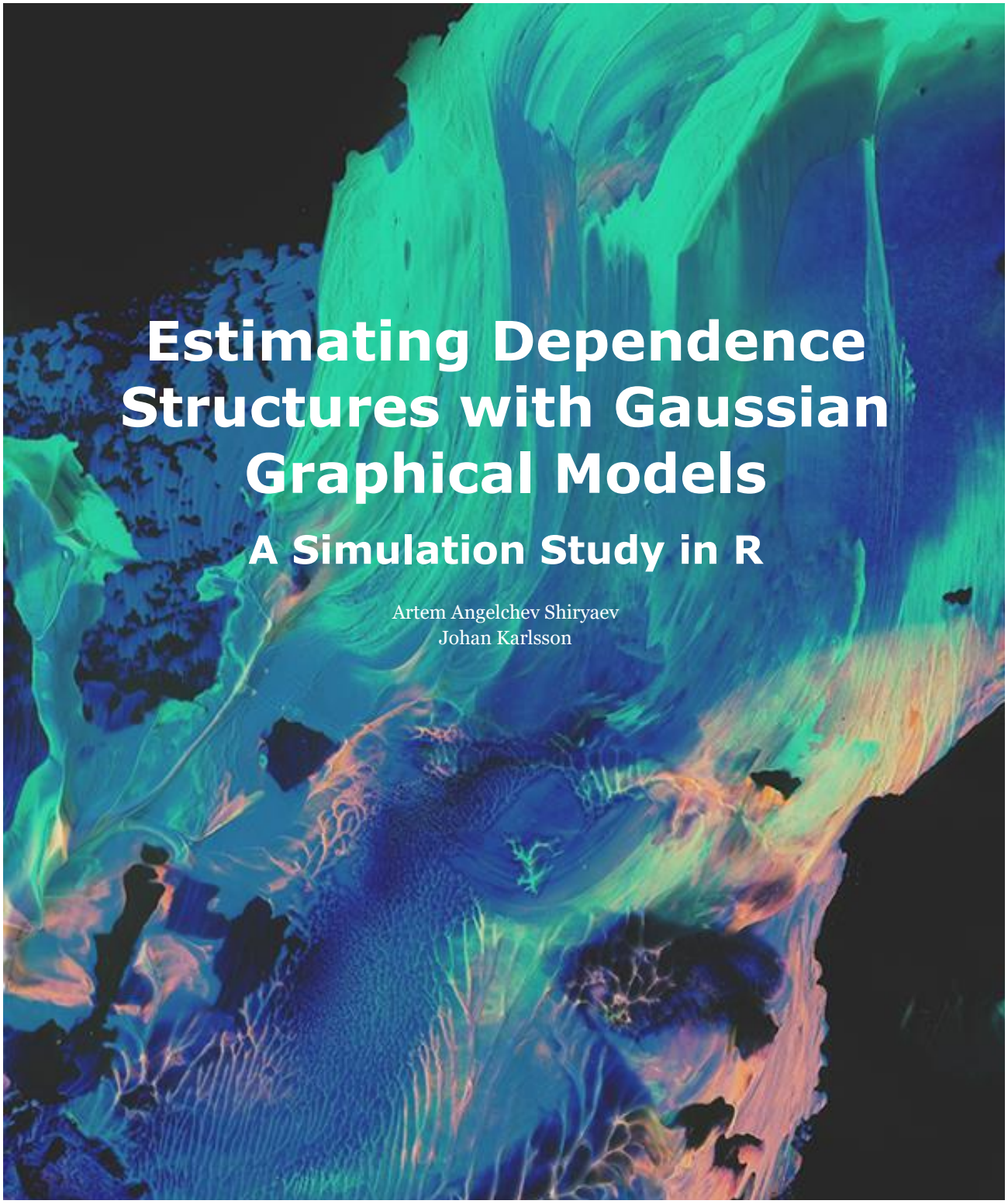
UMEÅ UNIVERSITY



Estimating Dependence Structures with Gaussian Graphical Models

A Simulation Study in R

Artem Angelchev Shiryaev
Johan Karlsson

Bachelor Thesis, 15 ECTS
Bachelor of Science in Statistics, 180 ECTS
Spring term 2021

Estimating Dependence Structures with Gaussian Graphical Models

A Simulation Study in R

Artem Angelchev Shiryaev
Johan Karlsson

Bachelor of Science in Statistics
Department of Statistics
Umeå University
Spring 2021

Abstract

Graphical models are powerful tools when estimating complex dependence structures among large sets of data. This thesis restricts the scope to undirected Gaussian graphical models. An initial predefined sparse precision matrix was specified to generate multivariate normally distributed data. Utilizing the generated data, a simulation study was conducted reviewing accuracy, sensitivity and specificity of the estimated precision matrix. The graphical LASSO was applied using four different packages available in **R** with seven selection criteria's for estimating the tuning parameter λ.

The findings are mostly in line with previous research. The graphical LASSO is generally faster and feasible in high dimensions, in contrast to stepwise model selection. A portion of the selection methods for estimating the optimal tuning parameter obtained the true network structure. The results provide an estimate of how well each model obtains the true, predefined dependence structure as featured in our simulation. As the simulated data used in this thesis is merely an approximation of real-world data, one should not take the results as the only aspect of consideration when choosing a model.

keywords: Simulation study, Graphical models, undirected Gaussian graphical model, Partial correlation, Precision matrix.

Beroendestruktur Skattning med Gaussianska Grafiska Modeller

En Simuleringsstudie i R

Sammanfattning

Grafiska modeller är effektiva verktyg vid skattning av komplexa beroendestrukturer för stora datamaterial. Uppsatsen avgränsar undersökningen till oriktade Gaussianska grafiska modellen. En förutbestämd så kallad gles matris skapas, som används för generering av multivariat normalfördelat data. På datamaterialet tillämpas en grafisk LASSO från olika paket i **R** , med olika selektionskriterier för skattning av hyperparametern λ. Från predikterade graferna beräknas specificitet, sensitivitet och träffsäkerheten för de olika selektionskriterierna.

Resultaten är mestadels i linje med tidigare forskning. Den grafiska LASSO är generellt sett snabbare och kan tillämpas i högre dimensioner, till skillnad från stegvis selektion. Valet av hyperparameter har stor betydelse för den grafiska LASSO, då olika modellutföranden generar olika grafiska modeller. Resultatet är en estimation av hur väl varje modell återhämtar en bestämd beroendestruktur. Läsaren bör ha i åtanke att datat är simulerat och enbart en approximation av ett verkligt data, således bör fler aspekter tas i beaktning vid val av modell.

nyckelord: Simuleringsstudie, Grafiska modeller, Oriktad Gaussianska grafiska modellen, Partiell korrelation, Precision matris.

Acknowledgements

We would like to give our sincerest *thank you*, to our supervisor Xavier de Luna for his valuable expertise, insight, positive attitude and constructive feedback throughout our thesis.

Populärvetenskapliga sammanfattningen

I dagens samhälle samlas datamaterial in oavbrutet, dygnet runt, på alla olika sorters områden. Som akademiker vill man gärna undersöka samt klargöra hur variabler påverkar varandra. Ett klassiskt nationalekonomiskt exempel är att generellt sett ju högre utbildning en individ har, desto högre inkomst. Svårigheterna som uppkommer i och med att tillgången av datamaterial exploderat det senaste decenniet är att korrelationen och samhörigheten av det stora antalet variabler inte är så enkelt som i exemplet ovan. Hur ska man då bedöma och dra enkla slutsatser när insamlat datamaterial har en oerhört komplex beroendestruktur?

I denna uppsats undersöks det hur man kan skatta en *undirected Gaussian graphical model* och således hur man kan uppskatta och visualisera en ungefärlig bild av verkligheten, givet vissa tekniska kriterier för variablerna inom datamaterialet. Uppsatsen använder sig av simuleringsmetoder för att skapa datauppsättningar med specifika kriterier för att sedan kontrolleras utifrån hur väl de berörda modellerna kan återskapa det sanna uppsättningar av samhörighet.

Resultatet kan tillämpas inom områden såsom finans, biologi och psykologi för att bättre kunna förstå hur variabler hänger ihop, samt vilka variabler som egentligen är mer relevanta och vilka som egentligen skulle kunna ignoreras. Resultatet påverkar således möjligheten att sålla bort irrelevanta samband direkt, som därmed skapar en snäv och precis problemformulering att utgå ifrån.

Arbetet i uppsatsen är viktigt för att bekräfta de nyskapande metoder som nyligen uppkommit på grund av tillgången till datamaterial. Bör akademiker använda sig just av denna modell? Uppsatsen påvisar att, givet att vissa tekniska kriterier är uppfyllda, är denna metod ett utmärkt val för en deskriptiv och ordentlig analys av variablernas beroendestruktur.

Contents

Chapter 1

Introduction

Exploratory data analysis is an important aspect of statistics and data science. By visualizing the data and applying robust statistical procedures, the analyst can obtain a further understanding of the data by reviewing patterns. These patterns are essential for generating hypotheses for subsequent testing, while also providing the analyst with a broad intuition of a given set of data, [3]. Typical methods for exploratory data analysis include estimation of means and standard deviations, visualizing the distribution of the data and estimating the correlation between features. Correlation is the standardized measurement of covariance, and is an estimate of the degree to which features are linearly dependent on each other, [21]. The key aspect of this estimate is that it only measures the so-called pairwise correlation, meaning that the estimate does not take other features into account, [28]. Often, especially when working with a large set of data, a considerable subset of features may be correlated, providing the analyst with no further information apart from the marginal dependence between a pair of features. Depending on the task at hand, this might be satisfactory, but a natural extension would be to examine the conditional dependence; the dependence between a pair of features given all other features.

In recent years, the use of graphical models in statistical analysis has gained popularity in a variety of fields by visualizing the joint distribution of a set of features. As a way of providing the reader with a non-technical, visual introduction into this subject, the starting point of this thesis is an example using the Auto dataset, featured in the ISLR package, [27]. Using the base packages included in **R** , the Pearson correlation coefficients are calculated using the five continuous variables *Miles per Gallon, Displacement, Horsepower, Acceleration* and *Weight*.

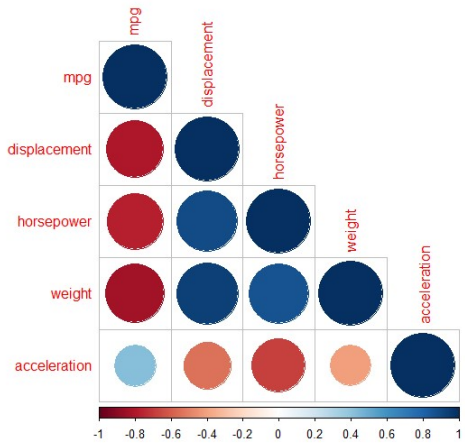


Figure 1.1: Correlation plot using the corrplot package in **R**

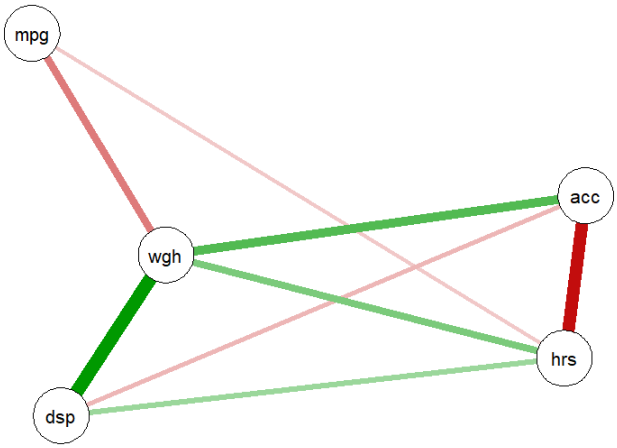


Figure 1.2: Gaussian graphical model using the qgraph package in **R**

By inspection of Figure 1.1, it is evident that a large share of the features are highly correlated. Interpretation is limited to stating that most features seem to have a positive or negative linear association between them. The plot does not provide any information regarding the underlying dependence structure, since pairwise correlation does not distinguish between direct and indirect associations. For further knowledge, the analyst must resort to other methods with varying amounts of

complexity, such as a linear regression model.

The graphical model in Figure 1.2, is an example of a so-called Gaussian graphical model. This model encodes a special property. Given that a few assumptions of the data are fulfilled, the graph represents the conditional dependence of the set of features. This representation is highly useful in many situations, as it provides a more intricate understanding of the data, while still being exploratory in nature and easy to interpret. Within graphical modelling, the set of vertices(also called nodes) correspond to a set of features, and the set of edges represent the conditional dependence of the aforementioned features, [23]. Returning to the Auto data, a way of interpreting Figure 1.2 is; *given* the number of horsepower and weight of a car, Miles per Gallon is conditionally independent of acceleration and displacement, since there exists no edge linking these features directly to one another.

It is necessary to mention that the thesis is limited to the undirected Gaussian graphical model. An undirected model implies there exists no arrows between features when graphically displayed. As such, it relies on fewer stringent assumptions compared to Directed Graphical Models, (DGM), [15]. For such models, as well as log-linear models used for discrete data, the reader is referred to [25].

1.1 Purpose and Research Questions

Producing statistically robust, reliable results is always preferred to inferior methods. Understanding how to estimate and manage highly correlated as well as partially correlated dependence structures in both low and high dimensions are vital for a better understanding of the data. Our proposed method for comparing different estimation techniques starts with randomly generating a multivariate data from the normal distribution with a predefined dependency structure. Using different packages in **R** to subsequentially estimate Gaussian graphical models, each estimate is compared to the true dependence structure.

Visualizing features of the data as nodes in a graph in not a new phenomenon, especially within psychometrics and econometrics, [12]. However, the techniques developed and integrated in the statistical programming language **R** are relatively new [38, 19], enabling the model to be used in many different areas of research. In essence, the overarching goal of this thesis is to empirically evaluate existing packages using simulation, with the addition of a general description of each package. The purpose is not to fully utilize and evaluate all encompassed functions in a given package, but

rather to give an overview and compare the results of different packages for readers not fully aware of the available methods regarding graphical models and their use.

This simulation study is intended to provide a fair empirical comparison on how well each estimation method can extract a known dependency structure. The primary reason of performing a simulation study rather than using real world data, is due to the easily verifiable normality assumption and also for verification of the given aforementioned structure. Using real world data is without doubt beneficial, as simulated data might represent an unrealistic scenario not encountered in the real world. However, in order to ensure the fairest possible comparison, guaranteeing a predefined dependency structure is of vital importance as a point of reference for all methods.

Subsequently, the research questions are:

- Describe and explore existing **R** packages for estimation of undirected Gaussian graphical models.
- How well does the various methods in **R** estimate the dependence structure of the simulated data?
- What are the main differences among the estimated dependence structures of the included packages?

This text aims at providing the reader with an understanding of the undirected Gaussian graphical model, and the different ways of how it can be implemented in **R** . It is organized as follows: Chapter 2, *Fields of Application* provides the reader with an understanding of the model through the different ways it has been used in applied research. Chapter 3 consists of the theoretical model and methodology investigated and is intended to be later applied within the packages. Chapter 4 provides a general description of the packages, providing an overview of the different functions featured in each package. Chapter 5 describes the aggregated simulation process, bringing together the covered theoretical method and application within the package. Furthermore, Chapter 6 presents the results of our simulation study. The discussion and concluding remarks are collected in Chapter 7.

Chapter 2

Fields of Application

The goal of this section is to provide the reader with a further understanding of the Gaussian graphical model through the context in which they are used. Featured are examples within psychology, biology, finance and nutritional science.

2.1 Psychology

Within psychology, a prominent theory of describing human behaviour is the so called Five-Factor Model, or the Big Five, [12, 4]. This model has been used as a tool for describing differences in the way humans interact with the world, where the actions taken by a certain individual relies primarily on the degree to which each underlying factor make up an individuals personality.

In a 2012 study, the authors, [8], opposes this as a theory for describing human behaviour, identifying that behaviour patterns are much more dynamic and changing depending on environment. Instead, they argue for a network-based approach when studying human behaviour. The network is constructed using data obtained from of a questionnaire, and the connections constructed between them are made through correlations. Thus, instead of looking at certain components of an individual and relating them to an underlying personality trait, they construct a network of the components. Using an **R** package developed by themselves, [16], the network can be represented in a graph displaying the structure of how certain components are linked to each other. Using this network approach creates a useful tool for visualizing dependence structures; how certain personality components are related to each other.

While standard correlation measures can be used as a basis for the network, they argue that partial correlations might be more meaningful. This is because standard

correlation measures do not take potential indirect effects into consideration. For instance, in a simple case with features A, B and C, where A is correlated with B, and B is correlated with C. A and C will most likely be correlated through the common dependence with B, even if there exists no direct association between these two features. While acknowledging the fact that the network based approach in psychometrics is still under development, the potential benefit of a network based approach is that of providing researchers with a more dynamic tool when examining human behaviour, [8].

2.2 Biology

In the field of biology, Gaussian graphical models are used to study the complex independence structure of gene expressions obtained from microarrays, [37]. A microarray is a glass slide in which segments of DNA have been placed, [24]. This type of data has become an important source of information in modern genomics, providing researchers with a much deeper understanding of the processes occurring on a cellular level.

Similar to within psychometrics, the task at hand lies in determining which features are dependent on each other. Like partial correlations, standard pairwise correlations can also be visualized as a graph, known as a relevance network. These do not however represent conditional dependence, and are less sparse, [7]. For these reasons, a Gaussian graphical model is favourable in a high-dimensional setting. But considering the fact that data generated from microarrays are in general quite noisy [22], and it is not uncommon that the number of features greatly exceed the number of observations, estimating this type of model is challenging. Current literature therefore focuses on overcoming these challenges by using different methods to estimate a Gaussian graphical model when using microarray data, see for instance, [49]. For further details regarding the problem of features outnumbering observations, $n < p$, see Chapter 3.

2.3 Finance

In the field of finance, Gaussian graphical models have been implemented to model the conditional independence structure between both domestic and international stock market indexes. In a 2005 study, the authors constructed a network using data

from the US stock market in order to organize stocks into clusters based on changes in their respective prices, [5]. Consider the case of diversifying a stock-portfolio; in order to mitigate the risk of investing, placing money in different stocks is a way of lowering the risk of losing money. However, this is only true given that the movement in prices of the included stocks are independent of each other. By analyzing the data in this manner, independent sets of stocks can be identified and used to mitigate the aforementioned risk when investing.

A study from 2008 considers the global stock market index, and uses a graphical model based on partial correlations to study the independence structure of financial markets, [1]. The standard correlation estimates between the financial markets is consistent with the literature; the domestic markets that constitute the global market are highly dependent on each other, meaning that price changes in a certain market effects other markets worldwide. Partial correlations however highlight different patterns in the way markets interact with each other. While all markets are correlated, not all are partially correlated, providing insight of which markets that are conditionally dependent on each other. As an example from the analysis, the Canadian market is independent of all other markets given the US market, and the US market is independent of the European market given the German market. The partial correlation estimates can so forth be used as a map in order to distinguish which markets directly interact with each other.

2.4 Nutritional Science

In the field of nutritional science, Principal Component Analysis (PCA), and cluster analysis is a way of exploring variation in dietary intake, [26]. In a study from 2006, the authors describe how undirected Gaussian graphical models can work as a viable alternative to these methods for pattern recognition and to create a more insightful explanatory analysis of nutritional intake. Although both PCA and other cluster methods do identify dietary patterns adequately, these methods are based on standard pairwise correlations. The primary advantage of using an undirected Gaussian graphical model is that it enables researchers to draw conclusions on how various foods are directly related to each other through conditional dependence. Furthermore, interpretation of the results of competing methods can be challenging due to underlying dependence structure. According to the article, red meat is highly correlated with poultry, processed meat, sauce, and potatoes. Based on this information, it would be misleading to infer any other relationship. For instance, an increase in consumption of red meat will increase the consumption of poultry or vice versa,

since standard correlation does not differentiate between indirect or direct dependence, [26]. In addition, the authors reiterate that the undirected Gaussian graphical model presents an intuitive dependence structure and is easily interpreted due to the conditional independence assumption when evaluating edges between nodes. In the authors case, the results showed that the intake of processed meat and poultry was conditionally dependent on red meat intake.

One of the main deficiencies highlighted by the authors of undirected Gaussian graphical models is the normality assumption, which was certainly not the case for some foods. In fact, many features had a skewed distribution, [26]. Therefore, a logarithmic transformation was applied to improve normality. To further evaluate the robustness of the results, the authors reconfirmed their finding with the semiparametric Gaussian copula graphical model, (SGCGM). SGCGM does not inherently require normality of the data and the estimated SGCGM had a strong resemblance of the undirected Gaussian graphical model.

2.5 Chapter Summery

To summarize this chapter, the undirected Gaussian graphical model can be used in many fields of research as a powerful tool for exploratory data analysis. Additional mentions of applications not covered here include signal processing, structural time-series, image analysis and spatial statistics, [43]. In comparison to standard correlation measures and even data-reduction methods such as PCA, the undirected Gaussian graphical model suggests a clear advantage. The identification of direct associations between features may be the most obvious one, since it enables a more meaningful analysis in situations in which the standard correlation matrix might be impractical due to high-dimensionality. The visual aspect should however not be neglected, as one could argue that the visual representation of the undirected Gaussian graphical model is more straightforward and intuitive to understand for a non-statistician in comparison to equivalent graphical output of PCA.

Chapter 3

Methods

3.1 Preliminaries of Graph Theory

A common way of mathematically describing a graph is $G = (V, E)$ where, V is a set of vertices (also called nodes) and E is a set of edges (also called links). An edge is associated with two distinct nodes, [23]. Combining probability theory and graph theory, one can utilize the described framework for statistical analysis. Consider the multivariate normal data X with features $(X_1, X_2, ..., X_p)$. Think of each node as a feature, and each edge as the dependence between any two features i and j given all other features.

Given a positive definite $p \times p$ covariance matrix Σ, all information needed to graphically visualize the dependence structure is stored in the *precision matrix*, [15], which is defined by the following formula, [25];

$$\Sigma^{-1} = \Omega. \tag{3.1}$$

For each element u in Ω, obtain the corresponding partial correlation coefficients by;

$$\rho_{ij} = \frac{\text{-}\, u_{ij}}{\sqrt{u_{ii} u_{jj}}}. \tag{3.2}$$

Given that the set of features is multivariate normally distributed, these partial correlation coefficients encode the conditional dependence of the joint distribution of the data, [23, 47]. For all coefficients of the precision matrix u:

$$u_{ij} = 0 \qquad if\ and\ only\ if \qquad X_i \amalg X_j \big| X_{-i,-j}, \tag{3.3}$$

meaning that X_i and X_j are independent given all other features, [10].

This can formally be stated as;

$$X_i \amalg X_j \big| X_{-i,-j} \qquad means \qquad p(x_i, x_j | x_{-i,-j}) = p(x_i|x_{-i})p(x_j|x_{-j}). \tag{3.4}$$

Consider the two Figures 3.1 and 3.2. By observing the coefficients of a given precision matrix, one can visualize the dependence structure by simply observing the non-zero elements. In the 4×4 precision matrix, the elements corresponding to the conditional dependence between X_1 and X_3 are 0, thus yielding no edge in the corresponding graph. Thus, in an undirected Gaussian graphical model, the set of edges E in a graph G corresponds directly to each element of the precision matrix.

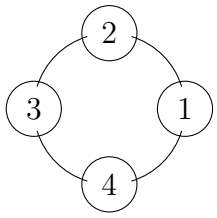


Figure 3.1:
Graph with 4 nodes and 4 edges

$$\begin{array}{c} \\ Node_1 \\ Node_2 \\ Node_3 \\ Node_4 \end{array} \begin{array}{c} \begin{array}{cccc} Node_1 & Node_2 & Node_3 & Node_4 \end{array} \\ \begin{pmatrix} u_{11} & u_{12} & 0 & u_{14} \\ u_{12} & u_{22} & u_{23} & 0 \\ 0 & u_{23} & u_{33} & u_{34} \\ u_{14} & 0 & u_{34} & u_{44} \end{pmatrix} \end{array}$$

Figure 3.2: Precision Matrix of Figure 3.1

3.2 Model Selection

A key aspect of graphical modelling is model selection. It corresponds to the methods used for adding or deleting edges in the graphical model, in order to obtain a estimate of the true network structure. Given the nature of graphical models, sparsity is an essential aspect. In a high-dimensional setting, a highly dense network structure would not be very informative from a visual standpoint. At the same time, the estimated model should not exclude edges that correspond to features which are conditionally dependent for the sake of sparsity alone. Before discussing the details regarding model selection, it is intuitive to think about the process of estimating an undirected Gaussian graphical model.

Given a set of data which is multivariate normal and with an estimated covariance matrix $\hat{\Sigma}$, by inverting and standardizing according to equation 3.2, the partial correlation coefficients can be displayed graphically as a set of edges in accordance with the dependence structure. This dependence structure is unknown, and is estimated using a sample from the population of interest. Given that the observations

are a random sample, the researcher can be confident that the estimate is consistent with the true population parameter, Σ. In the case of graphical modelling, the unknown quantity is the aforementioned dependence structure Ω. Considering the inherent sampling variation of real world data, how is the parameter Σ estimated?

The classical way of estimating the covariance matrix $\hat{\Sigma}$ is by using the sample covariance matrix;

$$S = \frac{1}{(n-1)} \sum_{k=1}^{N} (x_{ik} - \bar{x}_i)(x_{jk} - \bar{x}_j)^T. \tag{3.5}$$

As known, such statistic is unbiased, [45], and yields an acceptable estimate when $n >> p$ due to characteristics of the maximum likelihood estimator, [41]. The covariance matrix is a central part when performing statistical analysis, but when using graphical models, it is the main determining factor of the results. Therefore, estimating the covariance matrix and estimating an undirected Gaussian graphical model should not be viewed as separate entities. When using real world data, it will always inherit some sampling variation, [13]. As a consequence, features that have no dependency between each other may wrongly exhibit presence of covariance, correlation and also partial correlation. The purpose of model selection is in limiting these to obtain a more robust estimate of the dependence structure. Several methods are available for graphical models, two of them being stepwise selection and LASSO.

3.2.1 Stepwise Model Selection

Forward and backwards selection are methods used in statistics in which there are needs of some sort of model selection. Each step of the selection procedure corresponds to the addition or deletion of a feature in either *the null*(no features present in the model as a starting point) or *saturated model*(all features used as a starting point). This process of selection is typically made by using an α significance level at each step in the process for evaluation, [9]. Applied to graphical models, these operations correspond to the addition or deletion of an edge starting from either the null or saturated model. Therefore, the selection process considers each element of the precision matrix, and chooses the best fitting model by subsequently adding or removing coefficients in the aforementioned matrix based on a selection criteria.

Applied to the simple example in Figure 3.1, the stepwise procedure conducts a series of hypothesis tests performed for each element u_{ij} of the precision matrix.

Using backward selection as an example, for a model $\mathcal{M}_1$ which is contained in the full model $\mathcal{M}_0$, a likelihood-ratio test is performed.

$$H_0 : u_{ij} = 0 \qquad versus \qquad H_1 : u_{ij} \neq 0. \tag{3.6}$$

The likelihood-ratio test can be conducted in the following manner;

$$\text{likelihood-ratio test} = nlog\left[\frac{det(\hat{\Omega}_0)}{det(\hat{\Omega}_1)}\right], \tag{3.7}$$

where $\hat{\Omega}_0$ is the estimated precision matrix under $\mathcal{M}_0$, and $\hat{\Omega}_1$ is the estimated precision matrix under $\mathcal{M}_1$. Under $\mathcal{M}_0$, the likelihood-ratio test statistic is asymptotically χ^2 distributed. The degrees of freedom corresponds to the difference in the number of edges in $\mathcal{M}_1$ and $\mathcal{M}_0$. F-tests, AIC and BIC can also be used as a selection criteria when performing stepwise selection. F-tests will not be further explored, as these are more suited for small-size samples, [25]. AIC and BIC will be discussed in the upcoming section, both in the context of stepwise selection and also in the context of selecting a optimal tuning parameter in the graphical LASSO.

Before discussing feature selection methods such as LASSO, a few limitations regarding stepwise selection is worth mentioning.

Stepwise selection involves performing multiple χ^2 or F-tests for each model under consideration. This might not be an appropriate use of significance tests. However, the most prominent reason why stepwise selection is currently not considered to be the most suitable method of model selection available is for computational reasons. A model with ten features yields a total number of 45 edges in the saturated model, thus yielding the total number of potential models equal to 2^{45}, [11]. Given a high-dimensional data with thousands of features, this method is infeasible due to the vast number of potential models. There are methods of reducing the number of models under consideration, such as only considering a subset of edges for removal at each step. In general, this method is not preferable in high-dimensions. Nevertheless, stepwise methods can be a viable method for estimating a graph in some settings, and are included in this paper for comparison to the modern method developed for high dimensional graphs; the graphical LASSO.

3.2.2 Graphical LASSO

LASSO is a feature selection method. When used in linear regression, it limits the size of the estimated coefficients in accordance with a penalty parameter, λ. A larger

penalty term yields smaller estimated coefficients, where some are set equal to zero, [28]. An extension of this method has been implemented for graphical models, known as the graphical LASSO.

Assuming that the multivariate data $X_i = (x_1, \ldots, x_p)_i$ is normally distributed, the graphical LASSO estimates the precision matrix by maximizing the *penalized log-likelihood*

$$\hat{\Omega} = \underset{\Omega>0}{\operatorname{argmax}} \Big[l(\Omega) - \lambda \sum_{j \neq k} \big| u_{jk} \big| \Big], \qquad \Omega = \begin{bmatrix} u_{11} & \cdots & u_{1d} \\ \vdots & \ddots & \vdots \\ u_{d1} & \cdots & u_{dd} \end{bmatrix} \tag{3.8}$$

through numerical optimization.

Here the scalar function $l(\Omega)$ is defined as

$$l(\Omega) = \frac{n}{2} \ln \big(\det \Omega \big) - \frac{n}{2} \operatorname{trace} \big(\Omega \hat{\Sigma}_n \big) - \frac{np}{2} \ln(2\pi), \tag{3.9}$$

with $\hat{\Sigma}$ being the sample covariance and p is the number of features, [47]. If the optimization task is successfully solved, then the outcome can serve as a sparse estimate of the precision matrix. In practice, these equations imply that the graphical LASSO maximizes equation 3.8 such that the size of the estimated coefficients are limited in accordance with a given tuning parameter λ, forcing some to be estimates to be zero. An interested reader can check appendix where mathematical derivation and motivation of the $l(\cdot)$ function in equation 3.9 are given.

As shown by Meinshausen and Buhlman, this method greatly improves the computational load in comparison with forward and backward selection methods, as their version of the graphical LASSO can easily estimate the precision matrix where the number of features are more than 1000, in comparison to forward selection which struggles already to solve problems with 30 features, [38, 16]. Aside from the computational upside, the graphical LASSO can also be used in the setting in which $n < p$. Given the situation in which the number of variables far exceeds the number of observations, not uncommon within genomics, the sample covariance matrix does not perform well, [33].

When $n < p$, an increasing number of eigenvalues of the covariance matrix S become zero. Therefore, it is no longer full rank and cannot necessarily be inverted, [45]. The technical details are beyond the scope of this thesis. In short, by limiting the size of the estimated coefficients and by promoting sparsity in the covariance

matrix, the graphical LASSO is able to estimate the precision matrix even in the setting when $n < p$.

Although the problem of $n < p$ is central within many fields of application, it is not a part of the simulation. For all simulated data, $n > p$, thus enabling a direct comparison between stepwise selection and the graphical LASSO. Many methods have been developed that utilizes the LASSO algorithm when estimating the precision matrix. In spite of there being differences in the technical aspects between specific algorithms, the way in which they choose the optimal λ is of primary interest. A key aspect of this method is that a single value of λ is seldom used. Instead, a number of models are estimated for a range of different values of λ, [13]. Choosing the best model is therefore equivalent of choosing the optimal λ. Several methods, such as cross-validation and BIC, are implemented in different functions and packages. These are covered in the next chapter.

Chapter 4

Packages in R

Estimating undirected Gaussian graphical models in **R** can be done using available packages at CRAN. The packages used herein are quite extensively developed and include an abundance of functions outside of the ones used within this thesis. For denoting a certain package, it is written using **bold Sen serif**. For specific functions within packages, it is highlighted by using Sen serif. These are used to clarify the text, as packages and functions occasionally share identical names. All functions will not be evaluated in the simulation due to time-limitations, but are included to provide an overview of available estimation techniques to the novel reader. The packages and functions used in the simulation are presented in Table 4.1.

4.1 CVglasso

CVglasso acts as an wrapper package for the original for **glasso** package developed to solve the optimization equation 3.8, [29, 20]. **CVglasso** extends the original package by providing the user the ability to use cross-validation for selecting the tuning parameter λ. The default specifications within the package is 5-fold cross-validation for 10 values of λ. The default number of iterations for convergence are set to 10 000. Each iteration divides the data set into 80% training data and 20% test data and creates for each 5-fold, 10 different graphical models each using the 10 lambda values.

Amongst these ten models, the model with the lowest cross-validation errors is saved. If the default cross-validation error tolerance of 0.0001 is reached prior to completing all 10 000 iterations, the iterations are stopped and the found precision is outputted. If, however, the tolerance is not reached prior to completing all 10 000 iterations, then the algorithm outputs the precision matrix with the lowest errors

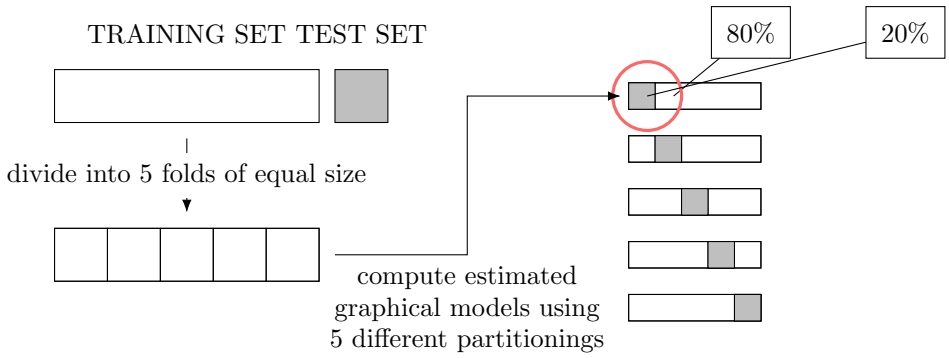


Figure 4.1: Illustration of one iteration using 5-fold cross-validation

from the saved list of matrices. Conceptual illustration of one iteration estimating the models can be seen in Figure 4.1.

The package applied the following criteria for evaluating the cross-validation errors: bayesian information criteria (BIC), akaike information criterion (AIC) and maximized log likelihood function. The criterias AIC and BIC are defined as follows:

$$\text{AIC} = 2|\mathcal{E}| - 2ln(\hat{L}), \tag{4.1}$$

$where, \mathcal{E}$ is the number of nonzero edges, $ln(\hat{L})$ = the maximized log-likelihood function

$$\text{BIC} = |\mathcal{E}|ln(n) - 2ln(\hat{L}), \tag{4.2}$$

$where, \mathcal{E}$ is the number of nonzero edges, $ln(\hat{L})$ = the maximized log-likelihood function,

n = the number of observations

One major advantage using **CVglasso** is the incorporation of the package **parallel**. It allows for parallelization which provides the computer central processing unit (CPU), the ability to use multiple CPU cores simultaneously when computing the optimal tuning parameter, drastically reducing computation time, [39].

4.2 bootnet

Similar to **CVglasso**, the **bootnet** package acts as an wrapper package for 24 different packages including **glasso**, **huge**, **qgraph** and many more. The package is aimed at condensing most available graphical estimation packages into one, as well

as extending their capabilities with new proprietary functions, [14].

The primary function within the package, is the estimateNetwork function. With estimateNetwork one can estimate the precision matrix by specifying the method that shall be used, originating from the wrapped packages. The featured estimation methods used include EBICglasso, pcor, huge, adalasso and mgm, [13].

- EBICglasso: Estimates a partial correlation network using a graphical LASSO to mitigate spurious dependencies between variables. The function estimates several networks for a range of different values of the tuning parameter λ used to optimize equation 3.8, and selects the best fitting network by minimizing the extended bayesian information criterion; EBIC, featured in the package **qgraph**. EBIC extends the BIC by adding the final term with a tuning parameter γ. If $\gamma = 0$, the optimization problem is solved using regular BIC.

$$\text{EBIC} = -2ln(\hat{L}) + |\mathcal{E}|ln(n) + 4|\mathcal{E}|\gamma ln(p), \tag{4.3}$$

  where, $p =$ number of features, $ln(\hat{L}) =$ the maximized log-likelihood function,

  $\mathcal{E}$ is the number of nonzero edges, $n =$ the number of observations,

  $\gamma \in [0, 1]$ is the EBIC tuning parameter

  The EBIC generates better properties in the setting of graphical models compared to the regular BIC, [18], conditional on that the tuning parameter γ has been selected appropriately. The extension penalizes models with high dimensionality harder, and is especially useful in dealing with a situation where $n << p$, [6].

- pcor: Estimates an unregularized partial correlation network. Originates from the package **corpcor**, [44].

- huge: Estimates the partial correlation network using the graphical LASSO. The default tuning parameter is the EBIC. Thorough description is done within section 4.3. Key differences between selecting EBICglasso lies within default values of tuning parameter λ and default application of a semiparametric transformation function huge.npn, when preparing the data. These functions originates from the **huge** package.

- adalasso: Originating from the **parcor** package, this function estimates the precision matrix using a weighted graphical LASSO, in which the tuning parameter is selected using cross-validation, [32, 51].

Summarizing, the package features several functions useful for determining the strength of the edges, the stability of the results, several methods for constructing confidence intervals using bootstrap, and possibilities involving simulation in order to test different estimation techniques. As well as gaining insight regarding the required sample size needed for a meaningful analysis, see the bootnet function within the **bootnet** package for more details, [14].

4.3 huge

The "High-dimensional Undirected Graph Estimation" package, or in short **huge** provides the user with the ability of estimating an undirected Gaussian graphical model, [30]. It features several options useful for graphical modelling, such as a built in function for generating data and methods of screening rules, [48, 31], which are useful from a computational standpoint when operating big data. Its primary function is huge. The function includes four methods for estimating the precision matrix. These are;

- glasso: The function estimates 10 precision matrices for 10 different randomly chosen values of the tuning parameter λ used to optimize equation 3.8. There are three selection criteria for choosing λ. These are EBIC, Stability Approach to Regularization Selection, (StARS) and Rotation information criterion, (RIC).

  StARS is based on the variability of subsamples of the data, [34]. The procedure generates subsamples and estimates a graph for each subsample. For λ equal to zero, the total variability across subsamples will be very small since all subsamples will generate a dense graph. Likewise, if the tuning parameter tends to infinity, the total variability will be zero since all edges would be excluded. λ is therefore set at a prespecified, high value in order to generate a sparse graph, while still maintaining the edges which vary the least across subsamples.
  RIC an additional method for selecting the optimal tuning parameter. For the technical details, the reader is referred to, [36]. It is not extensively discussed, since it lacks theoretical support for recovering the true graph structure in higher dimensions, [30, 50].

- mb: Utilizes the Meinshausen-Buhlmann graph estimation method, [38]. The conditional independence assumption allows one to consider the estimation

method to be equivalent to variable selection for linear regression with LASSO. Each feature is consequently regressed using the other as predictors. The estimated coefficients of each regression is used as elements within the precision matrix. This breaks down the graph structure to be estimated, with a total of $(2^{(p^2-p)/2})$ edges, into a neighbourhood selection problem with $2^{(p-1)}$ number of neighbourhoods.

- ct: Correlation thresholding graph estimation. Given that the number of nonzero elements u in the true dependence graph Ω is known, correlation thresholding identifies the u largest off-diagonal elements of the covariance matrix and sets all other elements to zero. Therefore, it does not involve any formal optimization task. It has been shown that under certain conditions, this method generates similar results to the graphical LASSO, [46].

- tiger: Tuning insensitive graph estimation. Asymptotically tuning-free LASSO estimation. By fixing the tuning parameter at the predefined value

$$\lambda = \zeta\pi\sqrt{\frac{log(p)}{2n}}, \tag{4.4}$$

  where n is the number of observations, p number of features and ζ is the one and only input to be chosen. This makes the method tuning-insensitive, since ζ is unrelated to the data. Setting

$$\zeta = \frac{\sqrt{2}}{\pi}, \tag{4.5}$$

  has been shown through simulation to yield precise estimates of the true network structure, [35].

A caveat implemented within the package when estimating graphs is the ability to specify an application of a screening rule. Two screening rules are available, when using estimation methods mb or glasso application of the so-called lossy screening is possible through the function scr(). When estimating graphs utilizing glasso, by default the so-called lossless screening is applied.

In essence, the lossy screening rule reduces feature dimensionality, hence preserving computational power. Yet, the rules' performance suffers from statistical bias due to only working optimally under certain conditions, [17]. In contrast, the lossless screening rule utilizes the Karush-Kuhn-Tucker conditions to reduce the required estimation size of the precision matrix. By nature, achieving equivalent statistical efficiency, [31, 48].

After graph estimation is performed using one of the above mentioned methods, one utilizes the function huge.select, in order to select the optimal precision matrix given a certain tuning parameter selection criteria.

4.4 qgraph

The origin and primary use of this package is to construct and visualize network models for psychometric data. Its main function, qgraph, is a wrapper function used to visualise a variety of weighted network structures. The weights do not necessarily have to be partial correlations, as standard correlations and p-values can also be displayed graphically, and the connections between nodes can either be directed or undirected, [12].

Functions within the package are mainly designed for estimation of a Gaussian graphical model. ggmModSelect is the main function which features estimation of such a model using both stepwise selection and the graphical LASSO, using the EBIC as method for selecting the optimal λ. The function takes as input a covariance matrix, and the following options are available for estimation;

- glasso: Shrinkage approach to estimate a sparse network, as implemented in previous packages with the difference being that the function is originated from the package **Lavaan**.
- Full: Starts from the full, saturated model before applying backwards selection.
- Empty: Starts from an empty network structure, before adding edges by forward selection, [16].

The graphical LASSO can be applied in combination to using forward or backward selection. In this way, the graphical LASSO is applied first, generating a more sparse network in which the best fit is chosen according by EBIC selection criteria. Subsequently, backward selection is applied to the sparse network, where edges are selected using the same procedure as previously described. **qgraph** is solely used in order to compare the stepwise model selection method to the more sophisticated variants of the LASSO, featured in previously described packages. As such, by specifying the Full model, and setting the EBIC tuning parameter to zero, the applied stepwise procedure is backwards selection using the normal BIC as a selection criteria. For computational reasons, at each step in the selection process, only a subset of edges are considered for removal as specified by the logical input Subset. By only

considering edges that previously indicated a lower value of the BIC, the total number of potential models decrease in comparison to if all possible models where to be tested at each step. This lowers the computational load, and should generate faster processes of estimation.

4.5 Overview of Packages

Table 4.1

Packages and Functions Evaluated

Package	Function	Method	Selection Criteria	Evaluated
CVglasso	CVglasso	glasso	log-likelihood	Yes
			AIC	Yes
			BIC	Yes
bootnet	estimateNetwork	EBICglasso	EBIC	Yes
		pcor	-	No
		huge	-	No
		adalasso	-	No
huge	huge	glasso	EBIC	Yes
			StARS	Yes
			RIC	No
		mb	-	No
		ct	-	No
		tiger	-	No
qgraph	ggmModSelect	stepwise	glasso	No
			full	Yes
			empty	No

Six methods were not evaluated, therefore additional information regarding their respective selection criteria are omitted from Table 4.1.

Chapter 5

Data Generation and Simulation

5.1 Data Generation

Data generation was performed using the inverse of the predefined sparse precision matrices in the function mvrnorm within the **MASS** package, [42]. By predefining these precision matrices, model identifiability is fulfilled.

The inverse of the precision matrix is the covariance matrix, denoted by $\mathbf{\Omega} = \mathbf{\Sigma}^{-1}$. mvrnorm generates a multivariate normally distributed data set from 1000 and 5000 observations with mean 0 and covariance Σ.

$$\mathbf{X} \sim \mathcal{N}_p(\boldsymbol{\mu}, \mathbf{\Sigma}), \tag{5.1}$$

where p-dimensional random vector is:

$$\mathbf{X} = (X_1, \ldots, X_p)^{\mathbf{T}}, \tag{5.2}$$

and the p-dimensional mean vector is:

$$\boldsymbol{\mu} = \mathrm{E}[\mathbf{X}] = (\mathrm{E}[X_1], \mathrm{E}[X_2], \ldots, \mathrm{E}[X_p])^{\mathbf{T}}, \tag{5.3}$$

and the $p \times p$ covariance matrix is

$$\Sigma_{i,j} = \mathrm{E}[(X_i - \mu_i)(X_j - \mu_j)] = \mathrm{Cov}[X_i, X_j] \tag{5.4}$$

The predefined precision matrices were obtained using data generating algorithm 1. Input variables for the algorithm include size of the matrix and how many nonzero entries there should be within the matrix. Additionally, two constraints were also

present when generating the matrices. These were that the generated matrix must be positive definite and therefore the determinant of the matrix is nonzero. Since all matrices are predefined with a known dependence structure, all subsequent models are identifiable.

Four matrices were generated in total, with the following specifications:

Table 5.1

Description of Generated Matrices

Nr.	Matrix Size	Nonzero entries
1	5×5	6
2	15×15	10
3	25×25	16
4	100×100	51

Below it is presented how Matrix 1 is predefined, for the other matrices the relevant information is put in appendix. One should interpret the matrix in the following way; if there is a 0 between the two nodes they are considered conditionally independent of each other. Otherwise, the two nodes are considered conditionally dependent. Since the matrix is symmetric, only the upper triangular is evaluated when counting the number of nonzero entries, also ignoring the diagonal values.

Matrix 1

Illustration of Precision Matrix 1

	$Node_1$	$Node_2$	$Node_3$	$Node_4$	$Node_5$
$Node_1$	1	0	0.07	0.01	0
$Node_2$	0	1	0.16	0.73	0.02
$Node_3$	0.07	0.16	1	0	0
$Node_4$	0.01	0.73	0	1	0.11
$Node_5$	0	0.02	0	0.11	1

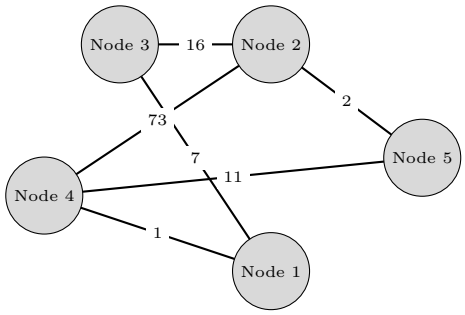


Figure 5.1: Graphical illustration of Matrix 1, in percentages

Creating the sparse matrices were done utilizing the function rsparsematrix built into the **Matrix** package, [2]. The function takes following input variables: the dimension of the matrix, proportion of nonzero entries, custom made function value insertion and if the matrix should be symmetric. Taking the inputs into account, rsparsematrix generates a random sparse matrix. However, not necessarily fulfilling the two required conditions for the matrix to be positive definite and having a determinant not equal to zero.

5.1.1 Matrix Generation Algorithm

Algorithm 1: Sparse Precision Matrix Generation

input : Amount of Variables, Amount of Nonzero entries, Starting Seed
output: A Sparse Precision Matrix with $V \times V$ dimensions

```
1  begin
2      library(Matrix);
3      set.seed(seed);
4      V ⟵ Variables;
5      N ⟵ Nonzero Entries;
6      Matrix ⟵ rsparsematrix(V, V, N, symmetric=T)
7      foreach Element defined as "nonzero" in the Matrix do
8          Select a number from X ~ U(−1, 1)
9          return(element value)
10     if |Matrix| ≠ 0 and the Matrix is positive-definite = TRUE then
11         return(Matrix)
12     else
13         while |Matrix| ≠ 0 and the Matrix is positive-definite = FALSE do
14             if while condition = FALSE then
15                 while while condition (13) = FALSE do
16                     N ⟵ N + 1 ;            // increased nonzero entries
17                     Matrix ⟵ rsparsematrix(V, V, N, symmetric=T);
18                     foreach Element defined as "nonzero" in the Matrix do
19                         Select a number from X ~ U(−1, 1)
20                         return(element value)
21                     Stop N > elements in upper triangular matrix
22             else
23                 while while condition (13) = FALSE do
24                     seed ⟵ seed + 1;       // increased and updated seed
25                     Matrix ⟵ rsparsematrix(V, V, N, symmetric=T);
26                     foreach Element defined as "nonzero" in the Matrix do
27                         Select a number from X ~ U(−1, 1)
28                         return(element value)
29             return(Matrix)
30         return(Matrix)
```

5.2 Simulation Methods

R allows for estimation of the predefined precision matrices by using the packages **huge**, **bootnet**, **CVglasso** and **qgraph**, discussed earlier in Chapter 4. The initial three packages numerically optimize equation 3.8. The key difference lies within the selection criterion of the tuning parameter λ. Within **huge**, the selection criteria EBIC and StARS are tested. For **bootnet**, the featured selection criteria is the EBIC. For **CVglasso**, cross-validation using BIC, AIC and log-likelihood are featured. The stepwise model search as described in **qgraph** is implemented using BIC as stepwise selection criteria.
The simulation method for these packages is structured in the following way:

Step 1: Take the inverse of the predefined precision matrix

Step 2: For each iteration generate a new multivariate normally distributed dataset described in data generation section, with mean 0 and covariance as the inverted precision matrix.

Step 3: For each dataset estimate an undirected Gaussian graphical model using **huge**, **bootnet**, **CVglasso** and **qgraph**.

Step 4: For each estimated model by **huge**, apply selection methods EBIC and StARS. For models estimated by **bootnet**, apply selection methods EBIC. For **CVglasso**, apply BIC, AIC and log-likelihood criteria for the tuning parameter λ. This is done in order to asses the optimal estimated precision matrix for each method and selection criteria. This step is excluded for **qgraph**, as the stepwise model search does not utilize any form of the graphical LASSO.

Step 5: For each element in the optimal precision matrix estimated by each model, if the element is nonzero, a 1 is input instead, converting the precision matrix into a binary matrix. The evaluation thus becomes, "an edge is present" = 1, "an edge is not present" = 0. Add the binary matrix results into the zero matrix.

Step 6: Repeat for all iterations and output the summed binary matrices as simulation results matrix.

5.2.1 Visualisation of Simulation and Evaluation Method

Precision Matrix

	$Node_1$	$Node_2$	$Node_3$	$Node_4$	$Node_5$
$Node_1$	1	0	0.07	0.01	0
$Node_2$	0	1	0.16	0.73	0.02
$Node_3$	0.07	0.16	1	0	0
$Node_4$	0.01	0.73	0	1	0.11
$Node_5$	0	0.02	0	0.11	1

Step 1.

Binary Precision Matrix

	$Node_1$	$Node_2$	$Node_3$	$Node_4$	$Node_5$
$Node_1$	0	0	1	1	0
$Node_2$	0	0	1	1	1
$Node_3$	1	1	0	0	0
$Node_4$	1	1	0	0	1
$Node_5$	0	1	0	1	0

Step 2.

Benchmark Matrix

	$Node_1$	$Node_2$	$Node_3$	$Node_4$	$Node_5$
$Node_1$	0	0	1000	1000	0
$Node_2$	0	0	1000	1000	1000
$Node_3$	1000	1000	0	0	0
$Node_4$	1000	1000	0	0	1000
$Node_5$	0	1000	0	1000	0

Step 3.

Estimated Precision Matrix

	$Node_1$	$Node_2$	$Node_3$	$Node_4$	$Node_5$
$Node_1$	0	0	19	0	0
$Node_2$	0	0	1000	1000	0
$Node_3$	19	1000	0	841	0
$Node_4$	0	1000	841	0	1000
$Node_5$	0	0	0	1000	0

Step 4.

Figure 5.2: Illustration of Simulation Procedure

Consider Figure 5.2, which is made for illustrative purposes of the simulation and evaluation methods. It showcases four matrices.

Step 1 starts by considering the first matrix denoted as *Precision Matrix*, which is the initial predefined Matrix 1, that the estimation methods aim to replicate. Step 2 denoted as *Binary Precision Matrix*, transforms the *Precision Matrix* in Step 1, into a binary version; the value of an element equal to 1 indicates an edge between two features being present and 0 otherwise. Step 3 utilizes the transformed *Binary Precision Matrix* in Step 2. Referring to this matrix as the *Benchmark Matrix*, as it represents a perfectly predicted precision matrix for all 1000 iterations. Step 4 considers the *Estimated Precision Matrix*, where utilization of a given selection criteria was performed to produce an estimate to the *Precision Matrix* in Step 1. This estimate is transformed into a binary version using the same logic as in Step 2, and each element is summed for all 1000 iterations into the final *Estimated Precision Matrix*.

5.3 Evaluation Methods

A *Confusion Matrix* will be used for comparing the ability of each selection criteria to estimate the predefined precision matrix. Emphasis will be put on specificity, sensitivity, accuracy and total simulation time for the evaluation.

To calculate the specificity, sensitivity and accuracy, a comparison is made between the perfect ideal results of the *Benchmark Matrix*, and the actual predicted results of the *Estimated Precision Matrix*, as illustrated in Figure 5.2. These comparisons allows for determination of numerical values for *True Positives*, *True Negatives*, *False Positives* and *False Negatives* for each selection criteria featured in the simulation.

- *True Positive* is defined as there being an edge present within the benchmark matrix as well as within the estimated optimal precision matrix.
- *True Negative* is defined as there not being an edge present in both the benchmark matrix and the estimated precision matrix.
- *False Positive* is defined as there being an edge present within the estimated precision matrix, however not within the benchmark matrix.
- *False Negative* is defined as there not being an edge estimated by the precision matrix, however being present within the benchmark matrix.

Specificity, sensitivity and accuracy are calculated as

$$Specificity = \frac{True\ Negative}{True\ Negative + False\ Positive}$$

$$Sensitivity = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

$$Accuracy = \frac{True\ Positive + True\ Negative}{True\ Positive + True\ Negative + False\ Negative + False\ Positive}$$

5.3.1 Visualisation of Evaluation Calculation

Benchmark Matrix

	$Node_1$	$Node_2$	$Node_3$	$Node_4$	$Node_5$
$Node_1$	0	0	1000	1000	0
$Node_2$	0	0	1000	1000	1000
$Node_3$	1000	1000	0	0	0
$Node_4$	1000	1000	0	0	1000
$Node_5$	0	1000	0	1000	0

Estimated Precision Matrix

	$Node_1$	$Node_2$	$Node_3$	$Node_4$	$Node_5$
$Node_1$	0	0	19	0	0
$Node_2$	0	0	1000	1000	0
$Node_3$	19	1000	0	841	0
$Node_4$	0	1000	841	0	1000
$Node_5$	0	0	0	1000	0

True Positive

	$Node_1$	$Node_2$	$Node_3$	$Node_4$	$Node_5$
$Node_1$	0	0	19	0	0
$Node_2$	0	0	1000	1000	0
$Node_3$	19	1000	0	0	0
$Node_4$	0	1000	0	0	1000
$Node_5$	0	0	0	1000	0

TP = 3019

True Negative

	$Node_1$	$Node_2$	$Node_3$	$Node_4$	$Node_5$
$Node_1$	0	1000	0	0	1000
$Node_2$	1000	0	0	0	0
$Node_3$	0	0	0	159	1000
$Node_4$	0	0	159	0	0
$Node_5$	1000	0	1000	0	0

TN = 3159

False Positive

	$Node_1$	$Node_2$	$Node_3$	$Node_4$	$Node_5$
$Node_1$	0	0	0	0	0
$Node_2$	0	0	0	0	0
$Node_3$	0	0	0	841	0
$Node_4$	0	0	841	0	0
$Node_5$	0	0	0	0	0

FP = 841

False Negative

	$Node_1$	$Node_2$	$Node_3$	$Node_4$	$Node_5$
$Node_1$	0	0	981	1000	0
$Node_2$	0	0	0	0	1000
$Node_3$	981	0	0	0	0
$Node_4$	1000	0	0	0	0
$Node_5$	0	1000	0	0	0

FN = 2981

Figure 5.3: Illustration of Evaluation Calculation

Analyzing Figure 5.3, which is an illustrative extension building upon the matrices in Figure 5.2 calculated by algorithms 2 to 6, stored within the appendix. These algorithms computes values for the definitions above in 5.3, comparing the *Benchmark Matrix* to the *Estimated Precision Matrix* to find the correct matrices. Outputting only a single value, for each definition by summing all elements within the found matrix and dividing it by two. This is due to the matrices being symmetric and information from a triangular matrix being sufficient to calculate the specificity, sensitivity and accuracy.

5.4 Computing Setup

These computational simulations will be conducted using **R** version 4.0.5, [40], on the following setup, using default BIOS settings with no overclocking or performance enhancing software. The motivation behind this decision is to provide a fair benchmark comparison for these specific components.

Table 5.4

Description of Computing setup

Operating System	**CPU**	**RAM**	**GPU**
Windows 10, 64 bit	Intel Core i7-7700k, 4.2GHz	16 GB, 2133MHz	Nvidia GTX 1080ti

Chapter 6

Results

6.1 Simulation Results

After generating the precision matrices, generating the random sets of data, applying the numerical optimization methods and the steps described in Chapters 4 and 5, the results of the simulation are presented in Table 6.1 and 6.2. Rounding off error is present due to 3 digits. Float point of **R** is IEEE 754 standard.

Since the **R** code will not be provided below within the appendix due to the extensive length, if the reader wishes to replicate the results of the study, contact the authors by e-mail.[1]

[1] arsh0015@student.umu.se

Table 6.1

Simulation Results 1 with 1000 observations

Selection Method	Matrix Size	Sensitivity	Specificity	Accuracy	Time
huge ebic	5×5	0.603	0.849	0.701	11.4 m
huge stars	5×5	0.157	1	0.494	2.990 h
bootnet ebic	5×5	0.690	0.749	0.713	4.954 s
CVglasso log-likeli	5×5	0.708	0.607	0.668	15.264 s
CVglasso BIC	5×5	0.58	0.822	0.677	14.673 m
CVglasso AIC	5×5	0.662	0.702	0.678	15.619 m
qgraph stepwise	5×5	0.565	0.987	0.734	1.422 m
huge ebic	15×15	0.725	0.974	0.950	11.62 m
huge stars	15×15	0.7	0.981	0.954	2.995 h
bootnet ebic	15×15	0.812	0.935	0.923	9.636 s
CVglasso log-likeli	15×15	0.877	0.703	0.719	20.773 s
CVglasso BIC	15×15	0.737	0.952	0.931	15.323 m
CVglasso AIC	15×15	0.809	0.866	0.861	15.737 m
qgraph stepwise	15×15	0.802	0.989	0.971	12 h
huge ebic	25×25	0.903	0.945	0.943	11.36 m
huge stars	25×25	0.831	0.952	0.946	3.113 h
bootnet ebic	25×25	0.929	0.906	0.907	28.038 s
CVglasso log-likeli	25×25	0.905	0.904	0.904	32.935 s
CVglasso BIC	25×25	0.844	0.945	0.940	15.481 m
CVglasso AIC	25×25	0.888	0.917	0.915	17.731 m
qgraph stepwise	25×25	-	-	-	-
huge ebic	100×100	0.894	0.995	0.994	11.89 m
huge stars	100×100	-	-	-	-
bootnet ebic	100×100	0.903	0.989	0.988	20.433 m
CVglasso log-likeli	100×100	0.775	0.987	0.985	4.105 m
CVglasso BIC	100×100	0.774	0.987	0.985	17.084 m
CVglasso AIC	100×100	0.775	0.987	0.985	19.267 m
qgraph stepwise	100×100	-	-	-	-

Table 6.2

Simulation Results 2 with 5000 observations

Selection Method	Matrix Size	Sensitivity	Specificity	Accuracy	Time
huge ebic	5×5	0.576	0.862	0.690	11.387 m
huge stars	5×5	0.206	1	0.523	3.03 h
bootnet ebic	5×5	0.752	0.712	0.736	6.786 s
CVglasso log-likeli	5×5	0.727	0.756	0.739	20.886 s
CVglasso BIC	5×5	0.690	0.806	0.736	15.513 m
CVglasso AIC	5×5	0.730	0.757	0.740	18.12 m
qgraph stepwise	5×5	0.691	0.994	0.812	1.371 m
huge ebic	15×15	0.700	0.979	0.952	11.848 m
huge stars	15×15	0.315	0.991	0.926	2.78 h
bootnet ebic	15×15	0.965	0.920	0.924	16.624 s
CVglasso log-likeli	15×15	0.943	0.853	0.862	37.212 s
CVglasso BIC	15×15	0.894	0.940	0.936	15.657 m
CVglasso AIC	15×15	0.933	0.873	0.880	17.929 m
qgraph stepwise	15×15	-	-	-	-
huge ebic	25×25	0.917	0.949	0.947	11.881 m
huge stars	25×25	0	1	0.95	2.98 h
bootnet ebic	25×25	0.940	0.883	0.886	35.794 s
CVglasso log-likeli	25×25	0.923	0.957	0.955	1.026 m
CVglasso BIC	25×25	0.921	0.956	0.955	15.554 m
CVglasso AIC	25×25	0.923	0.956	0.955	18.799 m
qgraph stepwise	25×25	-	-	-	-
huge ebic	100×100	0.893	0.997	0.996	12.904 m
huge stars	100×100	-	-	-	-
bootnet ebic	100×100	0.951	0.979	0.979	12.24 m
CVglasso log-likeli	100×100	0.767	0.997	0.996	6.657 m
CVglasso BIC	100×100	0.767	0.997	0.995	18.556 m
CVglasso AIC	100×100	0.767	0.997	0.995	22.08 m
qgraph stepwise	100×100	-	-	-	-

Simulation results for larger matrices using **qgraph** stepwise model, were omitted due to the extensive computational time of 12 hours for the 15×15 matrix presented within Table 6.1. Furthermore, results of **huge** stars for the 100×100 matrix for both simulations, were infeasible to compute due to RAM memory limitations.

Chapter 7

Discussion

7.1 Results

Reviewing and interpreting the results of the simulation, a few aspects should be noted. Firstly, the simulation time is a measurement of the iterative process as a whole. The measurement do not reflect estimation of a single network, but estimation of 1000 networks as well as storing the results for each method. Secondly, all true network structures were relatively sparse, a computational requirement for obtaining a matrix with the specified structure as well as being positive-definite. A model which only estimates zeros would be both specific and accurate. This is due to the proportion of zero to non-zero elements within the predefined precision matrices. Therefore, only considering accuracy would be misleading, as highly specific methods tend to likewise be highly accurate.

Consider StARS as implemented in **huge** as a selection critera for the tuning parameter within graphical LASSO. In general, the estimated network was highly sparse. Therefore, it is very specific, as it correctly identified true negatives to a large extent. A side effect of this estimate is low sensitivity, thus incorrectly estimating independencies between dependent features. This is not unexpected, as the method is developed for use in high-dimensions to generate sparse graphs, [34]. Given the nature of the simulation with relatively few nonzero entries in the true precision matrix, the measurement of accuracy is for that reason to be interpreted with caution. In summary, while being highly specific, this method is generally not very sensitive, while also being slower in comparison to other methods.

The results regarding **qgraphs** stepwise model using BIC as the selection criterion is in line with previous research. In terms of specificity, it performs well when $p = 5$ and $p = 15$. Only considering these sets of data, it is the most accurate. The problem arises in higher dimensions, as the simulation was extremely time consuming. 1000 iterations, using $p = 15$ and 1000 observations, the simulation was completed in circa 12 hours. As such, stepwise selection was omitted for estimating the larger sets of data. This is no surprise, since the package description states that the stepwise procedure is very slow when $p > 30$. Given that high-dimensional data is not uncommon in modern statistics, this method is infeasible in such instances. Using a smaller data set, it yields similar or slightly better results in comparison to the other estimation techniques. It is a viable alternative to the graphical LASSO given a data with only a few features, but it is non-applicable for higher dimensions.

The graphical LASSO using EBIC as selection criteria implemented in the **bootnet** package produced competitive results across all quantitative measurements. It exhibited high accuracy as an effect of both high sensitivity and specificity. The algorithm was also very fast, its slowest time was just over 20 minutes for completing 1000 iterations using the 100×100 data with 1000 observations. Aside from the results, it was also well-documented and easy to work with. The EBIC as implemented in **huge** displayed similar results, thus confirming the findings of [18], namely that the EBIC is useful as a selection criterion for the graphical LASSO. Comparing timings between the packages both implementing the EBIC, there are some variation. As the **bootnet** package generally becomes slower in higher dimensions, the EBIC as implemented in **huge** is generally stable at around 11 minutes for all iterations regardless of p.

Using cross-validation as a method for selecting the tuning parameter λ as implemented in **CVglasso**, also outputs adequate results for both sample sizes. All implementations of cross-validation yields comparable results to the EBIC using $p = 25$ or less. When $p = 100$, cross-validation is generally less sensitive in comparison to the EBIC. All variants of the cross-validation have a sensitivity of 0.767 when $n = 5000$, which is lower compared to the EBIC as implemented in **bootnet**(0.951) and **huge**(0.893). Apart from the actual results, the expected main drawback of using cross-validation to choose tuning parameter was in regards to simulation time. Reviewing the current literature, [35], it is considered to be a relatively challenging method from a computational standpoint. Based on our results, the cross-validation using AIC or BIC as selection criteria were the slowest methods apart from StARS and the stepwise procedure. Surprisingly, the cross-validation using log-likelihood as

selection criteria was among the fastest. This may be a result of the relatively low-dimensional matrices used in our simulation, as these may not be large enough for cross-validation to be computationally infeasible. Still, being that cross-validation is a well-known and proven technique within statistical analysis, its use within graphical model estimation cannot be opposed based on our results, at least when the number of features are low to moderate.

7.2 Concluding Remarks and Future Research

A vital assumption for the undirected Gaussian graphical model is normality. In the case of multivariate normality, all conditional distributions are also normal. By simulating data, this assumption is fulfilled and a given set of edges can be interpreted as conditional probabilities. Using real-world data, this assumption may or may not be fulfilled. As such, the results of our simulation regarding the ability of each method to extract the true network structure, can and should only be interpreted within the context of this perfect scenario. The carry-over effect from our results to real world data is an interesting yet hard question to answer. All models rely on the same fundamental assumption, but the different methods could be more or less robust to situations in which the normality assumption is questionable. For some insight, [26] concluded that the undirected Gaussian graphical model was quite robust when comparing the resulting network structure to that of a semi-parametric Gaussian copula graphical model. Given that their log-transformed set of features were not perfectly normally distributed, it provides some indication that an undirected Gaussian graphical model might be robust to the normality assumption. It does not give any feedback as to which particular method, being stepwise or the many implementations of the graphical LASSO, that provides the most robust estimate. As such, the suggestions of future research are;

- How robust is an undirected Gaussian graphical model to data which is not multivariate normal?
- Are there differences in the results of different estimation techniques in regards to non-normality?

Bibliography

[1] Allali, Abdelwahab and Amor, Oueslati and Abdelwahed, Trabelsi. "The analysis of the interdependence structure in international financial markets by graphical models". In: *International Research Journal of Finance and Economics ISSN Issue* 15 (Jan. 2008), pp. 1450–2887.

[2] Bates, Douglas M. and Maechler, Martin and Davis, Timothy A. and Oehlschlägel, Jens and Riedy, Jason and R Core Team. *Sparse and Dense Matrix Classes and Methods*. 2021. URL: https://cran.r-project.org/web/packages/Matrix/Matrix.pdf.

[3] Behrens, John T. "Principles and Procedures of Exploratory Data Analysis". In: *American Psychological Association, Inc* Vol. 2.2 (1997), pp. 131–106. DOI: https://doi.org/10.1037/1082-989X.2.2.131. URL: https://core.ac.uk/download/pdf/193648223.pdf.

[4] Benet, Veronica and John, Oliver. "Los Cinco Grandes Across Cultures and Ethnic Groups: Multitrait Multimethod Analyses of the Big Five in Spanish and English". In: *Journal of personality and social psychology* 75 (Oct. 1998), pp. 729–50. DOI: 10.1037//0022-3514.75.3.729.

[5] Boginski, Vladimir and Butenko, Sergiy and Pardalos, Panos M. "Statistical analysis of financial networks". In: *Computational Statistics Data Analysis* 48.2 (2005), pp. 431–443. ISSN: 0167-9473. DOI: https://doi.org/10.1016/j.csda.2004.02.004. URL: https://www.sciencedirect.com/science/article/pii/S0167947304000258.

[6] Chen, Jiahua and Chen, Zehua. "Extended Bayesian information criteria for model selection with large model spaces". In: *Biometrika* 95.3 (Sept. 2008), pp. 759–771. ISSN: 0006-3444. DOI: 10.1093/biomet/asn034. eprint: https://academic.oup.com/biomet/article-pdf/95/3/759/578041/asn034.pdf. URL: https://doi.org/10.1093/biomet/asn034.

[7] Chun, Hyonho and Chen, Min and Li, Bing and Zhao, Hongyu. "Joint conditional Gaussian graphical models with multiple sources of genomic data". In: *Multivariate Behavioral Research* 4 (2013), p. 294. URL: https://doi.org/10.3389/fgene.2013.00294.

[8] Cramer, Angélique O. J. and van der Sluis, Sophie and Noordhof, Arjen and Wichers, Marieke and Geschwind, Nicole and Aggen, Steven H. and Kendler, Kenneth S. and Borsboom, Denny. "Dimensions of Normal Personality as Networks in Search of Equilibrium: You Can't Like Parties if You Don't Like People". In: *European Journal of Personality* 26.4 (2012), pp. 414–431. DOI: https://doi.org/10.1002/per.1866. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/per.1866.

[9] Drton, Mathias and Perlman, Michael D. "A SINful approach to Gaussian graphical model selection". In: *Journal of Statistical Planning and Inference* 138.4 (2008), pp. 1179–1200. ISSN: 0378-3758. DOI: https://doi.org/10.1016/j.jspi.2007.05.035. URL: https://www.sciencedirect.com/science/article/pii/S0378375807002303.

[10] Drton, Mathias and Perlman, Michael D. "Multiple Testing and Error Control in Gaussian Graphical Model Selection". In: *Statistical Science* 22.3 (2007), pp. 430–449. DOI: 10.1214/088342307000000113. URL: https://doi.org/10.1214/088342307000000113.

[11] Edwards, David. *Introduction to Graphical Modelling*. 2nd ed. Springer Texts in Statistics. Springer-Verlag New York, 2000, pp. 158–172. ISBN: 978-0-387-95054-9. DOI: 10.1007/978-1-4612-0493-0.

[12] Epskamp, Sacha and Cramer, Angélique O.J. and Waldorp, Lourens J. and Schmittmann, Verena D. and Borsboom, Denny. "qgraph: Network Visualizations of Relationships in Psychometric Data". In: *Journal of Statistical Software* 48.4 (2012), pp. 1–18. ISSN: 1548-7660. DOI: 10.18637/jss.v048.i04. URL: https://www.jstatsoft.org/v048/i04.

[13] Epskamp, Sacha and Fried, Eiko I. "A tutorial on regularized partial correlation networks". In: *Psychological methods* 23.4 (Dec. 2018), pp. 617–634. ISSN: 1082-989X. DOI: 10.1037/met0000167. URL: https://doi.org/10.1037/met0000167.

[14] Epskamp, Sacha and Fried, Eiko I. *Bootstrap Methods for Various Network Estimation Routines*. 2020. URL: https://cran.r-project.org/web/packages/bootnet/bootnet.pdf.

[15] Epskamp, Sacha and Waldorp, Lourens J. and Mõttus, René and Borsboom, Denny. "The Gaussian Graphical Model in Cross-Sectional and Time-Series Data". In: *Multivariate Behavioral Research* 53.4 (2018). PMID: 29658809, pp. 453–480. DOI: 10.1080/00273171.2018.1454823. URL: https://doi.org/10.1080/00273171.2018.1454823.

[16] Epskamp, Sacha, Costantini, Giulio, Haslbeck, Jonas, Isvoranu, Adela, Cramer, Angelique O. J., Waldorp, Lourens J., Schmittmann, Verena D., Borsboom, Denny. *Graph Plotting Methods, Psychometric Data Visualization and Graphical Model Estimation*. 2021. URL: https://cran.r-project.org/web/packages/qgraph/qgraph.pdf.

[17] Fan, Jianqing and Lv, Jinchi. "Sure independence screening for ultrahigh dimensional feature space". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 70.5 (2008), pp. 849–911. DOI: https://doi.org/10.1111/j.1467-9868.2008.00674.x. eprint: https://rss.onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-9868.2008.00674.x. URL: https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-9868.2008.00674.x.

[18] Foygel, Rina and Drton, Mathias. *Extended Bayesian Information Criteria for Gaussian Graphical Models*. 2010. arXiv: 1011.6640 [math.ST].

[19] Friedman, Jerome and Hastie, Trevor and Tibshirani, Robert. "Sparse inverse covariance estimation with the graphical lasso". In: *Biostatistics* 9.3 (Dec. 2007), pp. 432–441. ISSN: 1465-4644. DOI: 10.1093/biostatistics/kxm045. eprint: https://academic.oup.com/biostatistics/article-pdf/9/3/432/17742149/kxm045.pdf. URL: https://doi.org/10.1093/biostatistics/kxm045.

[20] Galloway, Matt. *Lasso Penalized Precision Matrix Estimation*. 2018. URL: https://cran.r-project.org/web/packages/CVglasso/CVglasso.pdf.

[21] Gogtay, Nithya J. and Thatte, Urmila M. "Principles of Correlation Analysis". In: *The Journal of the Association of Physicians of India* 65 (Mar. 2017), pp. 78–81.

[22] Hartemink, Alexander and Gifford, David and Jaakkola, Tommi and Young, Richard. "Using graphical models and genomic expression to statistically validate models of genetic regulatory networks". In: *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing* 6 (Feb. 2001), pp. 422–33. DOI: 10.1142/9789814447362_0042.

[23] Hastie, Trevor and Tibshirani, Robert and Friedman, Jerome. *The elements of statistical learning: data mining, inference, and prediction*. Second. Springer Science, Business Media, 2009, 625–630.

[24] Hegde, Priti and Qi, Rong and Abernathy, Kristie and Gay, Cheryl and Dharap, Sonia and Gaspard, Renee and Hughes, J.E. and Snesrud, Erik and Lee, Norman and Quackenbush, John. "A Concise Guide to cDNA Microarray Analysis". In: *BioTechniques* 29 (2000), pp. 548–562. URL: https://www.future-science.com/doi/pdf/10.2144/00293bi01.

[25] Højsgaard, Søren and Edwards, David and Lauritzen, Steffen. *Graphical Models with R*. Use R! Springer Science+Business Media, May 2012, p. 87. ISBN: 978-1-4614-2298-3. DOI: 10.1007/978-1-4614-2299-0.

[26] Iqbal, Khalid and Buijsse, Brian and Wirth, Janine and Schulze, Matthias B. and Floegel, Anna and Boeing, Heiner. "Gaussian Graphical Models Identify Networks of Dietary Intake in a German Adult Population". In: *The Journal of Nutrition* 146.3 (Jan. 2016), pp. 646–652. ISSN: 0022-3166. DOI: 10.3945/jn.115.221135. URL: https://doi.org/10.3945/jn.115.221135.

[27] James, Gareth and Witten, Daniela and Hastie, Trevor and Tibshirani, Rob. *ISLR: Data for an Introduction to Statistical Learning with Applications in R*. R package version 1.2. 2017. URL: %7Bhttps://CRAN.R-project.org/package=ISLR%7D.

[28] James, Gareth and Witten, Daniela and Hastie, Trevor and Tibshirani, Robert. *An Introduction to Statistical Learning: with Applications in R*. Springer, 2013, p. 70–71, 219. URL: https://www.statlearning.com/.

[29] Jerome, Friedman and Hastie, Trevor and Tibshirani, Robert. *The Graphical Lasso: Estimation of Gaussian Graphical Models*. 2019. URL: https://cran.r-project.org/web/packages/glasso/glasso.pdf.

[30] Jiang, Haoming and Fei, Xinyu and Liu, Han and Roeder, Kathryn and Lafferty, John and Wasserman, Larry and Li, Xingguo and Zhao, Tuo. *High-Dimensional Undirected Graph Estimation*. R package version 1.3.4.1. 2020. URL: https://cran.r-project.org/web/packages/huge/huge.pdf.

[31] Jiang, Haoming and Fei, Xinyu and Liu, Han and Roeder, Kathryn and Lafferty, John and Wasserman, Larry and Li, Xingguo and Zhao, Tuo. *High-Dimensional Undirected Graph Estimation vignette*. 2020. URL: https://cran.r-project.org/web/packages/huge/vignettes/vignette.pdf.

[32] Krämer, Nicole and Schäfer, Juliane. *Regularized estimation of partial correlation matrices*. 2014. URL: https://mran.microsoft.com/snapshot/2017-02-04/web/packages/parcor/parcor.pdf.

[33] Ledoit, Olivier and Wolf, Michael. "A well-conditioned estimator for large-dimensional covariance matrices". In: *Journal of Multivariate Analysis* 88.2 (2004), pp. 365–411. ISSN: 0047-259X. DOI: https://doi.org/10.1016/S0047-259X(03)00096-4. URL: https://www.sciencedirect.com/science/article/pii/S0047259X03000964.

[34] Liu, Han and Roeder, Kathryn and Wasserman, Larry. *Stability Approach to Regularization Selection (StARS) for High Dimensional Graphical Models*. 2010. arXiv: 1006.3316 [stat.ML].

[35] Liu, han and Wang, Lie. "TIGER: A tuning-insensitive approach for optimally estimating Gaussian graphical models". In: *Electronic Journal of Statistics* 11 (2017), pp. 241–294. ISSN: 1935-7524. DOI: 10.1214/16-EJS1195. URL: https://doi.org/10.1214/16-EJS1195.

[36] Lysen, Shaun. "Permuted Inclusion Criterion: A Variable Selection Technique". In: *Publicly accessible Penn Dissertations* (Aug. 2009). URL: https://repository.upenn.edu/cgi/viewcontent.cgi?article=1024&context=edissertations.

[37] Ma, Shisong and Gong, Qingqiu and Bohnert, Hans J. "An Arabidopsis gene network based on the graphical Gaussian model". In: *Genome research* 17.11 (2007), pp. 1614–1625.

[38] Meinshausen, Nicolai and Bühlmann, Peter. "High-dimensional graphs and variable selection with the Lasso". In: *The Annals of Statistics* 34.3 (2006), pp. 1436–1462. DOI: 10.1214/009053606000000281. URL: https://doi.org/10.1214/009053606000000281.

[39] R Core Team. *Package 'parallel'*. R Foundation for Statistical Computing. 2021. URL: http://www.R-project.org/.

[40] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2021. URL: https://www.R-project.org/.

[41] Rajaratnam, Bala and Vincenzi, Dario. "A note on covariance estimation in the unbiased estimator of risk framework". In: *Journal of Statistical Planning and Inference* 175 (2016), pp. 25–39. ISSN: 0378-3758. DOI: https://doi.org/10.1016/j.jspi.2016.02.004. URL: https://www.sciencedirect.com/science/article/pii/S0378375816000252.

[42] Ripley, Brian and Venables, Bill and Bates, Douglas M. and Hornik, Kurt and Gebhardt, Albrecht and Firth, David. *Support Functions and Datasets for Venables and Ripley's MASS*. 2021. URL: https://cran.r-project.org/web/packages/MASS/MASS.pdf.

[43] Rue, Håvard and Held, Leonhard. *Gaussian Markov Random Fields: Theory and Applications*. Boca Raton, FL, US: Chapman Hall/CRC, Taylor Francis Group, 2005, Preface.

[44] Schäfer, Juliane and Opgen-Rhein, Rainer and Zuber, Verena and Ahdesmaki, Miika and Duarte Silva, A. Pedro and Strimmer, Korbinian. *Efficient Estimation of Covariance and (Partial) Correlation*. 2017. URL: https://cran.r-project.org/web/packages/corpcor/corpcor.pdf.

[45] Schäfer, Juliane and Strimmer, Korbinian. "A Shrinkage Approach to Large-Scale Covariance Matrix Estimation and Implications for Functional Genomics". In: *Statistical Applications in Genetics and Molecular Biology* 4 (1 2005). DOI: 10.2202/1544-6115.1175.

[46] Sojoudi, Somayeh. "Equivalence of Graphical Lasso and Thresholding for Sparse Graphs". In: *Journal of Machine Learning Research* 17.115 (2016), pp. 1–21. URL: http://jmlr.org/papers/v17/16-013.html.

[47] Wasserman, Larry. *Undirected Graphical Models*. 2019. URL: https://www.stat.cmu.edu/~larry/=sml/GraphicalModels.pdf.

[48] Witten, Daniela M. and Friedman, Jerome H. and Simon, Noah. "New Insights and Faster Computations for the Graphical Lasso". In: *Journal of Computational and Graphical Statistics* 20.4 (2011), pp. 892–900. DOI: 10.1198/jcgs.2011.11051a. URL: https://doi.org/10.1198/jcgs.2011.11051a.

[49] Xie, Yuying and Liu, Yufeng and Valdar, William. "Joint estimation of multiple dependent Gaussian graphical models with applications to mouse genomics". In: *Biometrika* 103.3 (2016), pp. 493–511. DOI: https://doi.org/10.1093/biomet/asw035. URL: https://academic.oup.com/biomet/article/103/3/493/1744485.

[50] Zhu, Yunan and Cribben, Ivor. "Sparse Graphical Models for Functional Connectivity Networks: Best Methods and the Autocorrelation Issue". In: *Brain Connectivity* 8 (Mar. 2018). DOI: 10.1089/brain.2017.0511.

[51] Zou, Hui. "The Adaptive Lasso and Its Oracle Properties". In: *Journal of the American Statistical Association* 101.476 (2006), pp. 1418–1429. DOI: 10.1198/016214506000000735. eprint: https://doi.org/10.1198/016214506000000735. URL: https://doi.org/10.1198/016214506000000735.

Appendix A

Graphs, Matrices, Mathematical Motivation and Algorithms

A.1 Predefined Matrices with Graphs

Matrix 1

Illustration of Precision Matrix 1

$$
\begin{array}{c}
 \\
Node_1 \\
Node_2 \\
Node_3 \\
Node_4 \\
Node_5
\end{array}
\begin{array}{c}
\begin{array}{ccccc}
Node_1 & Node_2 & Node_3 & Node_4 & Node_5
\end{array} \\
\left(\begin{array}{ccccc}
1 & 0 & 0.07 & 0.01 & 0 \\
0 & 1 & 0.16 & 0.73 & 0.02 \\
0.07 & 0.16 & 1 & 0 & 0 \\
0.01 & 0.73 & 0 & 1 & 0.11 \\
0 & 0.02 & 0 & 0.11 & 1
\end{array}\right)
\end{array}
$$

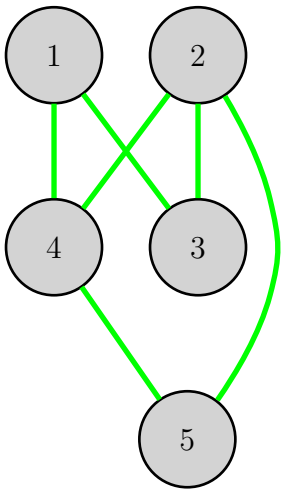


Figure A.1: Graphical Illustration of Precision Matrix 1
where green edges illustrate positive partial correlation, while red edges illustrate negative partial correlation between the nodes.

Matrix 2

Illustration of Precision Matrix 2

	$Node_1$	$Node_2$	$Node_3$	$Node_4$	$Node_5$	$Node_6$	$Node_7$	$Node_8$	$Node_9$	$Node_{10}$	$Node_{11}$	$Node_{12}$	$Node_{13}$	$Node_{14}$	$Node_{15}$
$Node_1$	1	0.83	0	0	0	0	0	0	0	0	0	0	0	0	0
$Node_2$	0.83	1	0	0	0	0	0	0	0	0	0	0	0	0	0
$Node_3$	0	0	1	0	0	0	0	0	0	0	0.84	0	0	0	0
$Node_4$	0	0	0	1	0	−0.31	0.03	0	0	0	0	0	0	−0.48	0
$Node_5$	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
$Node_6$	0	0	0	−0.31	0	1	0	−0.09	0	0	0	0	0	0	0
$Node_7$	0	0	0	0.03	0	0	1	0	0	0	0	0	0.03	0	0
$Node_8$	0	0	0	0	0	0	0	1	0	−0.40	0	0	0	0.52	0
$Node_9$	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
$Node_{10}$	0	0	0	0	0	0	0	−0.40	0	1	0	0	0	0	0
$Node_{11}$	0	0	0.84	0	0	0	0	0	0	0	1	0	0	0	0
$Node_{12}$	0	0	0	0	0	0	0	0	0	0	0	1	−0.76	0	0
$Node_{13}$	0	0	0	0	0	0	0.03	0	0	0	0	−0.76	1	0	0
$Node_{14}$	0	0	0	−0.48	0	0	0	0.52	0	0	0	0	0	1	0
$Node_{15}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

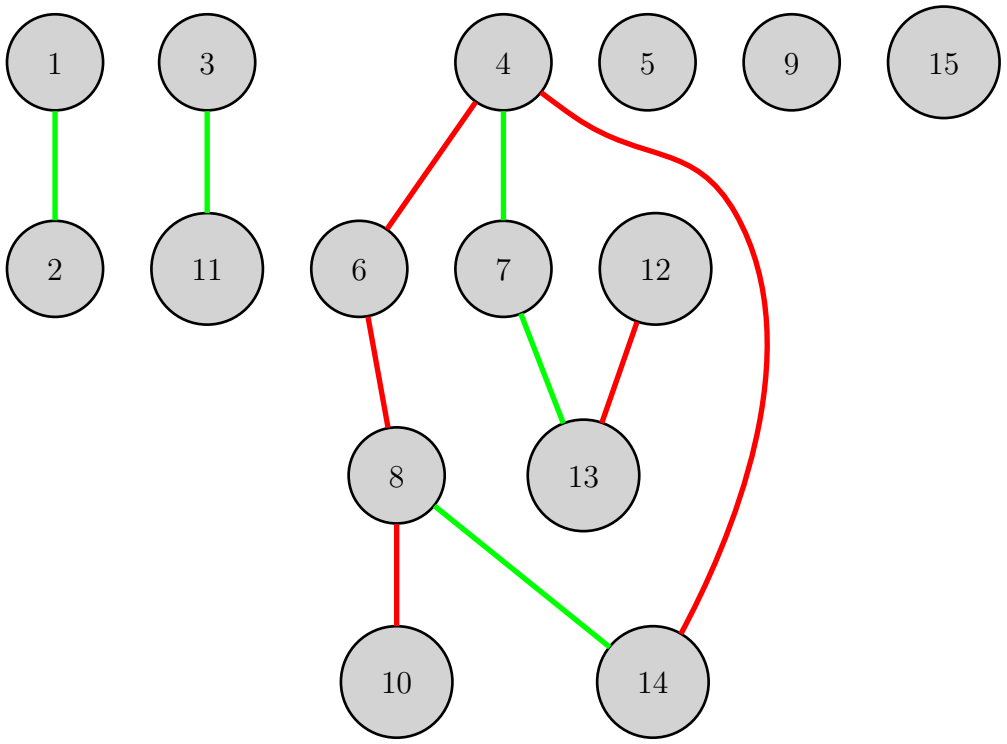


Figure A.2: Graphical Illustration of Precision Matrix 2
where green edges illustrate positive partial correlation, while red edges illustrate negative partial correlation between the nodes.

Matrix 3

Illustration of Precision Matrix 3

Part 1 of 2

	$Node_1$	$Node_2$	$Node_3$	$Node_4$	$Node_5$	$Node_6$	$Node_7$	$Node_8$	$Node_9$	$Node_{10}$	$Node_{11}$	$Node_{12}$
$Node_1$	1	0	0	0	0	0	0	0	0	0	0	0
$Node_2$	0	1	0	0	0	0	0	0	0	0	0	0
$Node_3$	0	0	1	0	0	0	0	0	0	0	0	0
$Node_4$	0	0	0	1	0	0	0.03	0	0	0	0	0
$Node_5$	0	0	0	0	1	0	0	0	0	0	−0.61	0
$Node_6$	0	0	0	0	0	1	0	0	0	0	0	0
$Node_7$	0	0	0	0	0	0	1	0	0	0	0	0
$Node_8$	0	0	0	0	0	0	0	1	0	0	0	0
$Node_9$	0	0	0	0	0	0	0	0	1	0	0	0
$Node_{10}$	0	0	0	0	0	0	0	0	0	1	0	0
$Node_{11}$	0	0	0	0	−0.61	0	0	0	0	0	1	0
$Node_{12}$	0	0	0	0	0	0	0	0	0	0	0	1
$Node_{13}$	0	0	0	0	0	0	0	0	0	0	0	0
$Node_{14}$	0	0	0	0	0	0	−0.003	0	0	0	0	0
$Node_{15}$	0	0	0	0	0	0	0	0	0	0	0	0
$Node_{16}$	0	0	0	0	0	−0.45	0	0	0	0	0.50	0
$Node_{17}$	0	0	0	0	0	0	0	0	−0.27	0	0	0
$Node_{18}$	0	−0.33	0	0	0	0	0	0	0	0	0	0
$Node_{19}$	0	0	0	0	0	0	0	0	0	0	0	0
$Node_{20}$	0	0	0	0	0	0	0	0	0	0	0	0
$Node_{21}$	0	0.07	0	0	0	0	0	0	0	0	0	0
$Node_{22}$	0	0	0	0	0	0	0	0	0	0	0	0
$Node_{23}$	0	0	0	0	0	0	0	0	0	−0.14	0	0
$Node_{24}$	0	0	0	0	0	0	0	0	0	0	0	0
$Node_{25}$	0	0	0	0	0	0	0	0	0	0	0	0

Part 2 of 2

	$Node_{13}$	$Node_{14}$	$Node_{15}$	$Node_{16}$	$Node_{17}$	$Node_{18}$	$Node_{19}$	$Node_{20}$	$Node_{21}$	$Node_{22}$	$Node_{23}$	$Node_{24}$	$Node_{25}$
$Node_1$	0	0	0	0	0	0	0	0	0	0	0	0	0
$Node_2$	0	0	0	0	0	−0.33	0	0	0.07	0	0	0	0
$Node_3$	0	0	0	0	0	0	0	0	0	0	0	0	0
$Node_4$	0	0	0	0	0	−0.31	0.03	0	0	0	0	0	0
$Node_5$	0	0	0	0	0	0	0	0	0	0	0	0	0
$Node_6$	0	0	0	−0.45	0	0	0	0	0	0	0	0	0
$Node_7$	0	−0.003	0	0	0	0	0	0	0	0	0	0	0
$Node_8$	0	0	0	0	0	0	0	0	0	0	0	0	0
$Node_9$	0	0	0	0	−0.27	0	0	0	0	0	0	0	0
$Node_{10}$	0	0	0	0	0	0	0	0	0	0	−0.14	0	0
$Node_{11}$	0	0	0	0.50	0	0	0	0	0	0	0	0	0
$Node_{12}$	0	0	0	0	0	0	0	0	0	0	0	0	0
$Node_{13}$	1	0	0	0	0	0	0	0	0	−0.14	0	0	0
$Node_{14}$	0	1	0	0	0	0	0	0	0	0	−0.88	0	0
$Node_{15}$	0	0	1	0	0	0	0	0	0	0	0	0	0
$Node_{16}$	0	0	0	1	0	0	0	0	0.16	0	0	0	0
$Node_{17}$	0	0	0	0	1	0	0.37	0	0	0	0	0	0
$Node_{18}$	0	0	0	0	0	1	0	−0.63	0	0	0	0	0
$Node_{19}$	0	0	0	0	0.37	0	1	−0.59	0	0	0	0	0
$Node_{20}$	0	0	0	0	0	−0.63	−0.59	1	0	0	0	0.24	0
$Node_{21}$	0	0	0	0.16	0	0	0	0	1	0	0	0	0
$Node_{22}$	−0.14	0	0	0	0	0	0	0	0	1	0	0	0
$Node_{23}$	0	−0.88	0	0	0	0	0	0	0	0	1	0	0
$Node_{24}$	0	0	0	0	0	0	0	0.24	0	0	0	1	0
$Node_{25}$	0	0	0	0	0	0	0	0	0	0	0	0	1

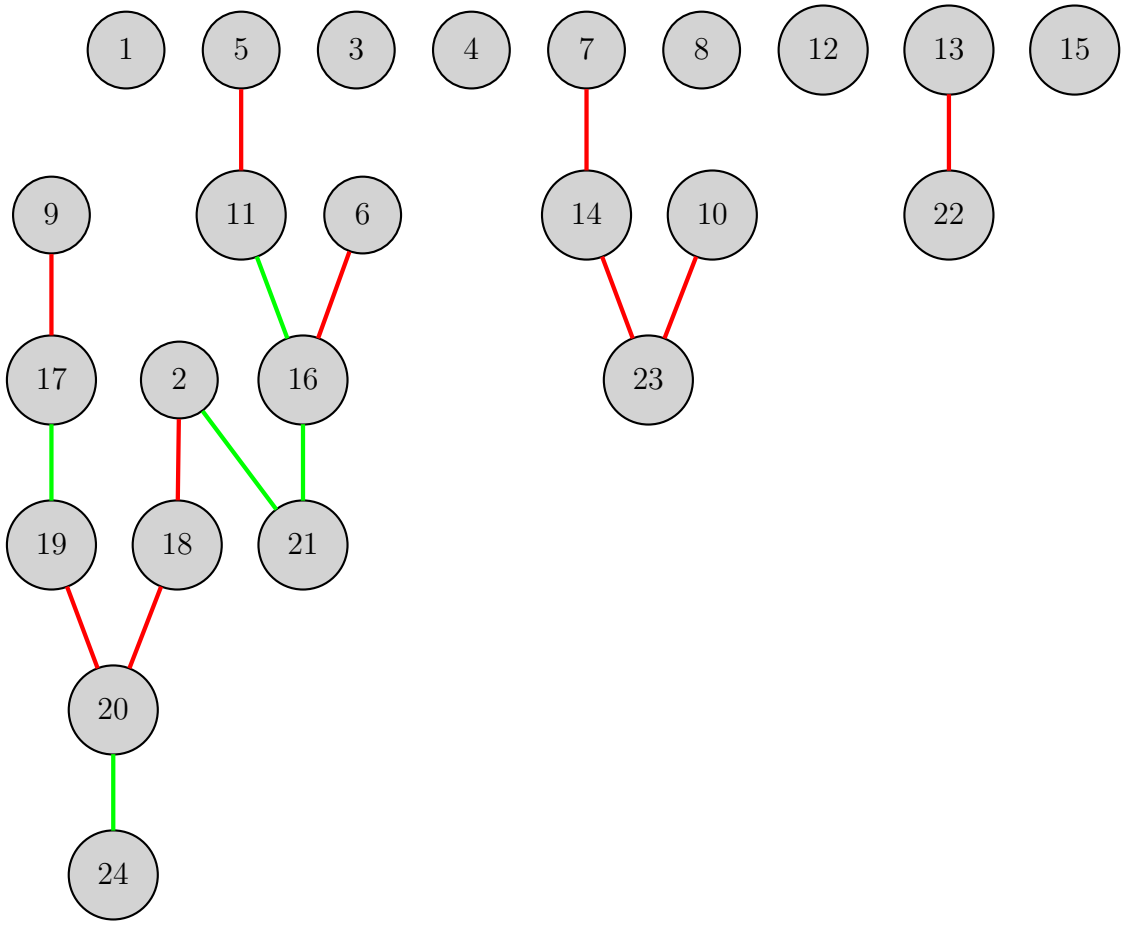


Figure A.3: Graphical Illustration of Precision Matrix 3
where green edges illustrate positive partial correlation, while red edges illustrate negative partial correlation between the nodes.

Matrix 4

Illustration of Precision Matrix 4

part 1 out of 8

	N_1	N_2	N_3	N_4	N_5	N_6	N_7	N_8	N_9	N_{10}	N_{11}	N_{12}	N_{13}	N_{14}	N_{15}	N_{16}	N_{17}	N_{18}	N_{19}	N_{20}	N_{21}	N_{22}	N_{23}	N_{24}	N_{25}
$Node_1$	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$Node_2$	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$Node_3$	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$Node_4$	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$Node_5$	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$Node_6$	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$Node_7$	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$Node_8$	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0.75	0	0	0	0
$Node_9$	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$Node_{10}$	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$Node_{11}$	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$Node_{12}$	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
$Node_{13}$	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
$Node_{14}$	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
$Node_{15}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
$Node_{16}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
$Node_{17}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
$Node_{18}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
$Node_{19}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
$Node_{20}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
$Node_{21}$	0	0	0	0	0	0	0	0.75	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
$Node_{22}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
$Node_{23}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
$Node_{24}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
$Node_{25}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
$Node_{26}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$Node_{27}$	−0.006	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$Node_{28}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$Node_{29}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$Node_{30}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$Node_{31}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$Node_{32}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$Node_{33}$	0	0	0	0	0.08	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$Node_{34}$	0	0	0	0	0	0	0	0	0	0.23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$Node_{35}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$Node_{36}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$Node_{37}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$Node_{38}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$Node_{39}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$Node_{40}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$Node_{41}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$Node_{42}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$Node_{43}$	0	0	−0.94	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$Node_{44}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$Node_{45}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$Node_{46}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	−0.33	0	0	0	0
$Node_{47}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$Node_{48}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$Node_{49}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$Node_{50}$	0	0	0	0	0	−0.19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

NOTE: Due to the size of the Matrix, the complete matrix will not be provided here. Readers are referred to contact the authors by e-mail to access the full generated matrix.

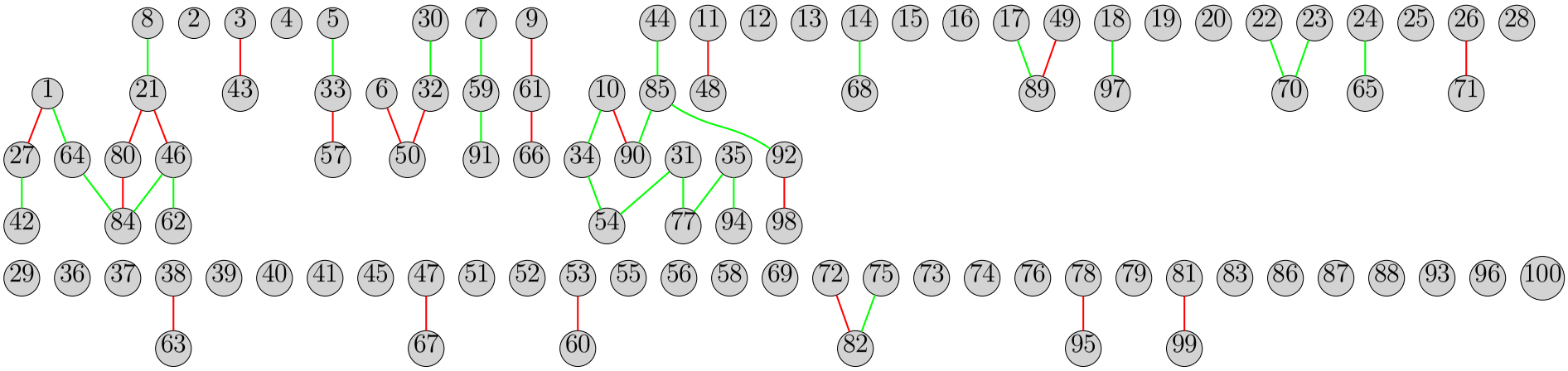


Figure A.4: Graphical Illustration of Precision Matrix 4 where green edges illustrate positive partial correlation, while red edges illustrate negative partial correlation between the nodes.

A.2 Mathematical Derivation

Let us assume the sampled multivariate data $\vec{X}_i$ be normally distributed and their mean and covariance are denoted as $\vec{\mu}$ and Σ, i.e.

$$\vec{X}_i = \begin{pmatrix} x_1 \\ \vdots \\ x_p \end{pmatrix}_i \qquad E[\vec{X}_i] = \vec{\mu} = \begin{pmatrix} \mu_1 \\ \vdots \\ \mu_p \end{pmatrix} \qquad V[X_i] = E(\vec{X}_i - \vec{\mu})(\vec{X}_i - \vec{\mu})^T = \Sigma.$$

Under such assumption, their density function equals to

$$f\big(\vec{X}\big) = \frac{1}{(2\pi)^{\frac{p}{2}}(\det\Sigma)^{\frac{1}{2}}} \cdot \exp\left[-\frac{1}{2}\big(\vec{X} - \vec{\mu}\big)^T \Sigma^{-1} \big(\vec{X} - \vec{\mu}\big) \right] \tag{A.1}$$

If the sample data constitute independent observations, $\vec{X}_1, \ldots, \vec{X}_n$, then their likelihood function can be defined as a product of the density functions evaluated on the collected samples:

$$\mathcal{L}\big(\{\vec{X}_1, \ldots \vec{X}_n\}, \vec{\mu}, \Sigma\big) := f(\vec{X}_1) \cdot f(\vec{X}_2) \cdot \ \ldots \ \cdot f(\vec{X}_n) \tag{A.2}$$

The straightforward calculations show then that the log-likelihood function is:

$$\begin{aligned} l(\cdot) \ &= \ \underbrace{\ln\left[\frac{1}{(2\pi)^{p/2}}\right] + \ln\left[\frac{1}{(\det\Sigma)^{1/2}}\right] + \left[-\frac{1}{2}(\vec{X}_1 - \vec{\mu})^T\Sigma^{-1}(\vec{X}_1 - \vec{\mu})\right]}_{=\ln f(X_1)} + \ldots \\ &\qquad + \underbrace{\ln\left[\frac{1}{(2\pi)^{p/2}}\right] + \ln\left[\frac{1}{(\det\Sigma)^{1/2}}\right] + \left[-\frac{1}{2}(\vec{X}_n - \vec{\mu})^T\Sigma^{-1}(\vec{X}_n - \vec{\mu})\right]}_{=\ln f(X_n)} \\ &= \ \underbrace{n\ln\left(\frac{1}{(2\pi)^{p/2}}\right)}_{=-\frac{k}{2}\ln 2\pi} + n\ln\left(\frac{1}{(\det\Sigma)^{1/2}}\right) - \frac{1}{2}\sum_{i=1}^{n}\left((\vec{X}_n - \vec{\mu})^T\Sigma^{-1}(\vec{X}_n - \vec{\mu})\right) \end{aligned} \tag{A.3}$$

If one denotes the inverse of Σ in Eqn. (A.3) as Ω and take into account the relations

$$\Omega\,\Sigma = I_p \quad \Rightarrow \quad \det\Omega \cdot \det\Sigma = 1 \quad \Rightarrow \quad \det\Omega = \frac{1}{\det\Sigma},$$

then the log-likelihood function will be of the form

$$l(\cdot) = -\frac{np}{2}\ln 2\pi + \frac{n}{2}\ln\big(\det\Omega\big) - \frac{1}{2}\sum_{i=1}^{n}\left((\vec{X}_n - \vec{\mu})^T\Omega(\vec{X}_n - \vec{\mu})\right) \tag{A.4}$$

Applied example:
For illustration purposes of the optimization arguments, let us consider the case $p = 2$ and thus

$$\vec{X} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}; \;\; \vec{\mu} = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}; \;\; \Omega = \begin{pmatrix} u_1 & u_{12} \\ u_{12} & u_2 \end{pmatrix}.$$

For the case (as for the general one) the optimization problem is to find the unknown values for the vector $\vec{\mu}$ and the positive definite matrix Ω that maximize the value of the function (A.4). Such optimization can be done in two steps:

Step 1: Assume that the matrix parameter Ω is known and and is set to a fixed value. Find $\vec{\mu}^*$ that maximizes $l(\cdot)$.

Step 2: Substitute $\vec{\mu} = \vec{\mu}^*$ in the function $l(\cdot)$ and search separately for a positive definite matrix Ω that maximizes $l(\cdot)$.

To complete **Step 1**, let us find the maximizer (A.4) for given , i.e.

$$\begin{aligned}
\vec{\mu}^* &= \underset{\mu}{\operatorname{argmax}}\, l(\cdot) = \underset{\mu}{\operatorname{argmax}} \left\{ -\frac{1}{2} \sum_{i=1}^{n} \left((\vec{X}_n - \vec{\mu})^T \Omega (\vec{X}_n - \vec{\mu}) \right) \right\} \\
&= \underset{\mu}{\operatorname{argmax}} \left\{ -\frac{1}{2} \sum_{i=1}^{n} \left(\begin{pmatrix} x_{1i} - \mu_1 \\ x_{2i} - \mu_2 \end{pmatrix}^T \begin{pmatrix} u_1 & u_{12} \\ u_{12} & u_2 \end{pmatrix} \begin{pmatrix} x_{1i} - \mu_1 \\ x_{2i} - \mu_2 \end{pmatrix} \right) \right\} \\
&= \underset{\mu}{\operatorname{argmax}} \left\{ -\frac{1}{2} \sum_{i=1}^{n} \left(u_1 (x_{1i} - \mu_1)^2 + 2u_{12}(x_{1i} - \mu_1)(x_{2i} - \mu_2) + u_2 (x_{2i} - \mu_2)^2 \right) \right\} \qquad \text{(A.5)}
\end{aligned}$$

Take partial derivatives of the function (A.5) with respect to each parameter μ_1, μ_2 and equating each of then with zero, we obtain the necessary conditions of optimality for the function $l(\cdot)$.

$$\frac{\partial l(\cdot)}{\partial \mu_1} = 0, \qquad \frac{\partial l(\cdot)}{\partial \mu_2} = 0 \qquad \text{(A.6)}$$

They can be reformulated as follows

$$\begin{aligned} 0 &= \frac{\partial l(\cdot)}{\partial \mu_1} = \frac{\partial}{\partial \mu_1}\Big(-\frac{1}{2}\sum_{i=1}^{n}\Big(u_1(x_{1i}-\mu_1)^2 + 2u_{12}(x_{1i}-\mu_1)(x_{2i}-\mu_2) + u_2(x_{2i}-\mu_2)^2\Big) \\ &= -\frac{1}{2}\sum_{i=1}^{n}\Big(2u_1(x_{1i}-\mu_1)\cdot(-1) + 2u_{12}\cdot(-1)(x_{2i}-\mu_2)\Big) \\ &= \sum_{i=1}^{n}\Big(u_1(x_{1i}-\mu_1) + u_{12}(x_{2i}-\mu_2)\Big) \\ &= u_1\big(\sum_{i=1}^{n}x_{1i}\big) - u_1\cdot n\cdot\mu_1 + u_{12}\big(\sum_{i=1}^{n}x_{2i}\big) - u_{12}\cdot n\cdot\mu_2 \end{aligned}$$

Dividing the last expression by n, we have

$$0 = u_1\left(\frac{1}{n}\sum_{i=1}^{n}x_{1i}\right) - u_1\cdot\mu_1 + u_{12}\left(\frac{1}{n}\sum_{i=1}^{n}x_{2i}\right) - u_{12}\cdot\mu_2 = u_1\big(\bar{x}_1-\mu_1\big) + u_{12}\big(\bar{x}_2-\mu_2\big) \tag{A.7}$$

Taking advantage of similar arguments in analyzing the second necessary optimality condition (A.6) for the function $l(\cdot)$, we can find that:

$$u_{12}(\bar{x}_1-\mu_1) + u_2(\bar{x}_2-\mu_2) = 0 \tag{A.8}$$

Combining both equations (A.7)-(A.8) and rewriting them in a matrix form results in the matrix equation

$$\underbrace{\begin{bmatrix} u_1 & u_{12} \\ u_{12} & u_2 \end{bmatrix}}_{\Omega}\begin{bmatrix} \bar{x}_1-\mu_1 \\ \bar{x}_2-\mu_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{A.9}$$

Since by assumption the matrix Ω is positive definite, then this matrix has well defined inverse. This immediately implies that the only candidate solution for the partial optimization task mentioned in **Step 1** is:

$$\mu_1{}^* = \bar{x}_1 \qquad \mu_2{}^* = \bar{x}_2.$$

To certify that this is indeed the optimizer, we need to check the Hessian matrix of $l(\cdot)$ at this point is negative definite. The direct calculations show that this is the case

$$\frac{\partial^2 l(\cdot)}{\partial \mu^2)} = \begin{bmatrix} \frac{\partial^2 l(\cdot)}{\partial {\mu_1}^2} & \frac{\partial^2 l(\cdot)}{\partial \mu_1 \partial \mu_2} \\ \frac{\partial^2 l(\cdot)}{\partial \mu_1 \partial \mu_2} & \frac{\partial^2 l(\cdot)}{\partial {\mu_2}^2} \end{bmatrix} = -\Omega < 0 \tag{A.10}$$

Utilizing the found maximum solutions, we proceed with **Step 2** in order to find Ω that maximizes $l(\cdot)$. We once again begin our search with equation A.4.

$$l(\{\vec{X}_1, \dots \vec{X}_n\}, \vec{\mu} = \vec{\mu}^*, \Omega) = \frac{-np}{2} \ln 2\pi + \frac{n}{2} \ln(\det \Omega) - \frac{1}{2} \sum_{i=1}^{n} \left((\vec{X}_i - \bar{X})^T \Omega (\vec{X}_i - \bar{X}) \right) \tag{A.11}$$

The last part of equation A.11, could be rewritten as:

$$-\frac{1}{2} \sum_{i=1}^{n} \left((\vec{X}_i - \bar{X})^T \Omega (\vec{X}_i - \bar{X}) \right) = \text{trace} \left[-\frac{1}{2} \sum_{i=1}^{n} \left((\vec{X}_i - \bar{X})^T \Omega (\vec{X}_i - \bar{X}) \right) \right] \tag{A.12}$$

$$= -\frac{1}{2} \sum_{i=1}^{n} \text{trace} \left[\underbrace{(\vec{X}_i - \bar{X})^T}_{A} \underbrace{\Omega (\vec{X}_i - \bar{X})}_{B} \right] \tag{A.13}$$

Quick Note: $trace(AB) = trace(BA)$.

$$= -\frac{1}{2} \sum_{i=1}^{n} \text{trace} \left[\underbrace{\Omega (\vec{X}_i - \bar{X})}_{B} \underbrace{(\vec{X}_i - \bar{X})^T}_{A} \right] \tag{A.14}$$

$$= -\frac{1}{2} \text{trace} \left[\Omega \sum_{i=1}^{n} (\vec{X}_i - \bar{X})(\vec{X}_i - \bar{X})^T \right] \tag{A.15}$$

Extend equation (A.15) by multiplying and dividing the expression on n,

$$= -\frac{n}{2} \text{trace} \left[\Omega \underbrace{\frac{\sum_{i=1}^{n} (\vec{X}_i - \bar{X})(\vec{X}_i - \bar{X})^T}{n}}_{Sample\, Covariance\ \hat{\Sigma}} \right] \tag{A.16}$$

Thus yielding literally the same result as in equation (3.9):

$$l\left(\left\{ \vec{X}_1, \dots \vec{X}_n \right\}, \vec{\mu} = \vec{\mu}^*, \Omega \right) = \frac{n}{2} \ln(\det \Omega) - \frac{n}{2} \text{trace}(\Omega \hat{\Sigma}) - \frac{np}{2} \ln 2\pi \tag{A.17}$$

A.3 Confusion Matrix Algorithms

Algorithm 2: Benchmark Matrix Algorithm

input : Estimated Precision Matrix, Iterations
output: Benchmark Matrix

```
1  begin
2      Original Matrix ⟵ abs(Original Matrix);
3      diag(Original Matrix) ⟵ ∅;
4      foreach Iteration do
5          Generate a zero matrix M;
6          forall Nonzero Elements in Original Matrix do
7              Input value 1 in M
8          Benchmark Matrix ⟵ M;          // Each iteration adds the
            original matrix edges into a new matrix, creating the
            true optimal precision matrix that the undirected
            Gaussian graphical model estimate strive towards
9          return(Benchmark Matrix)
10     return(Benchmark Matrix)
```

Algorithm 3: True Positive Algorithm

input : Benchmark Matrix, Simulation Results
output: Number of True Positive values in Estimated Model

```
1  begin
2      True Positive Matrix ⟵ ∅; M ⟵ Benchmark Matrix;
3      R ⟵ Simulation Results;
4      for i in 1:length(M) do
5          if M_i = 0 then
6              True Positive Matrix element i = 0
7          else
8              if R_i ≥ M_i then
9                  True Positive Matrix element i = M_i
10             else
11                 True Positive Matrix element i = R_i
12         return(True Positive Matrix);
13     for i in 1:(True Positive Matrix) do
14         sum all elements into numeric variable → TP;
15     return(TP/2);          // Since only upper triangle is considered
```

Algorithm 4: True Negative Algorithm

input : Benchmark Matrix, Simulation Results, Number of Iterations
output: Number of True Negative values in Estimated Model

```
1  begin
2      True Negative Matrix ⟵ ∅; M ⟵ Benchmark Matrix;
3      R ⟵ Simulation Results;
4      for i in 1:length(M) do
5          if M_i = 0 and R_i = 0 then
6              True Negative Matrix element i = Number of Iterations
7          else
8              if M_i = 0 then
9                  True Negative Matrix element i = [Number of Iterations − R_i]
10             else
11                 True Negative Matrix element i = 0
12         return(True Negative Matrix);
13     for i in 1:(True Negative Matrix) do
14         sum all elements into numeric variable → TN;
15     diag(TN) = 0;
16     return(TN/2);          // Since only upper triangle is considered
```

Algorithm 5: False Positive Algorithm

input : Benchmark Matrix, Simulation Results
output: Number of False Positive values in Estimated Model

```
1  begin
2      False Positive Matrix ⟵ ∅; M ⟵ Benchmark Matrix;
3      R ⟵ Simulation Results;
4      for i in 1:length(M) do
5          if R_i = 0 then
6              False Positive Matrix element i = 0
7          else
8              if M_i = 0 then
9                  False Positive Matrix element i = R_i
10             else
11                 False Positive Matrix element i = 0
12         return(False Positive Matrix);
13     for i in 1:(False Positive Matrix) do
14         sum all elements into numeric variable → FP;
15     return(FP/2);          // Since only upper triangle is considered
```

Algorithm 6: False Negative Algorithm

input : Benchmark Matrix, Simulation Results
output: False Negative values in Estimated Model

```
1  begin
2      False Negative Matrix ⟵ ∅; M ⟵ Benchmark Matrix;
3      R ⟵ Simulation Results;
4      for i in 1:length(M) do
5          if M_i = 0 and R_i = 0 then
6              False Negative Matrix element i = 0
7          else
8              if M_i ≠ 0 then
9                  False Negative Matrix element i = [M_i - R_i]
10             else
11                 False Negative Matrix element i = 0
12         return(False Negative Matrix);
13     for i in 1:(False Negative Matrix) do
14         sum all elements into numeric variable → FN;
15     return(FN/2);          // Since only upper triangle is considered
```