Graph Recovery Methods

Fin Brown

his project investigates a selection of models used in recovery of graph structures. Both global and local (node-wise) approaches are used in order to determine the edge set of the true distribution. We explain and then examine the performance of the differing approaches under various simulation settings, high- and low-dimensional, sparse and non-sparse.

Contents

1. Introduction

In recent years the study of graphical models has faced increasing interest. A graphical model is a probabilistic model with conditional dependence structure among p random variables, $\mathbf{X} = (X_1, ..., X_p)^t$, expressed as p vertices (nodes) connected by a set of edges. Two types of graphical models exist: undirected networks (Markov networks) and directed networks (Bayesian networks). The former has no directional structure, meaning two nodes are independent given the remaining nodes. For the latter, edges are directional, meaning there is conditional dependency on other variables. Analysis of this project includes focus on undirected networks. The set of edges describe the conditional dependence structure of the p variables. It represents the covariance between X_i and X_j, conditional on the remaining variables X_k for $1 \leqslant k \leqslant p$ and $k \neq i, j$. Nodes i and j are connected by an edge if and only if their conditional covariance is non-zero. The edge set is thus defined as the set of variables that have non-zero conditional covariance. In this report we describe and analyse different approaches for recovering the edge set from realisations of an unknown distribution.

The target of an approach is to identify if a relationship exists between two nodes or not. LASSO regularization has become popular in undirected graphical estimation as a result of its sparse solutions. There are many LASSO-based approaches; they can be categorized as global methods and local (node-wise) methods. The former uses regularized log-likelihood to estimate the precision matrix, while the latter divides the estimation into a series of linear regressions by estimating the neighbourhood of each node in the underlying graph. The models assume the random variable $\mathbf{X}$ is taken from multivariate Gaussian distribution with mean μ and covariance matrix $\Sigma \in \mathbb{R}^{p \times p}$. If an entry of the covariance is zero, the corresponding nodes are conditionally independent. Otherwise, they are conditionally dependent. Hence, implementation of LASSO penalty can help to increase sparsity.

1.1. Node-Wise LASSO

Node-wise LASSO approach was proposed by Meinshausen and Buhlmann (2006), suggesting to take each variable as a response variable and then regress it on the remaining variables. Applying a LASSO approach in such regression helps to identify which coefficients should be zero and helps to obtain a sparse solution. After obtaining LASSO estimators for the regression coefficients of node i, we say nodes i and j are connected if the corresponding coefficient, β_j, is non-zero. For each pair of nodes i and j, we will thus have two estimates for their conditional dependence: β_j from the i^{th} regression and β_i from the j^{th} regression. From this we derive two methods to estimate the edge set: the set of pairs of i and j where both of the above βs are non-zero or the set where either are non-zero. We will call these two estimators $\hat{E}_1$ and $\hat{E}_2$ respectively. Full details are provided in the appendix.

1.2. Graphical LASSO

Graphical LASSO is a global approach developed by Friedman, Hastie & Tibshirani (2007). They suggested estimating sparse graphs by maximizing a LASSO penalized Gaussian log-likelihood. Given a precision matrix $\mathbf{\Theta} = \Sigma^{-1}$, it can be shown that two nodes (i and j) are connected if and only if corresponding value $\mathbf{\Theta}_{ij}$ is non-zero. Since the covariance matrix is symmetric, $\mathbf{\Theta}_{ij}$ and $\mathbf{\Theta}_{ji}$ are equal. The log-likelihood maximised by graphical LASSO is defined as:

$$\log \det \mathbf{\Theta} - \mathrm{Tr}(\mathbf{S}\mathbf{\Theta}) - \lambda \sum_{i \neq j} \Theta_{ij} \tag{1}$$

where $\mathbf{S}$ is the sample covariance matrix of $\mathbf{X}_1, \ldots, \mathbf{X}$, and λ is a regularization parameter controlling solution sparsity. By obtaining a sparse estimate of $\mathbf{\Theta}$, we can estimate the edge set. Fuller details of the mathematical approach are provided in the appendix.

2. Methodology

2.1. Model Tuning

As both models incorporate some regularization, both include a regularization parameter λ that needs to be tuned. In the node-wise model, lambda refers to the regularization constraint for each individual sub-LASSO: λ could either be consistent across the sub-models or we could tune for each (i.e. p separate values for λ). Under the settings considered here there is no reason to believe multiple λs will systematically improve the model. For the graphical model, λ refers to the graphical regularization constraint. We can roughly interpret these two hyper-parameters as equivalent to each other and we therefore optimise across the same search space.

In order to find the optimal hyper-parameter values we must first define a test error metric. For the graphical model this is more straightforward; we can calculate the log-likelihood of some test data under the precision matrix given by the model fitted on some training data:

$$\log \det \mathbf{\Theta} - \mathrm{Tr}(\mathbf{S}\mathbf{\Theta}) \tag{2}$$

choosing the value of λ that maximises this log-likelihood. Note this is equivalent to an unpenalised version of the log-likelihood in equation (1). It is slightly less straightforward for the node-wise model however. In the absence of an obvious test error metric we can instead use the average of the MSE for each sub-model, that is for each of the p sub-models we first fit the model and then calculate the MSE of prediction against some test data and average across the p sub-models. Whilst this metric is far from ideal, it should suffice for the purpose of parameter tuning. A mathematical definition is given in the appendix.

For both models we use 3-fold cross-validation to obtain robust estimates of the test error and ensure a good selection for the tuning parameter.

At this point it is important to clarify that parameter tuning must be done in the absence of the true labels: any tuning method using these true values would result in an upwardly biased model performance (and in practice the true labels would be unknown anyway). Once the models have been correctly tuned we can compare their performance using a number of classification error metrics.

2.2. Model Validation and Comparison

We compare model performance in distinct ways. The first of these is by plotting the ROC curve for each model family indexed over a series of values for λ; for each model we extract the True Positive Rate (TPR) and False Positive Rate (FPR) and plot them against each other. Plotting multiple curves on one set of axes allows us to compare their performance. We also calculate the Area Under the ROC curve (AUROC), a metric we can use for quantitative comparison of performance. We compare these curves across a variety of simulation settings, details of which are discussed later on.

Second, we further compare performance of the models by tuning them for λ and then validating them against the true labels, as previously described in detail. We repeat the simulation 50 times, each time calculating TPR, FPR, Mean Misclassification Error (MMCE), Precision and F1 score. Mathematical definitions are given in the appendix. Precision tells us the percentage of positive predictions that are correct and is often preferred if the outcome is dominated by negative results. F1 score is a harmonic average of precision and handles imbalanced settings well. We store the results from each simulation and plot the results. These are presented and discussed in the results section.

2.3. Simulation Settings

There are a number of simulation parameters to consider, their values significantly impact the performance of the models. These include:

- p: number of nodes in the simulated graph
- δ: the magnitude of scalar addition to the theta diagonal
- p_e: probability two nodes are connected by an edge
- n: the number of simulated data points

There are a number of distinct cases we need to consider when assessing the performance of the models. By concurrently varying p and n we test performance under all the combinations resulting from n=50, 100 and p=25, 50, 75; this means we consider both high- and low-dimensional problems. As both models provide sparse estimates they should both be able to cope with high-dimensionality; this is examined further in the results section. In high dimensional settings sample covariance estimates are unstable and the resulting matrix $\mathbf{S}$ is not full rank. Under a low rank, and thus singular, $\mathbf{S}$ many models are unable to compute estimates. Both families of models considered here are able to compute estimates with a singular $\mathbf{S}$ as a result of the LASSO penalty term(s).

Whilst keeping p and n constant we vary the remaining parameters to test performance under other different settings. Unless stated otherwise, we compute $\mathbf{\Theta}$ using the smallest δ (as a multiple of 0.1) possible to guarantee $\mathbf{\Theta}$ is semipositive definite. Varying the value of delta has important implications on the prediction problem: as delta increases the signal to noise ratio will diminish, resulting in decreasing model performance. Thus by using a small delta we compare the models under the setting most conducive to predictive performance.

We can also vary the sparsity of the simulated data. This is dependent on p_e, the probability two nodes are connected by an edge. Unless stated otherwise, all the anaylsis in this project is done under the setting where $p_e = 0.1$. We will consider other cases, however, where p_e=0.25, 0.5. The implication of this on the classification task is that it is transformed from an imbalanced problem to a balanced problem, i.e. the output classes (1 or 0 depending on whether two nodes share an edge) become populated in roughly equal numbers.

The outcomes under each of the simulation settings described above are presented in the results section.

3. Results

Figure 1 displays ROC curves for various p and n settings. It is difficult to make any significant inferences from these plots and the values of the AUROCs alone. In a few of these settings the curves look particularly discontinuous (figures 1.1, 1.4) or are incomplete figure (1.3). In settings with small p there will be fewer positive results for the model to predict: for a model with 25 nodes there are 300 potential edges and thus we expect around 30 positive results for the model to predict, resulting in TPRs that jump between discrete levels. Incomplete curves arise in high-dimensional settings. Where rough comparisons between the models can be made, the best performing model seems to vary of the TPR range. In most cases, if not all, the node-wise model E2 outperforms for low TPR values with the other node-wise model E1 outperform for higher values of TPR. The graphical model E3 performs somewhere in between these two over the majority of the TPR range in all cases. Whilst the AUROCs are also displayed, care must be taken in using them as they may be unrepresentative due to incompleteness (in particular many of the curves don't span the full $[0,1]^2$ space). Many of the above issues arise due to complications resulting from imbalance in the outcome labels. This is discussed in more detail later; performance in a balanced setting is also examined.

Figure 2 shows boxplots of the performance metrics resulting from the model validation described earlier. The results are also displayed in table 1. As relative performance between the models seems roughly constant across differing settings of p and n, we only analyse these results for one setting (n=100, p=50). Interestingly the same model rankings can be seen across the three figures; it should be noted that two of the plots display error metrics (figures 2.1, 2.3) whist the other displays a positive metric (figure 2.2). This implies that whilst the graphical model E3 identifies a higher proportion of the positive results it also misidentifies more negatives as positives; on the other hand, the node-wise model E2 identifies the lowest proportion of positive results but also misidentifies the fewest negatives.

Results for performance in other settings is displayed in figure 3. All experiments here are run under the setting with n=100, p=50 and performance is compared only for the graphical model E3. As trends in relative performance between the models seem to be roughly flat, trends in performance for the graphical model E3 across the settings displayed in figure 3 can be generalised to the node-wise models (though this is discussed briefly below). The subplots show the relationships between the different performance metrics and the value of delta (as described in the simulation settings section). We compare over a discrete set of δ values (lowest, 4, 6, 8, 10) where lowest refers to the lowest possible delta (as a multiple of 0.1) averaged over the set of simulations. Note this is significantly lower for a sparse theta: from Gershgorin's theorem we know delta must be larger than the sum of non-diagonal entries for each row and the sparser a matrix the smaller the sum of non-diagonal entries.

As mentioned above, relative model performance seems to be insensitive to the simulation settings. Figure 4 displays ROC curves for the same p, n setting with different sparsity in the simulated precision matrix; it is important to observe that the relative the relative performance of the models seems roughly consistent (i.e. the behave similarly relative to each other in different sparsity settings). As mentioned above, this is what enables us to extend inferences about the graphical model E3 under varying settings to the node-wise models E1 and E2. Figure 4 also shows us that performance, as measured by AUROC, is actually better in the imbalanced setting (i.e. sparser setting). Figure 3 supports this observation.

4. Concluding Remarks

All three of the models have similar performance with none dominating another. Purely in terms of precision matrix recovery one might choose to utilise a model that fits the specific characteristics desired, i.e. graphical model E3 if high TPR is the priority or node-wise model E2 if FPR is to be minimised. It is likely, however, that computational speed will be considered when using these methods practically. The graphical model E3 significantly outperforms the node-wise models in terms of computational efficiency, scaling much more efficiently.

A. Appendix

A.1. Node-Wise model formulation

The node-wise model consists of p sub-models, each a LASSO regression with one of the nodes as target and the remaining $p-1$ as covariates. Let us call the LASSO estimate of the regression coefficient for node j regressed against node i as β_{ij}.

From the p regressions we then obtain the matrix β with entries β_{ij}. Note β is non-symmetric. From β we can obtain two different estimators for the edge set E:

- $\hat{E}_1$: $\{(i,j) : \text{for } \beta_{ij} \neq 0 \text{ and } \beta_{ji} \neq 0\}$
- $\hat{E}_2$: $\{(i,j) : \text{for } \beta_{ij} \neq 0 \text{ or } \beta_{ji} \neq 0\}$

For cross-validation we take as our test metric the average of the test MSE from the p sub-regressions:

$$M\bar{S}E = \frac{1}{p}\sum_{i=1}^{p} MSE_i \tag{3}$$

where MSE_i is the test error obtained for the regression of node i (trained on a training set and tested on a testing set).

A.2. Graphical model formulation

The graphical model is a global approach that sparsely estimates the precision matrix $\mathbf{\Theta}$. From here we can obtain the estimator for the edge set E:

- $\hat{E}_3$: $\{(i,j) : \text{for } \Theta_{ij} \neq 0\}$

For cross-validation we use the log-likelihood of the test data under the precision fitted on some training data, equation (2) described earlier.

A.3. Performance Metrics

We discuss various performance metrics in the paper. Here we will provide their mathematical definitions.

- MMCE: $\frac{1}{|E_{i,j}|}\sum_{i\neq j}(E_{i,j} \neq \hat{E}_{i,j})$
- TPR: $\frac{TP}{TP+FN}$
- FPR: $\frac{FP}{FP+TN}$
- Precision: $\frac{TP}{TP+FP}$
- F1 score: $\frac{2TP}{2TP+FP+FN}$

Where FP, FN, TP, TN refer to false positives, false negatives, true positives and true negatives respectively.

A.4. Figures and Tables

Table 1: Model performance under p=50, n=100 setting

Model	MMCE	TPR	FPR	Precision	F1
$\hat{E}_1$	0.16 $\pm$0.02	0.95 $\pm$0.03	0.18 $\pm$0.02	0.38 $\pm$0.03	0.53 $\pm$0.03
$\hat{E}_2$	0.07 $\pm$0.01	0.89 $\pm$0.04	0.08 $\pm$0.01	0.57 $\pm$0.04	0.68 $\pm$0.04
$\hat{E}_3$	0.23 $\pm$0.02	0.96 $\pm$0.02	0.25 $\pm$0.03	0.30 $\pm$0.03	0.45 $\pm$0.03

Fig. 1: ROC curves for a variety of settings

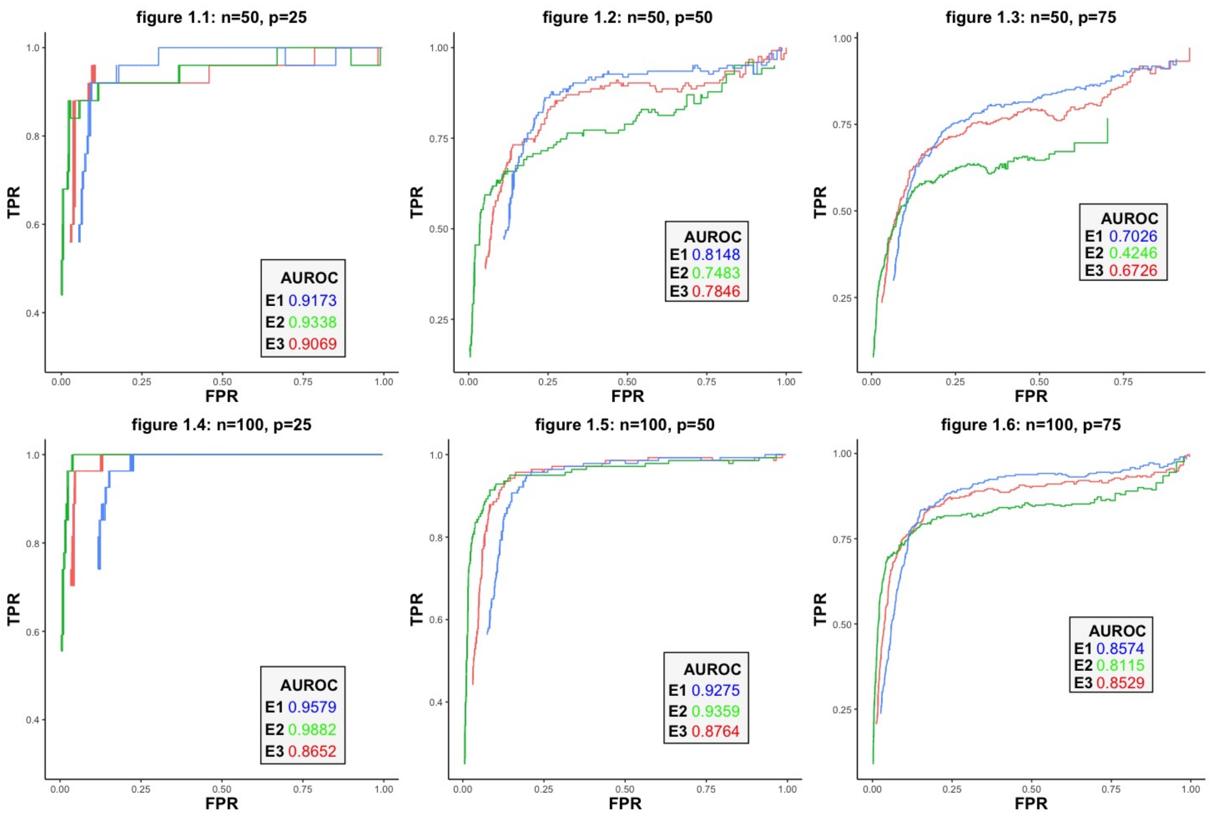


Fig. 2: Various Performance Metrics for Tuned Models

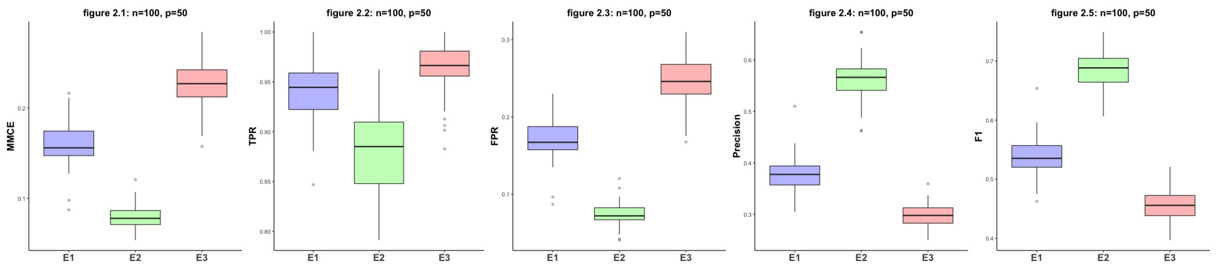


Fig. 3: Various Performance Metrics for Tuned E3 under a variety of settings

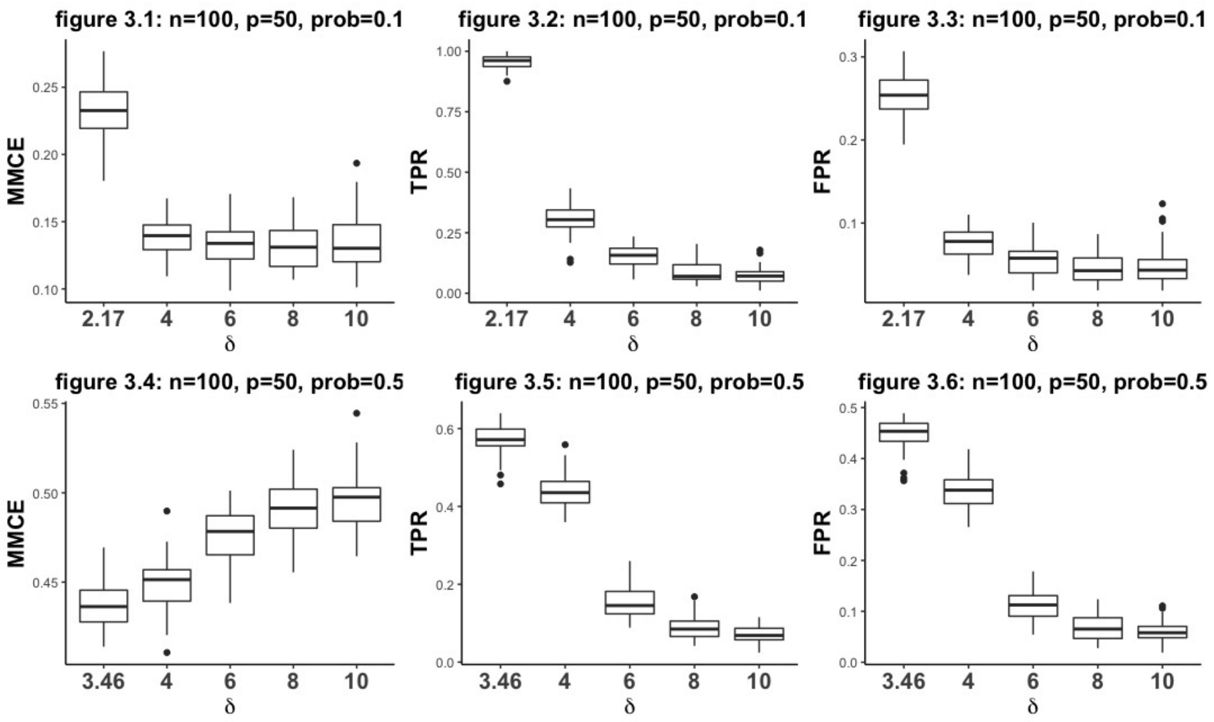
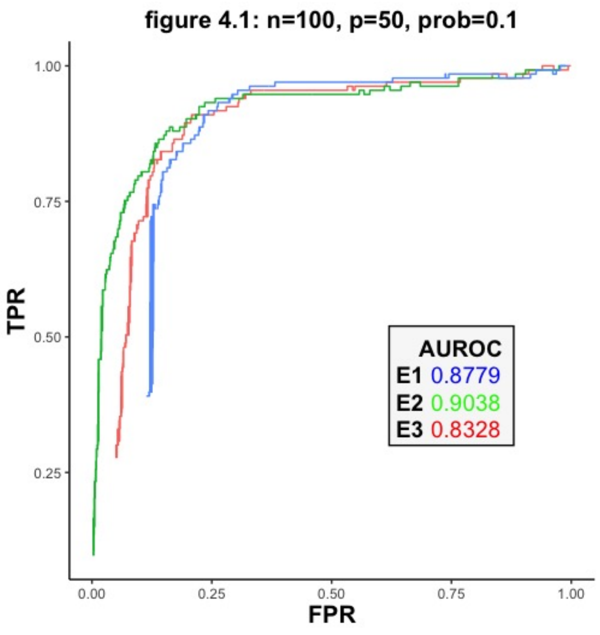
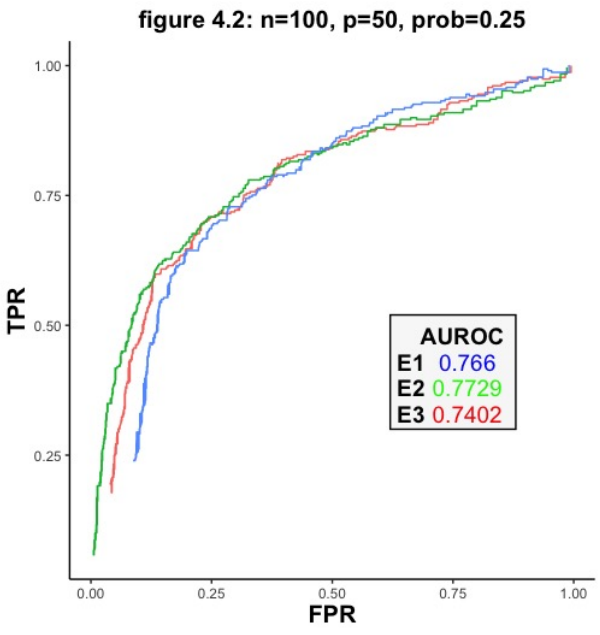


Fig. 4: ROC curves the models under different sparsity settings





References

Meinshausen, N. & Buhlmann, P. (2006), 'High dimensional graphs and variable selection with the lasso', Annals of Statistics 34, 1436–1462

Friedman, J., Hastie, T. & Tibshirani, R. (2007), 'Sparse inverse covariance estimation with the graphical lasso'

Friedman, J., Hastie, T. & Tibshirani, R. (2010), 'Applications of the lasso and grouped lasso to the estimation of sparse graphical models'

Jankova, J., Sara van de Geer (2018), 'Inference in high-dimensional graphical models'

Xinghao Qiao, Shaojun Guo, and Gareth M. James (2019), 'Functional Graphical Models'

Friedman, J., Hastie, T., & Tibshirani, R. (2018). 'Glasso: Graphical lasso for R [Computer Software]'.