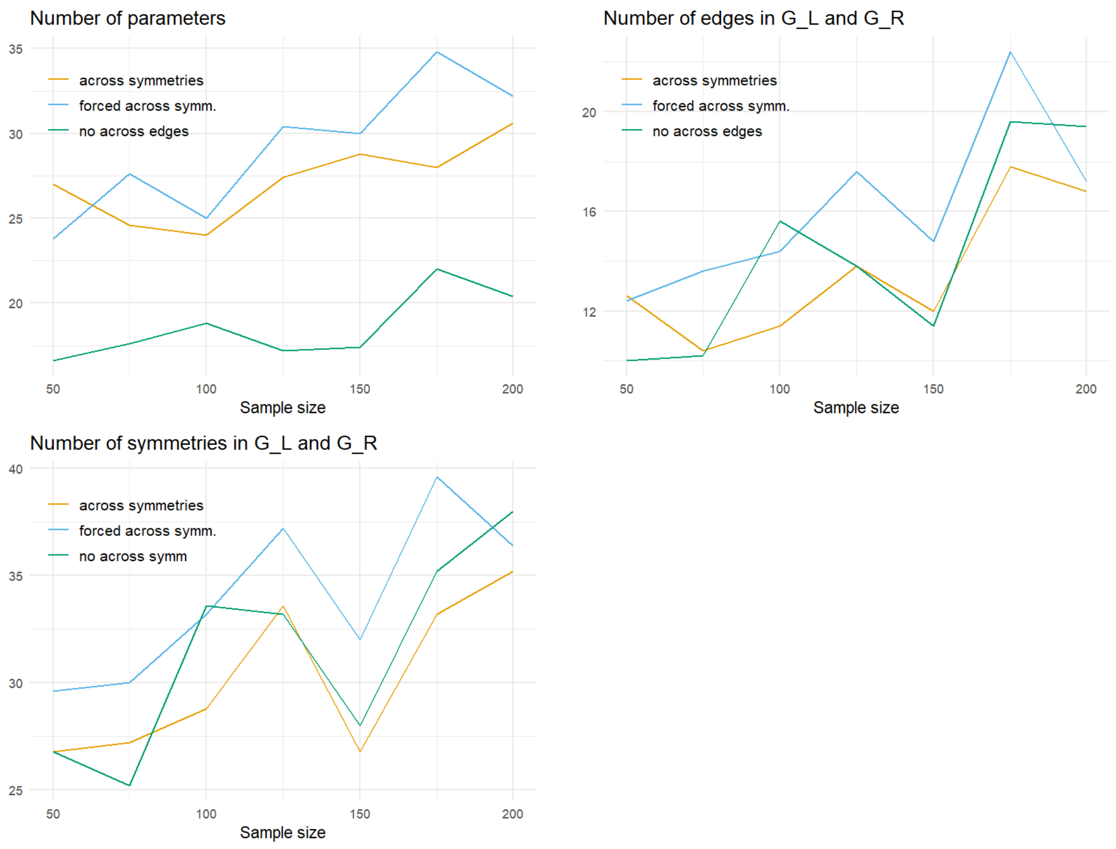


FIGURE D.8: Total number of parameters estimated and numbers of edges and non-zero parametric symmetries in $\mathcal{G}_L$ and $\mathcal{G}_R$ for the high density, fully symmetric setting, when $\mathtt{p} = 50$.

D.3 Low density, low symmetry setting

In the low density, low symmetry scenario forcing across-block symmetries leads once again to a slightly better edge-F1 score (Figure D.9) and has a higher number of symmetries in the two subgraphs (Figure D.12), with a better symmetry-True Positive Rate (Figure D.10), although there are no clear differences in terms of Frobenius norm loss when compared with the model having $\lambda_2^{(V)} = \lambda_2$ (Figure D.11).

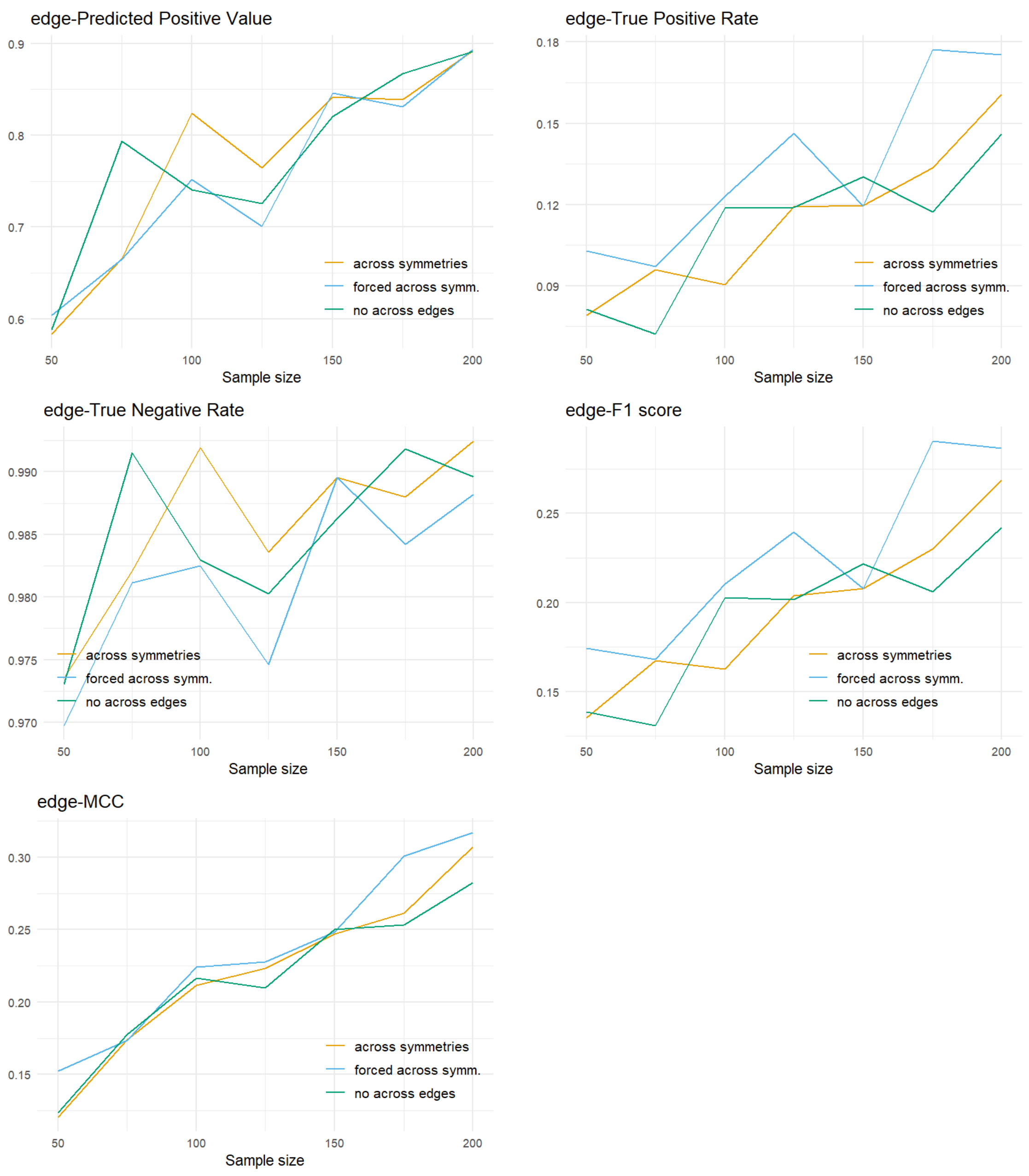


FIGURE D.9: Structure recovery measures of $\mathcal{G}_L$ and $\mathcal{G}_R$ for the low density, low symmetry setting, when $\mathtt{p} = 50$.

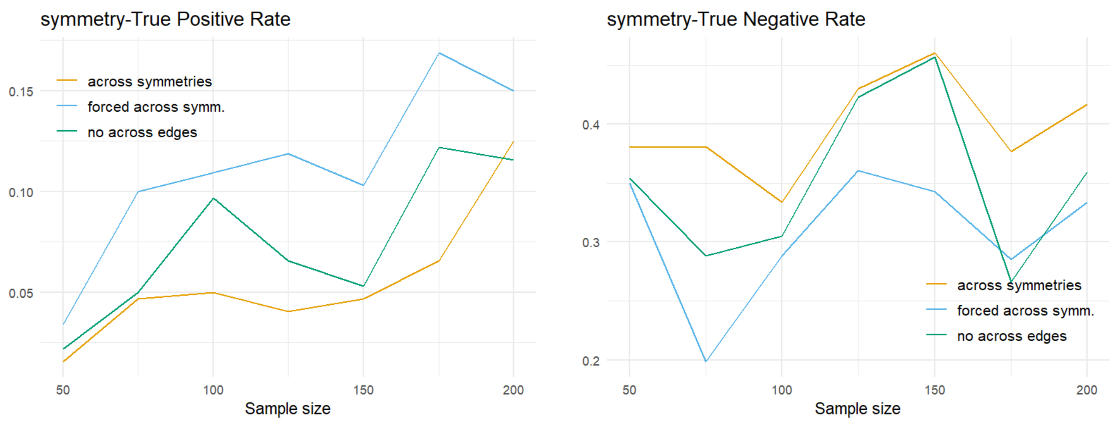


FIGURE D.10: Symmetric structure recovery measures of $\mathcal{G}_L$ and $\mathcal{G}_R$ for the low density, low symmetry setting, when $\mathtt{p} = 50$.

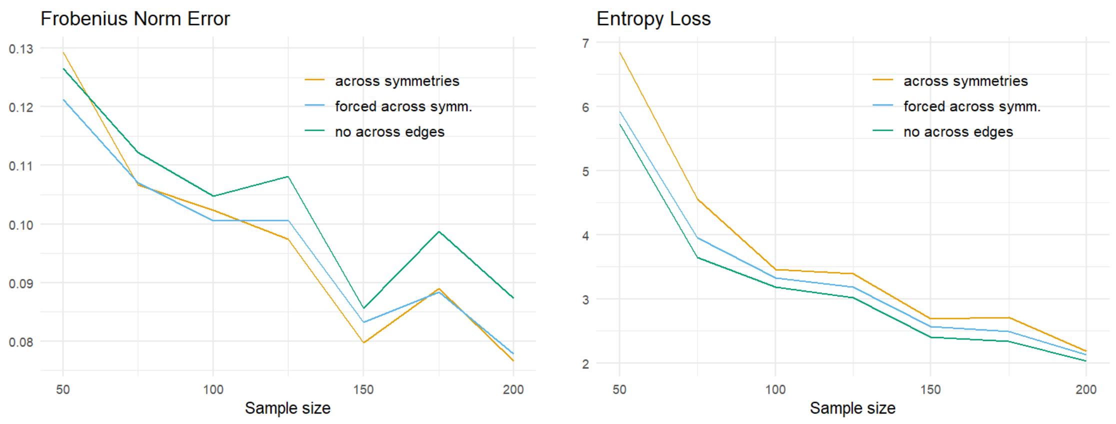


FIGURE D.11: Estimation accuracy measures of Θ_{LL} and Θ_{RR} for the low density, low symmetry setting, when $\mathtt{p} = 50$.

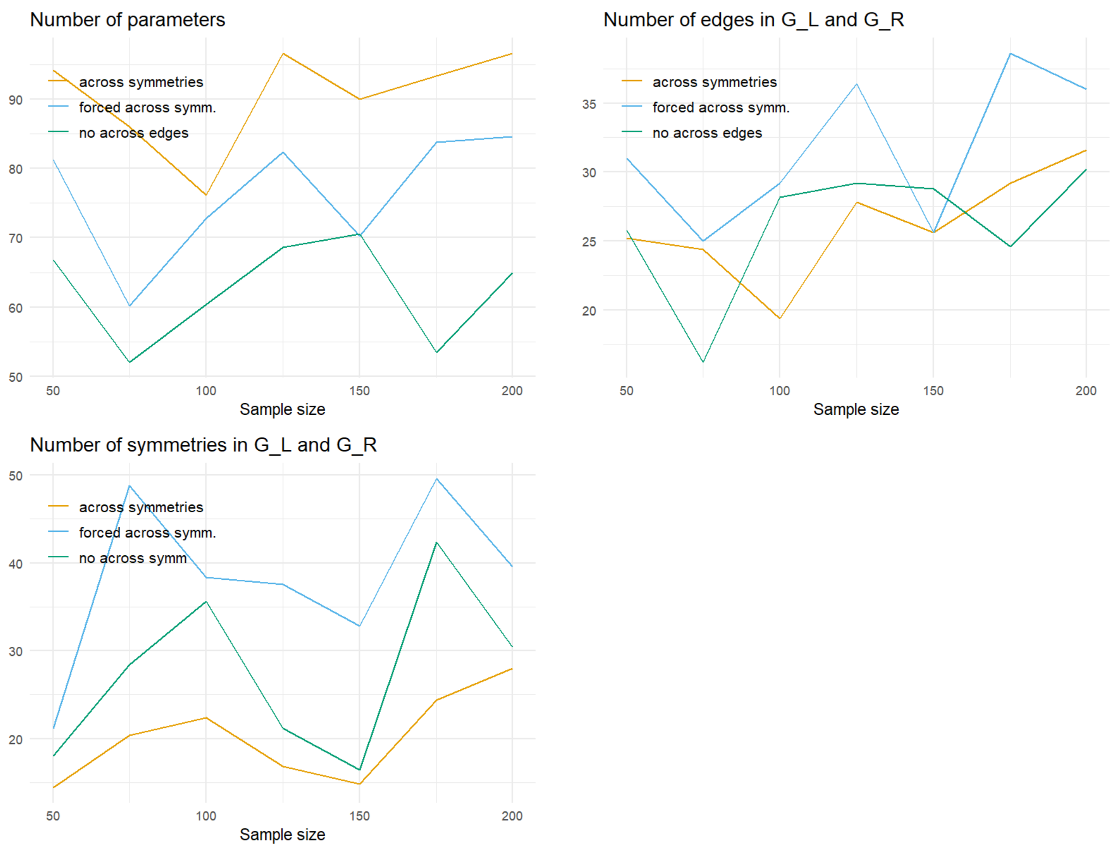


FIGURE D.12: Total number of parameters estimated and numbers of edges and non-zero parametric symmetries in $\mathcal{G}_L$ and $\mathcal{G}_R$ for the low density, low symmetry setting, when $\mathrm{p} = 50$.

Bibliography

Anderson, T. W. (1973). Asymptotically Efficient Estimation of Covariance Matrices with Linear Structure. *The Annals of Statistics* **1**, 135 – 141.

Banerjee, O., El Ghaoui, L. & d'Aspremont, A. (2008). Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data. *The Journal of Machine Learning Research* **9**, 485–516.

Boyd, S., Parikh, N., Chu, E., Peleato, B. & Eckstein, J. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning* **3**, 1–122.

Clifford, P. & Hammersley, J. M. (1971). Markov fields on finite graphs and lattices. Technical report, University of Oxford.

Cox, D. R. & Wermuth, N. (1993). Linear dependencies represented by chain graphs. *Statistical Science* **8**, 204–218.

Desikan, R. S., Ségonne, F., Fischl, B., Quinn, B. T., Dickerson, B. C., Blacker, D., Buckner, R. L., Dale, A. M., Maguire, R. P., Hyman, B. T., Albert, M. S. & Killiany, R. J. (2006). An automated labeling system for subdividing the human cerebral cortex on mri scans into gyral based regions of interest. *NeuroImage* **31**, 968–980.

Dey, D. K. & Srinivasan, C. (1985). Estimation of a covariance matrix under stein's loss. *The Annals of Statistics* **13**, 1581–1591.

Friedman, J., Hastie, T., Höfling, H. & Tibshirani, R. (2007). Pathwise coordinate optimization. *The annals of applied statistics* **1**, 302–332.

Friedman, J., Hastie, T. & Tibshirani, R. (2008). Sparse inverse covariance estimation with the graphical lasso. *Biostatistics* **9**, 432–441.

GAO, X. & SONG, P. X.-K. (2010). Composite likelihood bayesian information criteria for model selection in high-dimensional data. *Journal of the American Statistical Association* **105**, 1531–1540.

GEHRMANN, H. (2011). Lattices of Graphical Gaussian Models with Symmetries. *Symmetry* **3**, 653–679.

GOH, K.-I., CUSICK, M. E., VALLE, D., CHILDS, B., VIDAL, M. & BARABÁSI, A.-L. (2007). The human disease network. *Proceedings of the National Academy of Sciences* **104**, 8685–8690.

GRIMMETT, G. R. (1973). A theorem about random fields. *Bulletin of the London Mathematical society* **5**, 81–84.

HASTIE, T., TIBSHIRANI, R. & WAINWRIGHT, M. (2015). *Statistical Learning with Sparsity: The Lasso and Generalizations.* Chapman & Hall/CRC.

HØJSGAARD, S. & LAURITZEN, S. L. (2008). Graphical gaussian models with edge and vertex symmetries. *Journal of the Royal Statistical Society Series B: Statistical Methodology* **70**, 1005–1027.

HØJSGAARD, S. & LAURITZEN, S. L. (2007). Inference in graphical gaussian models with edge and vertex symmetries with the grc package for r. *Journal of Statistical Software* **23**, 1–26.

JENSEN, S. T., JOHANSEN, S. & LAURITZEN, S. L. (1991). Globally convergent algorithms for maximizing likelihood function. *Biometrika* **78**, 867–877.

KOSKI, T. & NOBLE, J. M. (2009). *Bayesian Networks: An Introduction.* Wiley Series in Probability and Statistics. Wiley, 1st ed.

LAURITZEN, S. (1996). *Graphical Models.* Oxford Statistical Science Series. Clarendon Press.

LETAC, G. & MASSAM, H. (2007). Wishart distributions for decomposable graphs. *The Annals of Statistics* **35**.

LI, D., LI, T., NIU, Y., XIANG, J., CAO, R., LIU, B., ZHANG, H. & WANG, B. (2019). Reduced hemispheric asymmetry of brain anatomical networks in attention deficit hyperactivity disorder. *Brain Imaging and Behavior* **13**, 669–684.

LI, Q., GAO, X. & MASSAM, H. (2016). Approximate bayesian estimation in large coloured graphical gaussian models.

Li, Q., Sun, X., Wang, N. & Gao, X. (2021). Penalized composite likelihood for colored graphical gaussian models. *Statistical Analysis and Data Mining: The ASA Data Science Journal* **14**, 366–378.

Lüdecke, D., Patil, I., Ben-Shachar, M. S., Wiernik, B. M., Waggoner, P. & Makowski, D. (2021). see: An R package for visualizing statistical models. *Journal of Open Source Software* **6**, 3393.

Massam, H., Li, Q. & Gao, X. (2018). Bayesian precision and covariance matrix estimation for graphical Gaussian models with edge and vertex symmetries. *Biometrika* **105**, 371–388.

Qiong Li, X. G. & Massam, H. (2020). Bayesian model selection approach for coloured graphical gaussian models. *Journal of Statistical Computation and Simulation* **90**, 2631–2654.

Ranciati, S. & Roverato, A. (2024a). On the application of gaussian graphical models to paired data problems. `https://arxiv.org/abs/2307.14160`, manuscript.

Ranciati, S. & Roverato, A. (2024b). *pdglasso: Graphical Lasso for Coloured Gaussian Graphical Models for Paired Data*. R package version 1.1.0.

Ranciati, S., Roverato, A. & Luati, A. (2021). Fused Graphical Lasso for Brain Networks with Symmetries. *Journal of the Royal Statistical Society Series C: Applied Statistics* **70**, 1299–1322.

Ravikumar, P., Wainwright, M. J., Raskutti, G. & Yu, B. (2011). High-dimensional covariance estimation by minimizing l1-penalized log-determinant divergence. *Electronic Journal of Statistics* **5**.

Rothman, A. J., Bickel, P. J., Levina, E. & Zhu, J. (2008). Sparse permutation invariant covariance estimation. *Electronic Journal of Statistics* **2**.

Roverato, A. & Nguyen, D. N. (2022). Model inclusion lattice of coloured gaussian graphical models for paired data. *Proceedings of The 11th International Conference on Probabilistic Graphical Models* **186**, 133–144.

Salvan, A., Sartori, N. & Pace, L. (2020). *Modelli Lineari Generalizzati*, vol. 124 of *UNITEXT*. Milano: Springer Milan.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B: Statistical Methodology* **58**, 267–288.

Varin, C., Reid, N. & Firth, D. (2011). An overview of composite likelihood methods. *Statistica Sinica* **21**, 5–42.

Wickham, H. (2016). *ggplot2: Elegant Graphics for Data Analysis.* Springer-Verlag New York.

Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L. D., François, R., Grolemund, G., Hayes, A., Henry, L., Hester, J., Kuhn, M., Pedersen, T. L., Miller, E., Bache, S. M., Müller, K., Ooms, J., Robinson, D., Seidel, D. P., Spinu, V., Takahashi, K., Vaughan, D., Wilke, C., Woo, K. & Yutani, H. (2019). Welcome to the tidyverse. *Journal of Open Source Software* **4**, 1686.

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**, 892–900.