university of groningen / faculty of science and engineering / mathematics and applied mathematics

Cross-comparison of inference methods for Gaussian Graphical Model

Bachelor's Project Mathematics

July 2018

Student: S.T. Schotanus

First supervisor: dr. M.A. Grzegorczyk

Second assessor: Dr. W. Krijnen

Abstract

In Gaussian graphical modeling statistics and graph theory go hand in hand. With applications in many fields this widely applied technique is used to estimate the partial correlations among variables of a given dataset. This thesis covers four of the popular approaches within Gaussian graphical modeling, the Moore-Penrose pseudoinverse, the Lasso (inclusion of the ℓ_1-term), the shrinkage and the bootstrap approach. These four methods will be cross compared on the basis of several tests. These tests cover the accuracy of the estimation, the computational speed and the user friendliness of each method. The first test is based on randomly generated datasets of which the true partial correlations are know. Every tested method is then applied to two datasets extracted from the *R* package 'GeneNet'. In the last section of this thesis it is concluded that, despite a high demand on the computational cost, the Lasso approach supplies the user with the most accurate estimation. However it has to be noted that it is suspected that the shrinkage approach did not achieve its full potential due to mistakes made along the journey of programming.

Contents

1 Introduction

When interested in the partial correlations of variables of a given dataset one needs to construct a covariance matrix. However when dealing with a dataset with a small number of samples ,n, and a large number of variables ,p, a reliable estimate of the covariance matrix has to be used. This problem is adressed by Gaussian graphical modeling. However, within Gaussian graphical modeling there are many methods available. This large variety of alternatives may cause the researcher in question to lose overview. These different approaches show a large difference in the accuracy of the results is observed when these techniques are applied. This thesis is thus meant to help researchers choose a method that fits the needs of their research. Out of the available methods within Gaussian graphical modeling four are described and tested. This concerns the Moore-Penrose pseudoinverse method proposed by Penrose [3], the lasso (with inclusion of the ℓ_1-term) applied according to the proposed strategy of Friedman et al. [2], the shrinkage approach proposed by Schäfer [5] and the bootstrap method which is suggested by Breiman [1].

In the first section a brief introduction to graph theory is given. The next sections will give an introduction to the general technique of Gaussian graphical modeling, and give a description of the theory that drives each of the four methods subject to the comparison test. Once these preliminaries are treated each of the methods is applied in *R*. Each method is first applied to several generated datasets which are multivariate normally distributed and of which the true partial correlations are known. The results of these applications are then used in the comparison test, the core of this thesis. In order to show the capabilities of Gaussian graphical modeling each method is also applied to a real E.coli dataset.

In the last section the results of the cross comparison are used to draw conclusions on the accuracy, the computing time and the user friendliness of each method. These conclusions are then summarized in an overall conclusion.

In the appendix the reader is supplied with the code that is used to genarate a random dataset with the true partial correlations. This appendix also contains the code that is used to generate the graphs according to each method.

2 Graph theory

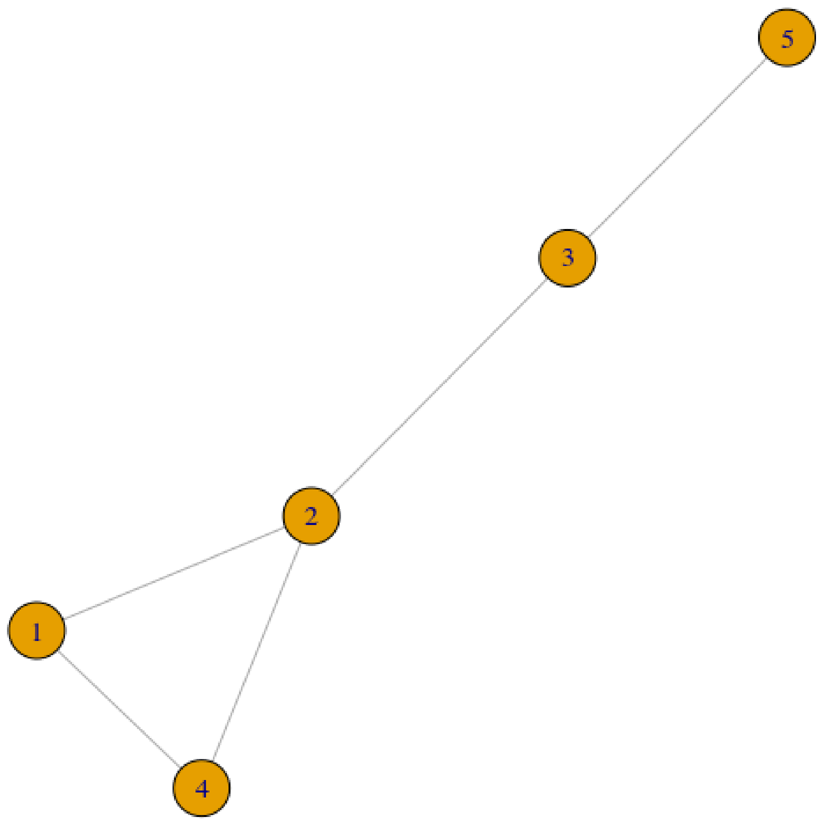

Figure 1: A visual representation of the graph $G(V, E)$ with $V = \{1, 2, 3, 4, 5, \}$ and $E = \{\{1, 2\}, \{1, 4\}, \{2, 3\}, \{2, 4\}, \{3, 5\}\}$

2.1 Introduction

A graph is a set of points, called nodes, connected by lines, called edges. The resulting graph is then represented in the form $G(V, E)$, where V is the set of nodes and E consists of subsets of 2 elements. The vertices in a graph may be constrained by a direction as well. However since the dependency between variables is symmetric we are only interested in undirected graphs. So the nodes are either connected or they are not, no direction is given to the edges.

Example 1. The image on top of this page visualizes the graph $G(V, E)$ with $V = \{1, 2, 3, 4, 5\}$ and $E = \{\{1, 2\}, \{1, 4\}, \{2, 3\}, \{2, 4\}, \{3, 5\}\}$. Note that for every set in E $\{a, b\} = \{b, a\}$ for all $a, b \in \mathbb{N}$ with $a \neq b$.

2.2 Graph theory in Gaussian graphical modeling

The example shown in the previous subsection lists 5 edges between 5 different nodes. In Gaussian graphical modeling these nodes represent variables and the edges indicate partial correlations between these variables. For the construction of a Gaussian graphical model either the definition of conditional indendence of the definition of partial correlation is used. The remainder of this section focusses on the general technique based on the conditional independency between variables.

Definition 2.1 *Two variables X_i and X_j are conditionally independent if:*

$$f(X_i, X_j) = f_{X_i}(X_i) f_{X_j}(X_j)$$

where $f(X_i, X_j)$ is the joint probability density function and $f_{X_i}(X_i)$ and $f_{X_j}(X_j)$ denote the marginal probability density function for X_i and X_j respectively. The following notation will be used to denote the dependency between variables

$$X_i \perp\!\!\!\perp X_j | X_{V\setminus\{i,j\}}$$

Where the the set $X_{V\setminus\{i,j\}}$ is defined as $X_{V\setminus\{i,j\}} = \{X_1, \ldots, \hat{X}_i, \hat{X}_j, \ldots, X_p\}$, where the entries fitted with a hat are omitted.

In the following definition the relation between graph theory and the conditional dependencies is made

Definition 2.2 *For the set $X = (X_1, \ldots, X_p)$ the graph $G = (V, E)$ with $V = \{1, \ldots, p\}$ is determined using the following relationship*

$$\{i, j\} \notin E \Leftrightarrow X_i \perp\!\!\!\perp X_j | X_{V\setminus\{i,j\}}$$

In the following section the general technique to Gaussian graphical modeling using the definition of partial correlation will be treated.

3 Gaussian Graphical Modeling

As mentioned before, Gaussian graphical models are accompanied by undirected graphical models. The corresponding undirected graphical model depicts the partial correlation between two variables. Gaussian graphical modeling is commonly used within genetics because the models are able to deal with datasets with small number of samples, n, and large numbers of variables, p, i.e. $n \ll p$. A sample which is always considered to be multivariate normally distributed, i.e.

$$\{X_1, ..., X_p\} \sim \mathcal{N}(\mu, \Sigma)$$

When dealing with datasets conform these characteristics one may have to rely on an estimation of the covariance matrix. In the next subsection the definition of the covariance matrix and its empirical estimate are given, either one of these matrices or another estimate is used in the construction of a Gaussian graphical model. Section 4 elaborates on the other techniques that could provide an estimate of the covariance matrix. In subsection 3.2 the reader is provided with an example.

3.1 Preliminaries

The next definitions elaborate on the covariance matrix and its empirical estimation used in the shrinkage approach. The first definition relies on the definitions of variance and covariance, definition 3.3 subsequently relies on the covariance matrix that can be constructed using this first definition.

Definition 3.1 (Covariance matrix) *The covariance matrix is constructed as follows:*

$$\Sigma = \begin{pmatrix} \mathrm{Var}(X_1) & \mathrm{Cov}(X_1, X_2) & \dots & \mathrm{Cov}(X_1, X_p) \\ \mathrm{Cov}(X_2, X_1) & \mathrm{Var}(X_2) & & \\ \vdots & & \ddots & \\ \mathrm{Cov}(X_p, X_1) & & & \mathrm{Var}(X_p) \end{pmatrix}$$

One type of estimation for this covariance matrix, called the empirical covariance matrix, is defined as described in the next definition

Definition 3.2 *According to the following formula an empirical covariance matrix can be calculated.*

$$S = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{\mu})(x_i - \hat{\mu})^T \quad \text{with} \quad \hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Note that this matrix relies on means instead of the expected values.

Now in order to determine which elements are partially correlated we introduce the following definition:

Definition 3.3 (Partial correlation matrix) *The partial correlation matrix* Π *is determined by the elements of the concentration matrix* $\Omega = \Sigma^{-1}$.

$$\Pi = (\pi_{ij}) = \frac{-\omega_{ij}}{\sqrt{\omega_{ii}\omega_{jj}}} \tag{1}$$

where the element ω_{ij} *is the entry in the* i^{th} *row and the* j^{th} *column of the correlation matrix.*

Either the covariance matrix or the empirical covariance matrix are used in the construction of this partial correlation matrix. The following section elaborates on the construction of a Gaussian graphical model using this data.

3.2 General technique to Gaussian graphical modeling

The subsets of E of the graph $G(V, E)$ are then determined by the non-zero elements of the correlation matrix Π. This is done by introducing a threshold to required strength of the partial correlation. In this study this threshold is set such that the number of estimated edges for each method is equal. More on this in section 6.

The following example brings this into practice:

Example 1. Consider the following partial correlation matrix Π

$$\Pi = \begin{pmatrix} 1 & 0.78 & 0.10 & 0.96 & 0.31 \\ 0.78 & 1 & 0.70 & 0.57 & 0.36 \\ 0.10 & 0.70 & 1 & 0.18 & 0.92 \\ 0.96 & 0.57 & 0.18 & 1 & 0.56 \\ 0.31 & 0.36 & 0.92 & 0.56 & 1 \end{pmatrix}$$

The $(i, j)^{th}$ shows the partial correlation between the variables X_i and X_j. So, as one can observe, for every $i, j \in 1, \ldots, p$ we have that $\Pi_{ij} = \Pi_{ji}$. Now in order the draw 5 edges we have to introduce a threshold of 0.57. With this threshold edges are drawn between the i^{th} and the j^{th} node if and only if $\Pi_{ij} \geq 0.57$. Leaving us with the graph shown in figure 2

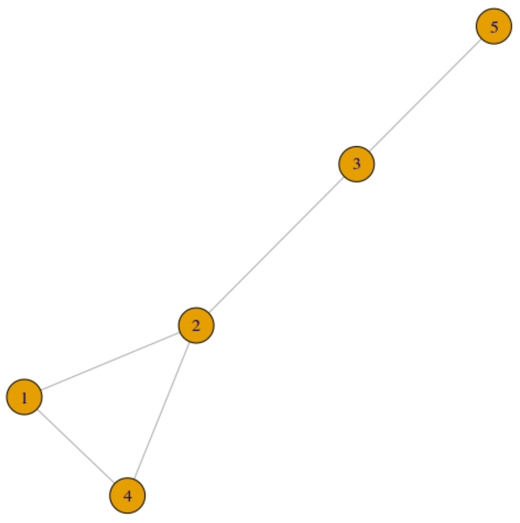

Figure 2: Graph corresponding to the correlation matrix Π

4 Selected methods

Various authors, Breiman [1], Penrose [3], Schäfer et al. [5] and Zhou et al. [7] have proposed different methods in order to determine an estimator. Each of these strategies differ in their approach and hence in several characteristics as well. Schäfer even suggests to combine the bootstrap and the Moore-Penrose approaches to get, depending on the dataset, more accurate results [4]. However, in order to truly test each method on its own performance the methods won't be combined in this thesis. The next subsections treat a selection of the most widely used methods below.

4.1 The Moore-Penrose pseudoinverse method (PINV)

The PINV method is based on the standard value decomposition (SVD). For this method the approach of Stifanelli [6] is used. This method is based on the paper by Penrose [3]. Stifanelli suggests to use the standard value decomposition to decompose the covariance matrix in order to construct an inverse for this matrix.

This decomposition and it's pseudoinverse are related in the following way:

$$\Sigma = U\Delta V^T$$

where, since $\Sigma \in \mathbb{R}^{p\times p}$, the matrices U, Δ and $V \in \mathbb{R}^{p\times p}$

$$\Sigma^+ = V\Delta^+ U^T$$

Note that in this equation the inverse is denoted by a $+$sign instead of a $-$sign. This follows from the way in which this generalized inverse is calculated. The second matrix of the decomposition, Δ^+, is determined by transposing the original matrix Δ and then replacing all non-zero elements of the diagonal by it's reciprocal.

This pseudoinverse or generalized inverse Σ^+ is then used as our estimation for Ω, denoted by $\Omega^M = (\omega_{ij}^M)$. From this we to construct the correlation matrix Π^M as such:

$$\Pi^M = (\pi_{ij}^M) = \frac{-\omega_{ij}^M}{\sqrt{\omega_{ii}^M \omega_{jj}^M}}$$

Note that the subscript 'M' in Π^M is used to show that this partial correlation matrix is determined with the use of the Moore-Penrose method. Which in turn is then used to plot the corresponding graph.

4.2 Lasso (inclusion of the ℓ_1-penalty-term)

Least absolute shrinkage and selection operator (lasso) is a regression analysis method that regularizes the data points using the other data as predictors. The process of regularization uses additionally introduced information to solve an ill-posed problem, a process which helps us in finding an inverse of the estimated covariance matrix.

Zhou et al. [7] have included a brief summary of the standard lasso procedure in their paper. The multivariate normally distributed dataset $X_1, ..., X_p$ is assumed, without loss of generality, to have a mean equal to zero. Each point in this dataset is then regressed versus its counterpoints $\{X_k | i \neq k\}$:

$$X_i = \sum_{j\neq i} \beta_j^k X_k + V_k$$

where V_i follows the same distribution as the dataset, i.e. a multivariate normal distribution with mean zero. The constant in this sum is defined as $\beta_j^i = -\frac{\omega_{ij}}{\omega_{ii}}$. Zhou et al. then define $\beta_{min} = \min_{i,j} |\beta_j^i|$. The non-zero entries $|(\omega_{ij})|$ are required to be upper bounded by β_{min}. By assuming $(\omega_{ii}) = 1$ and our definition of β_{min}, an estimated covariance matrix can be constructed. This method reduces the variance of a given dataset.

To make an even more accurate estimation, Friedman et al.[2] propose to include the lasso penalty term. Friedmans definition of the Lasso estimate of Ω . This definition includes the ℓ_1 penalty term.

$$\Omega_L = \max_{\Omega}(\log(\det(\Omega)) - \operatorname{tr}(S\Omega) - \lambda\|\Omega\|_1) \tag{2}$$

Where the term $\|\Omega\|_1$ is the ℓ_1-term, λ is the penalty parameter and the matrix S in this equation represents the empirical covariance matrix defined as:

$$S = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{\mu})(x_i - \hat{\mu})^T \quad \text{with} \quad \hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

The penalty parameter λ is determined by iteration. Meaning that the algorithm is run several times trying to determine the best value for λ. The next equation shows how the partial correlation matrix corresponding to this method is determined. Note that the 'L' relates the partial correlation matrix Π^L to the Lasso method.

$$\Pi^L = (\pi_{ij}^L) = \frac{-\omega_{ij}^L}{\sqrt{\omega_{ii}^L\omega_{jj}^L}} \tag{3}$$

Friedman et al. also include the algorithm for determining the estimated covariance matrix. This is done in the following way:

Algorithm 4.1 (Lasso algorithm)

1. *Take the following estimate for the covariance matrix Σ: Let $W = S + \lambda I$, where I denotes the identity matrix with the appropriate dimension*
2. *For all $j = 1, 2, \dots, p$ solve:*

$$\hat{\beta} = \min_{\beta}\left(\frac{1}{2}\|W_{11}^{\frac{1}{2}}\beta - W_{11}^{-\frac{1}{2}}s_{12}\|^2 + \lambda\|\beta\|_1\right) \tag{4}$$

The following block coordinates for the matrices S and W are used:

$$W = \begin{bmatrix} W_{11} & w_{12} \\ w_{12}^T & w_{22} \end{bmatrix}, \qquad S = \begin{bmatrix} S_{11} & s_{12} \\ s_{12}^T & s_{22} \end{bmatrix}$$

The output, $\hat{\beta}$ is a vector of length $p - 1$. This vector is used to fill in the corresponding column of W using $w_{12} = W_{11}\hat{\beta}$

3. *Continue until convergence.*

The algorithm described here is used in the software as well.

4.3 Shrinkage

Schäfer [5] describes the general shrinkage method as the weighted average between the original dataset, $X = \{X_1, ..., X_p\}$, and a corresponding constrained submodel of this set, say Y. A weighted average would be calculated as follows:

$$X_S = \bar{\lambda}_s X + (1 - \bar{\lambda}_s)Y \tag{5}$$

where $\bar{\lambda}_s$ is the shrinkage estimator. Schäfer suggests to construct an unbiased empirical covariance matrix Σ^S, where the s distinguishes this estimated covariance matrix from the other estimations. In order to determine this shrinkage estimator, Schäfer uses the following equalities. The unbiased empirical covariance equals

$$\widehat{\operatorname{Cov}}(x_i, x_j) = \sigma_{ij}^s = \frac{n}{n-1}\bar{w}_{ij}$$

and the unbiased empirical variance equals

$$\widehat{\operatorname{Var}}(x_i) = \sigma_{ii}^s = \frac{n}{n-1}\bar{w}_{ii}$$

With $w_{kij} = (x_{ki} - \bar{x}_i)(x_{kj} - \bar{x}_j)$ and $\bar{w}_{ij} = \frac{1}{n}\sum_{k=1}^{n} w_{kij}$, here the s in σ^s relates this value to Σ^s. With these values Schäfer suggests to calculate the shrinkage estimator λ_s as described below:

$$\lambda_s = \frac{\sum_{i=1}^{p} \widehat{\mathrm{Var}}(x_i) - \widehat{\mathrm{Cov}}(x_i, y_i) - \widehat{\mathrm{Bias}}(x_i)(x_i - y_i)}{\sum_{i=1}^{p}(x_i - y_i)^2} \tag{6}$$

In order to avoid 'overschrinkage' or a negative shrinkage Schäfer truncates this estimator, using $\bar{\lambda_s} = \max(0, \min(1, \lambda_s))$. This shrinkage estimator is then used to calculate the weighted average of the dataset.

4.4 Bootstrap

In its essence bootstrapping is a variance reduction method. The bootstrap approach replaces a subset of the dataset by samples from itself. If X is a dataset then the bootstrap algorithm 'creates' another dataset from the original set, consisting solely, but not necessarily completely of elements from this original set. This is done B times, where B is in the order of $10, 100$, or even 1000. Taking the average of the B samples gives the bootstrap estimate. Other authors have concluded that the bootstrap strategy is computationally very demanding when the dimension of the dataset are large (say p $>$ 1000) [5]. Instead Schäfer suggests the shrinkage method as a more efficient, but also more accurate alternative [5].

Nonetheless, since bootstrapping is a method that is encountered quite often, the procedure of bootstrapping will be analyzed here. In a previous paper Schäfer [4] refers to Breiman [1] for the procedure of bagging, an acronym for bootstrap aggregating. This method is implemented in the software as well.

Breiman defines a learning set, O, as $\{(X_i, Y_i), i = 1, \ldots, n\}$, with X_i representing a column of the original dataset and the entries Y_i are class labels or a numerical response. In this algorithm Breiman assumes to have a certain predictor, $\varphi(X, O)$. This predictor is used to make a bootstrapped sample for Y_i, based on X_i. The algorithm in question works as follows:

Algorithm 4.2 (Bootstrap algorithm)

1. *We start with, what Breiman calls a learning set, in the application in R an empty matrix with an equal number of columns is used. The number of rows in this matrix is dependent on the number of bootstrapped samples, B, the user requires. This learning set is then defined as follows:*

   $$O = \{(Y_i, X_i), i = 1, \ldots, n\}$$

   where the component Y_i is the empty set and X_i is the original dataset.

2. *Now a predictor is introduced:*

   $$\varphi(X, O)$$

   In the bootstrap approach that is used in this thesis this predictor is equipped with the identity function. That means that, with the input X_i, the empty entries of Y_i are replaced, in a random order, by entries of X_i.

3. *Step 2 is repeated until the empty entries of Y_i are all replaced by entries of X_i for all $i \in \{1, \ldots, n\}$. Note that the input matrix is of dimension $n \times p$ and that the bootstrapped sample is of dimension $B \times p$.*

This extended dataset improves the quality of the empirical covariance matrix relative to the the original dataset.

5 Application of Gaussian graphical modeling

For all of the described methods *R* packages are available. This availability and the straightforward application in *R* is why this software is used. In the next section the application of the methods and the visualization in *R* graphics will be laid out.

5.1 Application in R

The CRAN repository is equipped with many packages. Within this repository, due to its popularity, many packages related to Gaussian graphical modeling are available. This means that for the proposed methods more methods may be available, for this study the following selection has been made:

Method	**Package**
Moore-Penrose method (PINV)	"ppcor"
Lasso, inclusion of the ℓ_1-penalty-term	"glasso"
Bootstrap	"boot"
Shrinkage	"GeneNet
Graphical modeling	"igraph"

All of which are included within CRAN repository. The package "GeneNet" also contains an E.coli- and a gene-dataset. These sets contain the data of 102 E.coli bacteria with 8 observations per bacteria and 800 variables with 22 observations each respectively.

5.1.1 Moore-Penrose method

The package used for this method is supplied with several functions, however, only one of those is used in this study, namely:

Action	R entry
Installs packages	install.packages("ppcor")
Adds package to current pc's library	library(ppcor)

Once added to your own library the pseudoinverse of a matrix X can then be found using:

```
"pcor(X, 'pearson')$estimate"
```

The application of this method is very straightforward. The command "pcor(X)" generates a partial correlation matrix using the Moore-Penrose method. The "$estimate"-part then extracts the estimated out of the output that this command supplies.

5.1.2 Lasso (inclusion of the ℓ_1-penalty-term)

The application of the lasso (inclusion of the ℓ_1-penalty-term) relies on the following package.

Action	R entry
Installs packages	install.packages("glasso")
Adds package to current pc's library	library(glasso)

The package "glasso" provides an excellent function to estimate the covariance matrix (w) and it's estimated inverse (wi).

The following code can be used to plot the inverse of a covariance matrix of an original matrix X in a graph:

```
Sigma = cov(X)
X.lasso = glasso(Sigma,rho=0.1)
X.lasso = X.lasso$wi
X.lasso.cor = cor(X.lasso)
```

The first line creates a covariance matrix for the supplied data, this matrix is then used to create a lasso estimate of it's inverse using the "glasso" command. The last line calculates the correlation matrix from the estimated concentration matrix.

5.1.3 Bootstrap

At the time of writing this thesis the bootstrap approach is already outdated, Schäfer suggests to use the shrinkage approach instead. However, since it is still used, the method is included in this cross comparison.

Action	R entry (linux ubuntu)
Installs packages	install.packages("boot")
Adds package to current pc's library	library("boot")

Unfortunately the package which is suggested and used by Schäfer isn't available anymore. An alternative is found in the "boot" package. When using a matrix X the following code can be copied into the in R prompt in order to create a bootstrap sample:

```
identity <- function(X,d) {return(X[d])}

bootstrap <- function(X,R){
B = mat.or.vec( R ,ncol(X))
for (i in 1:ncol(X) ){
bi = boot(X[,i] , identity , R = R )
B[,i] = bi$t
}
return(B)
}
```

These lines of R-code first creates a function 'identity' which is our predictor for the bootstrap algorithm, the predictor as mentioned in the algorithm 4.2. Secondly a bootstrap function suitable for matrices is defined. The function 'bootstrap' requires an input of a matrix X and a number of bootstrap samples 'R'. This algorithm is based on the package "boot" mentioned in the table above.

5.1.4 Shrinkage

For the shrinkage approach the guidelines laid out by Schäfer are followed. Her approach to the shrinkage method is applied in the following package:

Action	R entry
Installs packages	install.packages("GeneNet")
Adds package to current pc's library	library(GeneNet)

The approach is both computationally as well as statistically very efficient, it is applicable to "small n, large p" data, and always returns a positive definite and well-conditioned covariance matrix according to the author of the package Schäfer [5].

```
X.shrink <- ggm.estimate.pcor(X)
```

This approach applies the method described in section 4.3,

5.2 Visualisation

The visualization of the graphs is done using the following package:

Action	R entry
installing package	install.packages"igraph"
Adds package to current pc's library	library(igraph)

Using this package and the following commands generates a graph for a matrix called "PartialCorrelation"

```
g <- graph.adjacency( PartialCorrelation > .1,
         mode="undirected", diag=FALSE )
plot(g,main="Test Graph")
```

Note the > .1 entry in the first line. This entry determines the threshold (0.10 in this case) for drawing an edge. This means that an edge is drawn if and only if the partial correlation is stronger than 0.10. This threshold can be adjusted according to the needs of the user. In the next sections this threshold is set as variable such that every method creates an equal number of edges.

6 Comparison test

The results contributing to the main goal of this thesis are going to be derived from the conclusion of this comparison test. This test is designed especially for this purpose. In the first subsection the methods are applied to a several randomly generated dataset of which the true edges are known. The next section will address the computational cost of each algorithm. The conclusions from these sections will form an overall conclusion.

Thehe second subsection each of the four methods are used to estimate a graph for the E.coli and arth800 datasets, containing 102 variables with 8 observations each and 800 variables with 22 observations per variable respectively. These computations are timed, based on the results of this timing a conclusion on the computing time is drawn. Both of these datasets are extracted from the GeneNet package.

In the last subsection some remarks about the applicability of each method are made. Together with the preceding two subsections, this subsection is going to lay the basis for the overall conclusion.

6.1 Accuracy

For the test on accuracy of each method a randomly generated dataset is used. This set is generates with the help of the following code:

```
library(huge)
X <- huge.generator(n,p, graph)
Data <- X$data
Sigma <- X$sigma
Omega <- X$omega
Theta <- X$theta
```

This code first adds the package 'huge' to the library. Once that is done then the data is generated, from this the original data 'Data', the covariance matrix 'Sigma', the precision matrix 'Omega' and the adjacency matrix of the true graph structure are extracted.

Applying the four methods to this data and then comparing the estimated edges to the true edges supplies us with enough data to compare. The following table shows the percentage of edges estimated correctly by each method for different dimensions of data. Recall that 'p' represents the number of variables and 'n' does so for the number of samples.

	Bootstrap	Lasso	Shrinkage	Moore-Penrose
$n = 4$, $p = 20$	$\mu = 24\%$ $s = 22.8\%$	$\mu = 94\%$, $s = 6\%$	$\mu = 29.7\%$, $s = 16\%$	$\mu = 50.7\%$, $s = 30\%$
$n = 10$ $p = 50$	$\mu = 22.8\%$ $s = 23.6\%$	$\mu = 92.4\%$ $s = 10.9\%$	$\mu = 16.3\%$ $s = 11.2\%$	$\mu = 45.2\%$ $s = 30.1\%$
$n = 50$ $p = 200$	$\mu = 31.4\%$ $s = 27.2\%$	$\mu = 99\%$ $s = 2\%$	$\mu = 8.5\%$ $s = 16\%$	$\mu = 48.4\%$ $s = 39.5\%$
$n = 100$ $p = 400$	$\mu = 30.2\%$ $s = 27.14\%$	$\mu = 98.6\%$ $s = 1.1\%$	$\mu = 0.4\%$ $s = 0.3\%$	$\mu = 7.2\%$ $s = 1.05\%$

In this test an average of 5 tests for each dataset is used. The table above shows that the Lasso algorithm proves to deliver the most accurate and also most consistent estimations. The estimations of the other three methods are not as consistent and certainly not as accurate. Moore-Penrose's approach seems to do quite well until the dataset reaches a much larger dimension. However, the high level of accuracy for the Lasso algorithm comes at a cost; the time it takes to compute the Lasso estimate is significantly higher than that of its peers. The next section elaborates on this.

6.2 Computing time

For this comparison test the two datasets of considerable different dimension are used. The following function is used to determine the time it takes for the graphs to compute:

```
system.time("function")
```

Using this function on each of our 'core functions', the function that is used to compute the partial correlation matrix from the data, will output the time used by these functions. Each function is applied to the E.coli- the arth800 and a random $n = 100$, $p = 400$ dataset. The following table shows how demanding each function is:

	Bootstrap	Lasso	Shrinkage	Moore-Penrose
E.coli	0.87s	2.12s	0.01s	0.01s
Arth800	8.52s	924.43s	0.44s	4.07s
Random	7.16s	3.78s	0.20s	0.56s

The table above shows that there is a large difference in computing time [1]. The table shows that the Moore-Penrose method and the shrinkage method are computationally very fast. For this test the bootstrap algorithm is instructed to generate a total of 1000 observation points, this seems to get more demanding when the number of variables gets larger. Despite its poor performance on accuracy in the previous section, the shrinkage method does prove to be really fast. The most significant result seems to be the computational time of the Lasso method on the largest dataset, the 'arth800' gene-dataset, the algorithm takes over 15 minutes to complete.

6.3 User friendliness

Since every method is applied in the same software the difference in the applicability of three out of the four methods is not significant enough to play a decisive role. There is a fourth method however in which a some difficulties are observed, the bootstrap method. The bootstrap method is, according to Schäfer [5], too demanding for large datasets, i.e. when the number of variables becomes larger. Because of this a package once distributed by her is no longer available. This means that for this cross comparison an older, less advanced, package is used. This package is not equipped to deal with matrices, but only with vectors. The code displayed in section 5.1.3 shows how the 'boot'-package is used to generate a bootstrap given its restrictions.

It can be concluded that the application of the bootstrap method is the least user friendly whereas the other three methods similar in their application.

[1] Ran on Linux Ubuntu equipped with an Intel Pentium Dual Core

7 Application to real life data

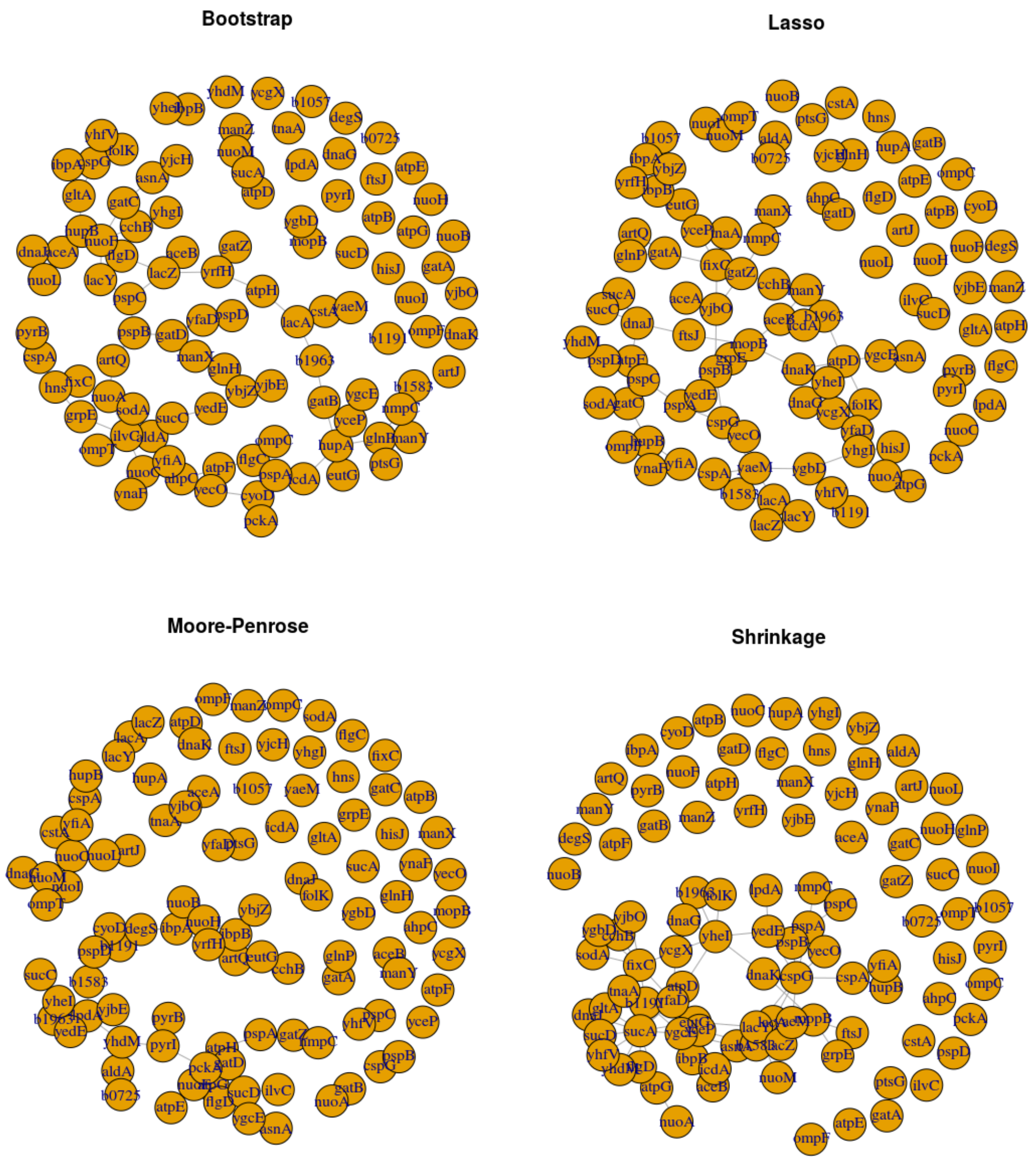

Figure 3: 102 nodes with 80 edges estimated by 4 different methods

In this section the E.coli dataset is used to apply each method. As mentioned, this data consists of 102 variables with 8 observations each. The reason that this dataset is chosen to be depicted here is that the relatively small amount of variables allows us to still have a bit of overview, using this same motivation there are only 80 edges depicted. Every node is fitted with the name of the bacteria, the igraph package redistributes the nodes such that the partially correlated nodes are near each other.

8 Conclusion

In the section 6 the accuracy, the computational time and the user friendliness of each method are addressed. In this section the results of these comparison tests are discussed per approach and afterwards an overall conclusion is drawn.

8.1 Conclusions per method

The following subsections address the results of each method based on the analysis in the first part of this thesis.

8.1.1 Conclusions on Lasso

The difference on performance on accuracy of the estimation could already be a decisive factor. On average the Lasso algorithm correctly estimates more than 90% of the edges, on some tests this algorithm even made a 100% correct estimation. However, this algorithm is by far the most computationally demanding. In section 6.2 it is shown that for the gene-dataset with 800 variables the algorithm takes more than 15 minutes to complete. This means that for larger datasets this method could be too demanding.

8.1.2 Conclusions on Bootstrap

When the number of variables, p, becomes large the time it takes to compute the bootstrap sample becomes significantly higher. The accuracy is not as impressive as the accuracy of the Lasso method, it can be concluded from the relatively high standard deviation shown in the table in 6.1 that the method is not very consistent. Another remark to be made about this method is that it is the least user friendly method.

8.1.3 Conclusions on Shrinkage

Although Schäfer suggests to use the shrinkage method instead of the bootstrap aggregating, the accuracy of the shrinkage method was very poorly. In section 6.1 it is shown that a bootstrapped sample will correctly estimate an average of around 30%, where the shrinkage method scored less than 1% on every test. This suggests that mistakes were made in the application of the package 'GeneNet' and its used function 'ggm.estimate.pcor('Data')'. Further research may be necessary in order to rule out any insecurities on this method.

8.1.4 Conclusions on Moore-Penrose

Despite its easy application and fast computing times the results on accuracy weren't that overwhelming. That is, for the first three datasets (up to $n = 50$ with $p = 200$) this method made an average of correctly estimated edges around 50%. However the test shows that the standard deviation is very large, even for the smallest dataset the results varied between 8% and 88%. This inaccurate performance could be a decisive factor.

8.2 Overall conclusion

Two of the analyzed methods perform on a low accuracy, the bootstrap- and the Moore-Penrose approach. Although the Moore-Penrose method is easy applicable the inconsistent performance on its accuracy is a large disadvantage. This large disadvantage is observed in the same way for the bootstrap approach, average accuracy with a high inconsistency. This is accompanied by a less user friendly package.

The lasso approach performs very well, even on the largest dataset that is used for this test. It has to be noted though that the computation time of this method is significantly higher than for its peers.

The shrinkage approach unfortunately performs poorly in this test, despite the fact that it is a recommended approach by Juliane Schäfer [5]. This suggests that further research is needed in order to rule out any mistakes made in this cross comparison.

In summary; from the results of this cross comparison it can be concluded that, without any accurate results on the shrinkage approach, the lasso approach supplies the user with the most accurate results. In 8 of the 20 tests this approach correctly estimates 100% of the edges. Further endevours could be pointed towards alternative methods such as the 'G-Wishart prior-method'. Comparing alternative methods to the lasso- and the, inconclusive shrinkage approach, could help to provide a full-scale manual. A manual which could provide researchers with well-founded information regarding a choice of method.

References

[1] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, Aug 1996.

[2] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008.

[3] R. Penrose. A generalized inverse for matrices. *Mathematical Proceedings of the Cambridge Philosophical Society*, 51(3):406–413, 1955.

[4] Juliane Schäfer and Korbinian Strimmer. An empirical bayes approach to inferring large-scale gene association networks. *Bioinformatics*, 21(6):754–764, 2004.

[5] Juliane Schäfer and Korbinian Strimmer. A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics. *Statistical applications in genetics and molecular biology*, 4(1), 2005.

[6] Patrizia F Stifanelli, Teresa M Creanza, Roberto Anglani, Vania C Liuzzi, Sayan Mukherjee, and Nicola Ancona. A comparative study of gaussian graphical model approaches for genomic data. *arXiv preprint arXiv:1107.0261*, 2011.

[7] Shuheng Zhou, Sara van de Geer, and Peter Bühlmann. Adaptive lasso for high dimensional regression and gaussian graphical modeling. *arXiv preprint arXiv:0903.2515*, 2009.

A Code generating graphs and comparison table

The code provided in this appendix generates a multivariate normal distribution, applies each method to this generated dataset and then outputs the time it took to compute these estimations and the percentage of correctly estimated edges.

The following code installs the necessary packages to the library of the pc. Note that this is only needed if these packages are not installed previously.

```
install.packages("ppcor")
install.packages("igraph")
install.packages("boot")
install.packages("glasso")
install.packages("GeneNet")
install.packages("huge")
```

The following code adds the necessary packages to the library of the pc, this action is needed every time *R* is restarted again.

```
library(ppcor)
library(igraph)
library(boot)
library(glasso)
library(GeneNet)
library(huge)
```

This small line specifies the number of plots that fit into *R* Graphics. With the code supplied below the graphs are plotted in 3 rows and 2 columns

```
par(mfrow=c(3,2))
```

The first of the next four lines is used to determine the type of multivariate normally distributed dataset. The next three lines are used to determine the number of observations, the number of variables and the number of times that the sample is bootstrapped respectively.
Note: the values given to these variables may be changed according to ones needs

```
graph <- "random"
n <- 10
p <- 40
R <- 1000
```

In the next lines the multivariate normally distributed dataset is generates according to the dimensions specified in the lines above. The next four lines extract 4 types of data that are generates by the package 'huge'. Namely:

1. X$data extracts the data from X, from now on called 'Data'
2. X$sigma supplies the user with the true covariance matrix
3. X$omega assigns the concentration/precision matrix to the name 'Omega'
4. X$theta extracts the adjacency matrix of true graph structure. This matrix is used in the comparison test

```
X <- huge.generator(n,p, graph)

Data <- X$data
Sigma <- X$sigma
Omega <- X$omega
Theta <- X$theta
```

The adjacency matrix is symmetric, presenting a value '1' in the $\{i,j\}^{th}$ and the $\{j,i\}^{th}$ position when the nodes X_i and X_j are partially correlated. The matrix 'Theta" is first reduced to an upper diagonal matrix such that it only presents one value '1' for every edge. Afterwards the total amount of edges is counted, this count is used in the application of the four methods subject to the comparison

```
Theta.reduced <- Theta
Theta.reduced[lower.tri(Theta.reduced)] <- 0
edges = length(which(Theta.reduced == 1) )
```

The first line in the following piece of code creates a string of names of the following form: Var_i. The lines after that assign these names to the columns and rows of the used matrices.

```
names <- paste("Var_", 1:p)

colnames(Data) <- names

colnames(Sigma) <- names
rownames(Sigma) <- names

colnames(Omega) <- names
rownames(Omega) <- names

colnames(Theta) <- names
rownames(Theta) <- names
```

The next two lines plot the graph belonging to the ground truth, i.e. the graph depicting the true partial correlations.

```
t <- graph.adjacency(Theta == 1,mode = "undirected" ,diag =FALSE)
plot(t, main = "Ground Truth")
```

The next section of code starts with timing the function that estimates the partial correlation matrix according to the Moore-Penrose method. After that the names of the variables are assigned to the columns and rows. The next two lines create an upper diagonal matrix which only shows the partial correlations in the upper diagonal part of the matrix. Then, to the name 'm.highest' the strongest partially correlated entries are assigned in descending order. At last the graph is plotted with exactly the same number of edges as for the true graph.
Note: this approach is used for all four methods

```
time.ginv <- system.time(
X.ginv <- pcor(Data)$estimate
)

colnames(X.ginv) <- names
rownames(X.ginv) <- names

X.ginv.reduced <-  X.ginv -diag(ncol(X.ginv))
X.ginv.reduced[lower.tri(X.ginv)] <- 0

m.highest <- order(abs(X.ginv.reduced), na.last=TRUE,
          decreasing=TRUE)[1:edges]

m <- graph.adjacency(abs(X.ginv) >= abs(X.ginv[m.highest[edges]]),
          mode ="undirected",diag=FALSE)

plot(m,main="Moore-Penrose")
```

The outline of this next part of code is the same as for the Moore-Penrose method. Though first 1 line is used to define a function which is used by the 'boot' function. The next 8 lines create the function 'bootstrap' based on the function 'boot'. Since the function 'boot' is not equipped to handle matrices this function containing a for loop is introduced.

The rest of this part of code relies on the same strategy as the Moore-Penrose method.

```
identity <- function(X,d)  {return(X[d])}

bootstrap <- function(X,R){
B = mat.or.vec( R  ,ncol(X))
for  (i  in  1:ncol(X)  ){
bi  = boot(X[,i]  ,  identity  , R = R  )
B[,i] = bi$t[,1]
}
return(B)
}

time.boot <- system.time(
X.boot <- bootstrap(Data,R)
)

X.boot.pcor <- pcor(X.boot)
X.boot.pcor <- X.boot.pcor$estimate
```

```
colnames(X.boot.pcor) <- names
rownames(X.boot.pcor) <- names


X.boot.pcor.reduced <-  X.boot.pcor -diag(ncol(X.boot.pcor))
X.boot.pcor.reduced[lower.tri(X.ginv)] <- 0

b.highest <- order(abs(X.boot.pcor.reduced), na.last=TRUE,
          decreasing=TRUE)[1:edges]


b <- graph.adjacency(abs(X.boot.pcor) >= abs(X.boot.pcor[b.highest[edges]]),
          mode ="undirected",diag=FALSE)

plot(b,main="Bootstrap")
```

This part of code also relies on the same strategy as the Moore-Penrose application.

```
time.lasso <- system.time(
X.lasso <- glasso(Sigma,rho=0.1)
)

X.lasso = X.lasso$wi
X.lasso.pcor = pcor(X.lasso)
X.lasso.pcor <- X.lasso.pcor$estimate

colnames(X.lasso.pcor) <- names
rownames(X.lasso.pcor) <- names

X.lasso.pcor.reduced <- X.lasso.pcor
diag(X.lasso.pcor.reduced) <- 0
X.lasso.pcor.reduced[lower.tri(X.lasso.pcor)] <- 0

l.highest <- order(abs(X.lasso.pcor.reduced), na.last=TRUE,
          decreasing=TRUE)[1:edges]

l <- graph.adjacency( abs(X.lasso.pcor) >= abs(X.lasso.pcor[l.highest[edges]]),
          mode ="undirected", diag=FALSE)

plot(l,main="Lasso")
```

Once again the same strategy is used.

```
time.shrink <- system.time(
X.shrink <- ggm.estimate.pcor(Data)
)



X.shrink.reduced <-  X.shrink
diag(X.shrink.reduced) <- 0
```

```
X.shrink.reduced[lower.tri(X.shrink)] <- 0

colnames(X.shrink) <- names
rownames(X.shrink) <- names

s.highest <- order(abs(X.shrink.reduced), na.last=TRUE,
          decreasing=TRUE)[1:edges]

s<- graph.adjacency(abs(X.shrink) >=  abs(X.shrink[s.highest[edges]]),
          mode="undirected",diag=FALSE)

plot(s,main="Shrinkage")
```

Below the previously generates names are assigned to our reduced matrix 'Theta'. The value 'kt' then represents the number of edges.

```
colnames(Theta.reduced) <- names
rownames(Theta.reduced) <- names
```

This section of code creates an upper triangular binary matrix. This matrix contains a value '1' in entry $\{i, j\}$ with $i > j$ for every edge that is estimated by the lasso procedure.
Note: this same approach is used for each of the four methods

```
X.lasso.reduced = X.lasso.pcor
diag(X.lasso.reduced) <- 0
X.lasso.reduced[lower.tri(X.lasso.reduced)] <- 0
X.lasso.reduced[X.lasso.reduced < 0] <-0
X.lasso.reduced[X.lasso.reduced < X.lasso.reduced[l.highest[edges]]] <-0
X.lasso.reduced[X.lasso.reduced > X.lasso.reduced[l.highest[edges]]] <-1

kl <- length(which(X.lasso.reduced > 0))
for (i in 1:kl){
X.lasso.reduced[which(X.lasso.reduced > 0)[i]] <- 1
}
```

Below the same approach is used to create an upper triangular binary matrix.

```
X.shrink.reduced = X.shrink
diag(X.shrink.reduced) <- 0
X.shrink.reduced[lower.tri(X.lasso.reduced)] <- 0
X.shrink.reduced[X.shrink.reduced < 0] <-0
X.shrink.reduced[X.shrink.reduced <
          X.shrink.reduced[s.highest[edges]]] <-0
X.shrink.reduced[X.shrink.reduced >
          X.shrink.reduced[s.highest[edges]]] <-1


ks <- length(which(X.shrink.reduced > 0))
for (i in 1:ks){
X.shrink.reduced[which(X.shrink.reduced > 0)[i]] <- 1
}
```

Once again, the same approach is used

```
X.ginv.reduced = X.ginv
diag(X.ginv.reduced) <- 0
X.ginv.reduced[lower.tri(X.ginv.reduced)] <- 0
X.ginv.reduced[X.ginv.reduced < 0] <-0
X.ginv.reduced[X.ginv.reduced <
          X.ginv.reduced[m.highest[edges]]] <-0
X.ginv.reduced[X.ginv.reduced >
          X.ginv.reduced[m.highest[edges]]] <-1

kg <- length(which(X.ginv.reduced > 0))
for (i in 1:kg){
X.ginv.reduced[which(X.ginv.reduced > 0)[i]] <- 1
}
```

Once more, an application of this approach

```
X.boot.reduced = X.boot.pcor
diag(X.boot.reduced) <- 0
X.boot.reduced[lower.tri(X.boot.reduced)] <- 0
X.boot.reduced[X.boot.reduced < 0] <-0
X.boot.reduced[X.boot.reduced < X.boot.reduced[b.highest[edges]]] <-0
X.boot.reduced[X.boot.reduced > X.boot.reduced[b.highest[edges]]] <-1

kb <- length(which(X.boot.reduced > 0))
for (i in 1:kb){
X.boot.reduced[which(X.boot.reduced > 0)[i]] <- 1
}
```

The following code creates an empty matrix which will eventually display the true edges and the correctly estimated edges for each method.

```
Agreement.matrix <- matrix(numeric( (kt+1)*7 ),ncol=7)

colnames(Agreement.matrix) <- list("Node 1", "Node 2","Ground truth",
          "Bootstrap","Lasso","Shrinkage","Moore.Penrose")
```

The following 'for loop' enters the variables which are truly partially correlated into the 'Agreement.matrix'

```
for (i in 1:edges){
t <- which(Theta.reduced == 1)
k <- arrayInd(t[i],dim(Theta.reduced))

Agreement.matrix[i,1] <- rownames(Theta.reduced)[k[,1]]
Agreement.matrix[i,2] <- colnames(Theta.reduced)[k[,2]]
}
```

The 'for loop' displayed here enters a value '1' into the 'Agreement.matrix' for every edge that is correctly estimated by each method.

```
for (i in 1:kt){
t <- which(Theta.reduced == 1)
k <- arrayInd(t[i],dim(Theta.reduced))
Agreement.matrix[i,3] <- Theta.reduced[k]
Agreement.matrix[i,4] <- X.boot.reduced[k]
Agreement.matrix[i,5] <- X.lasso.reduced[k]
Agreement.matrix[i,6] <- X.shrink.reduced[k]
Agreement.matrix[i,7] <- X.ginv.reduced[k]
}
```

The next lines calculate and enter the percentage of correctly estimated edges for each method. The resulting matrix will give a nice overview of correctly estimated edges.

```
Agreement.matrix[(kt+1),1] <- "Correctly"

Agreement.matrix[(kt+1),2] <- "estimated:"

Agreement.matrix[(kt+1),3] <-
        (sum(as.numeric(Agreement.matrix[,3]))/kt)*100

Agreement.matrix[(kt+1),4] <-
        (sum(as.numeric(Agreement.matrix[,4]))/kt)*100

Agreement.matrix[(kt+1),5] <-
        (sum(as.numeric(Agreement.matrix[,5]))/kt)*100

Agreement.matrix[(kt+1),6] <-
        (sum(as.numeric(Agreement.matrix[,6]))/kt)*100

Agreement.matrix[(kt+1),7] <-
        (sum(as.numeric(Agreement.matrix[,7]))/kt)*100
```

This last section outputs the matrix containing the percentage of correctly estimated edges and the time it took every method to compute this estimate.

```
Agreement.matrix

time.boot
time.lasso
time.shrink
time.ginv
```