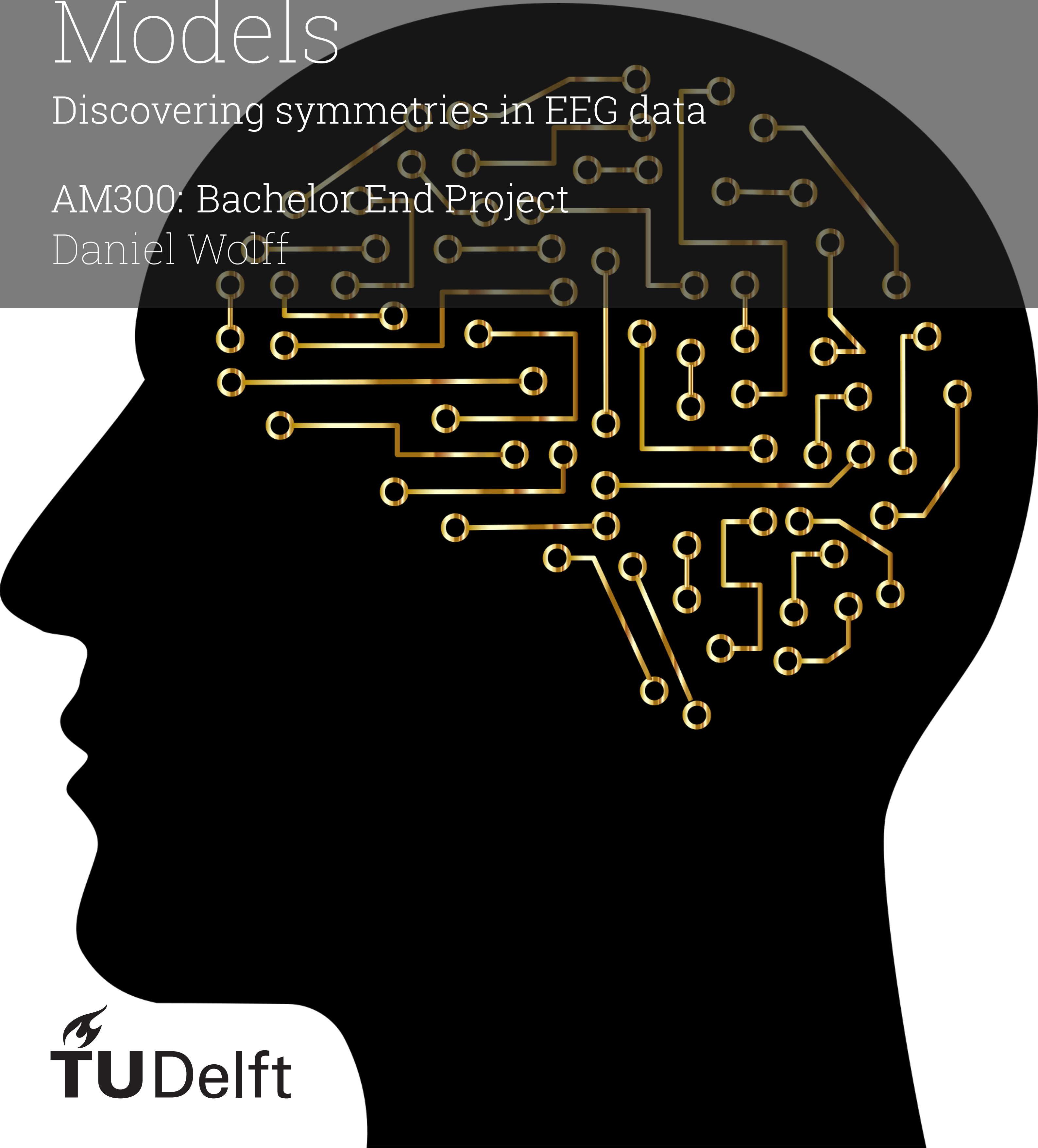
Symmetries in Graphical Gaussian Models

Discovering symmetries in EEG data

AM300: Bachelor End Project

Daniel Wolff



TU Delft

Symmetries in Graphical Gaussian Models

Discovering symmetries in EEG data

by

Daniel Wolff

to obtain the degree of Bachelor of Science

at the Delft University of Technology,

to be defended publicly on Monday July 25, 2024 at 12:00.

Project duration:	April 22, 2024 – July 25, 2024	
Thesis committee:	Dr. D. (Dorota) Kurowicka,	TU Delft, supervisor
	Dr. N. (Nestor) Parolya,	TU Delft

Cover:	Pixabay
Style:	TU Delft Report Style, with modifications by Daan Zwaneveld

TUDelft

Summary

Laymen summary: This thesis investigates the potential of enhancing our understanding of brain activity through the analysis of electroencephalography (EEG) data using Gaussian Graphical Models (GGMs). EEG data, which contains the electrical activity of the brain, is often complex and challenging to interpret. The main idea is to use a graphical representation and discover symmetry patterns in the EEG data to make the relationships between brain regions more clear. By detecting symmetries, we can reduce the complexity of the data. This approach was validated through simulations and EEG data analysis, showing that it helps reveal important brain activity patterns. The findings could improve how we analyze brain signals, helping medical research and potentially leading to better diagnostic tools.

Summary: This thesis explores the application of Gaussian Graphical Models (GGMs) with a specific focus on identifying symmetries within EEG data. A significant challenge in using GGMs is achieving accurate estimations with limited observations, which is common in medical data. This research proposes a methodology for detecting and integrating symmetric structures in GGMs, thereby reducing the number of parameters and improving model interpretability. We follow developments presented by Højsgaard en Lauritzen (2008)[1]. The study includes a detailed explanation of the multivariate Gaussian distribution, Maximum Likelihood Estimation (MLE), and the implementation of Iterative Partial Maximization for parameter estimation. Additionally, hierarchical clustering is employed to systematically identify symmetry classes. Results indicate that incorporating symmetry constraints enhances the accuracy and interpretability of GGMs. The research shows that symmetry constraints simplify models, making them more robust and easier to understand. A simulation study was conducted to test the efficiency and accuracy of the developed algorithm for symmetry detection. The findings from these simulations validate the proposed approach, demonstrating significant improvements in model performance. Finally, the methodology was applied to an EEG dataset, highlighting practical applications in neuroscience. The results from the EEG data analysis further confirm that symmetry constraints can reveal underlying patterns in brain connectivity, offering valuable insights into the neural dynamics. This thesis contributes to the existing literature by providing a systematic approach to detect symmetries in high-dimensional data models, particularly its practical utility with real-world EEG data.

Contents

Nomenclature

Abbreviations

Abbreviation	Definition
GGM	Graphical Gaussian Model
IPM	Iterative Partial Maximization
HC	Hierarchical Clustering

Symbols

Symbol	Definition
G	Graph
V	Vertex class
E	Edge class
$\mathcal{V}$	Coloured Vertex Class
$\mathcal{E}$	Coloured Edge Class
σ	Standard deviation of a one dimensional normal distribution
Σ	Covariance matrix of a multidimensional normal distribution
K	Inverse Covariance matrix, Precision matrix
$\mathcal{K}$	Rewritten Precision matrix for interpretability
μ	Either a value or vector with the mean values of a normal distribution

1 Introduction

High-dimensional datasets have grown increasingly common across a wide range of industries, as the ability to collect and store data grows. However, analysis of such high-dimensional datasets poses major challenges [2]. Where patterns and relationships were more easily found in small datasets, it is hard to keep track of them in large datasets. It becomes difficult to generalize findings in data and more often than not the number of observations will not be sufficient to create accurate estimations of how these variables interact [3]. Luckily, there are techniques that allow data to be organized in an intuitive and easy to interpret manner. In this thesis a technique called Gaussian Graphical Models (GGMs) is presented.

Graphical models have played an increasingly vital role in the interpretation of multivariate data with the recent advancements in multivariate data analysis [4]. GGMs are specifically useful due to their ability to represent complex dependencies between variables in an interpretable way. This allows the study of important problems not only in theoretical statistics but also in practical applications like genomics, network analysis, and machine learning, where the structure and relationships needs to be analyzed using high-dimensional data sets.

A GGM is a statistical model in which a set of variables is assumed to be normally distributed with direct dependencies between variables included as edges of a graph. The nodes of the graph represent these normal variables, and the absence of an edge between any two nodes indicates their conditional independence given the remaining variables. For the Gaussian distribution all conditional independencies can be read from the precision matrix, which is the inverse of the covariance matrix.

In this thesis we will discuss how GGMs can be implemented to simplify the vast structure of brain signals captured by electroencephalograms (EEG's). The common way to measure these brain signals is placing electrodes on the head of a patient in the regions depicted by the international standard 10-20 system shown in Figure 1.1. In the left figure the locations of the electrodes can be seen together with the names of the brain regions.

EEG signals measure electrical activity in the brain and are able to show the complex patterns that reflect the underlying neuronal dynamics and interactions in the brain. Understanding these patterns is important and can lead to better knowledge of brain functioning. Signals are recorded from multiple electrodes placed at different spots on the scalp. The connectivity and symmetries could be present in the collected data due to synchronized activity between brain regions. Thus modeling these symmetries can reveal interesting information about the workings of our brains [5].

Exploring symmetries in EEG signals and modeling them with GGMs offers a systematic approach to discovering the neural dynamics of the brain. By leveraging these symmetries, researchers can gain deeper insights into brain connectivity.

The research on GGMs over the last decade has focused on parameter estimation, in particular the estimation of the precision matrix, corresponding directly to the edge structure of the underlying graph (Graphical Models in R, 2012) [6]. The recent extension of GGMs (Højsgaard & Lauritzen, 2008) [1] that include symmetries into this model is the main topic of this thesis. Various types of symmetries can

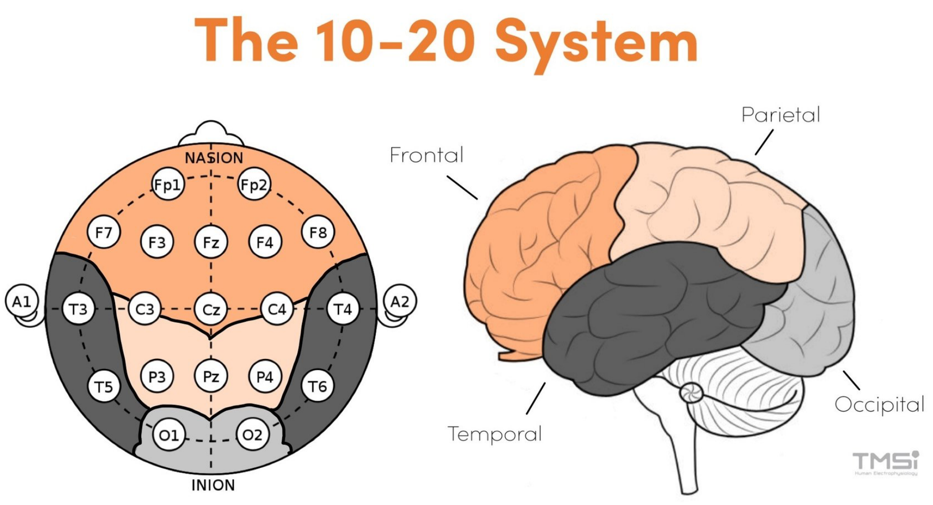


Figure 1.1: The 10/20 system used for EEG readings.

be considered, including edge symmetries (where pairs of variables share similar strengths of relationships) and vertex symmetries (where variables have interchangeable roles). An R software has been developed for handling the computations based on these symmetrical restrictions, known as gRc [7]. However, not too much attention has been given to how the symmetries can be found from the available data. Obviously all possible models with different classes can be estimated. This approach, however, is computationally too expensive. In this thesis we propose to advance the class selection problem. The main goals of the thesis are:

1. Introduce GGM with and without symmetries
2. Discuss estimation of parameters of GGMs.
3. Identify symmetry classes.
4. Evaluate the introduced algorithms via simulation study.
5. Apply GGMs to EEG data.

The outlook of this thesis is:

Initially, we will discuss the maximum likelihood estimation for the multivariate normal distribution in Chapter 2. To asses whether a graph could contain fewer edges the likelihood ratio test will be used in Chapter 2. This technique will also be useful to check whether one could impose symmetry constraints and will be presented in Chapter 3. Next, we will implement hierarchical clustering to systematically detect these symmetries in Chapter 3. In Chapter 4, the results of a simulation study to test the efficiency and accuracy of the algorithm developed for symmetry detection are presented. Finally, we will test and analyze a dataset consisting of EEG data in Chapter 5. This will then end with conclusions and discussions in Chapter 6.

2 Gaussian Graphical Models

This chapter introduces Gaussian Graphical Models (GGMs), a powerful tool for understanding the relationships between multiple variables. We begin by discussing graphs, which form the qualitative part of the model and graphically represent the relationships between variables.

The quantitative part of GGMs is the Gaussian distribution. We will use Maximum Likelihood Estimation (MLE) to estimate the parameters of the Gaussian distribution in GGMs from data.

2.1. Graphs

Consider a graph $G = (V, E)$ where $V = \{v_1, v_2, ..., v_k\}$ is a set of vertices or nodes, and $E \subset V \times V$ represents the undirected edges connecting these vertices. Hence if $(i, j) \in E$, then $(j, i) \in E$ as well. Loops and multiple edges are not allowed ($(i, i) \notin E$ for all $i = 1, .., k$).

Figure 2.1 illustrates an undirected graph with six nodes and eight edges. Additionally, it shows one loop (in green) and one multiple edge (in red), which are not permitted.

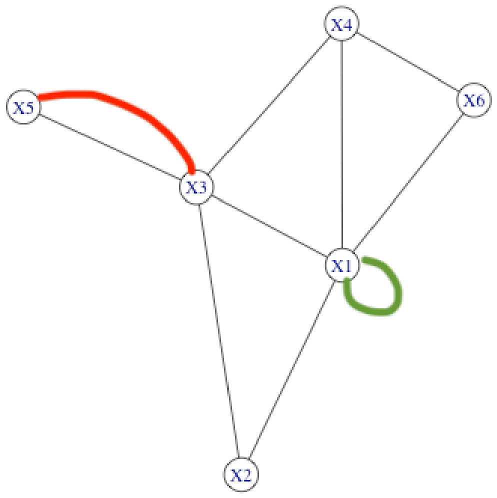


Figure 2.1: A graph with 6 vertices and 8 edges

In the next section we show how the undirected graphs are used to represent direct relationships between random variables in GGMs.

2.2. Gaussian Graphical Models

A Graphical Model is a statistical framework that represents the structure of a set of variables and their probabilistic relationships using a graph. In a graphical model, nodes represent variables, and edges represent direct relationships between these variables. Through the concept of separation (Whittaker, 1990)[8], we can read all conditional independencies represented by the graph. In GGMs, the underlying distribution of the random variables, with the conditional independencies represented by the graph, is the Gaussian distribution. This distribution is convenient due to its parameterization by the mean vector and the covariance matrix. Conditional independencies can be read from the precision matrix, which is the inverse of the covariance matrix.

2.2.1. The Multivariate Gaussian Distribution

This thesis focuses on the Gaussian distribution, often referred to as the Normal distribution, with an emphasis on the multivariate Gaussian distribution.[1]

Consider k random variables, forming a vector $X = (X_1, X_2, \dots, X_k) \sim \mathcal{N}_k(\mu, \Sigma)$, where μ is a k-dimensional vector and Σ is a $k \times k$ covariance matrix, which is symmetric and positive definite.

The probability density function for the multivariate Gaussian distribution is given by:

$$f_X(x; \mu; \Sigma) = \frac{1}{(2\pi)^{k/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right) \tag{2.1}$$

where $x \in R^k$.

Figure 2.2 illustrates a bivariate Gaussian distribution.

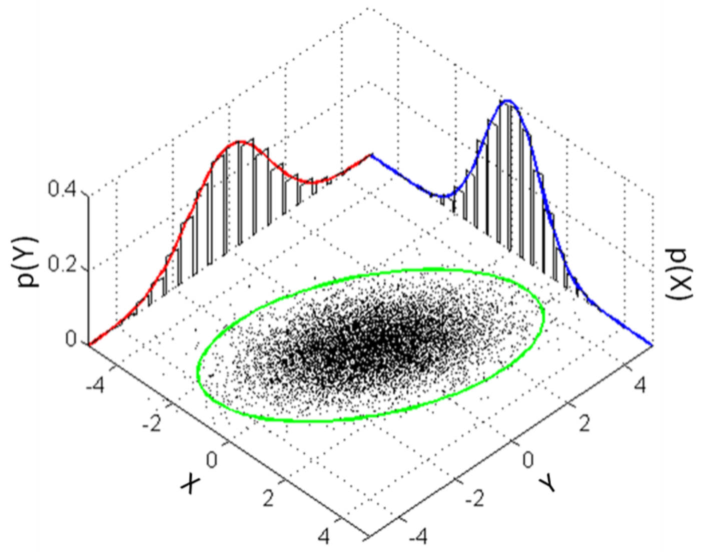


Figure 2.2: An example of a bivariate Gaussian distribution for $\mu = (0, 0)^T$ and $\Sigma = \begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix}$

The multivariate Gaussian distribution is very useful for examining the interactions between random variables, as the off-diagonal term (i, j) of the precision matrix $K = \Sigma^{-1}$, when normalized to have ones on the main diagonal, are known to be equal to negative partial correlations of variables X_i and X_j with respect to all other variables:

$$\rho_{ij|V\setminus\{v_i,v_j\}} = -\frac{\kappa_{ij}}{\sqrt{\kappa_{ii}\kappa_{jj}}}$$

[1]For completeness and reference, calculations of the univariate Gaussian distribution can be found in Appendix A

where κ_{ii} is the partial variance of node X_i given all other nodes.

For the Gaussian distribution partial correlations are equal to conditional correlations. Since for the joint Gaussian distribution all margins and conditional distributions are also Gaussian, the conditional correlation being equal to zero means that the variables are conditionally independent.

$$X_i \perp X_j | \bigcup_{v \in V \setminus \{v_i, v_j\}} (X_v) \iff \rho_{ij|V \setminus \{v_i,v_j\}} = 0 \iff \kappa_{ij} = 0 \tag{2.2}$$

This implies that there would be no edge between the corresponding vertices.

All information about the dependencies in the joint Gaussian distribution is included in the covariance matrix Σ or the precision matrix K. It is useful to rewrite K as follows, where $\alpha_i = \sqrt{\kappa_{ii}}$:

Define two matrices A and C

$$A = \begin{pmatrix} \sqrt{\kappa_{11}} & 0 & \cdots & 0 \\ 0 & \sqrt{\kappa_{22}} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sqrt{\kappa_{kk}} \end{pmatrix} \tag{2.3}$$

$$C = \begin{pmatrix} 1 & \frac{\kappa_{12}}{\alpha_1\alpha_2} & \cdots & \frac{\kappa_{1k}}{\alpha_1\alpha_k} \\ \frac{\kappa_{21}}{\alpha_2\alpha_1} & 1 & \cdots & \frac{\kappa_{2k}}{\alpha_2\alpha_k} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\kappa_{k1}}{\alpha_k\alpha_1} & \frac{\kappa_{k2}}{\alpha_k\alpha_2} & \cdots & 1 \end{pmatrix} \tag{2.4}$$

Then

$$K = ACA.$$

To be able to see the elements of the precision matrix and partial correlations in one matrix, we will use matrix $\mathcal{K}$, where the diagonal contains the square root of the partial variances, the upper triangular part the elements of the precision matrix and the lower triangular part the partial correlations.

$$\mathcal{K} = \begin{pmatrix} \sqrt{\kappa_{11}} & \kappa_{12} & \cdots & \kappa_{1k} \\ -\frac{\kappa_{21}}{\alpha_2\alpha_1} & \sqrt{\kappa_{22}} & \cdots & \kappa_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ -\frac{\kappa_{k1}}{\alpha_k\alpha_1} & -\frac{\kappa_{k2}}{\alpha_k\alpha_2} & \cdots & \kappa_{kk} \end{pmatrix}$$

Let us consider the following simple example.

Example 1. *Let $X = (X_1, X_2, X_3) \sim \mathcal{N}(\mu, \Sigma)$. with mean*

$$\mu = (0, 0, 0)^T$$

and

$$K = \begin{pmatrix} 3 & 0 & 3 \\ 0 & 10 & 5 \\ 3 & 5 & 8 \end{pmatrix} \tag{2.5}$$

the precision matrix.

Since the (1,2) element of this matrix is zero, the relationships between random variables in this example can be displayed as a graph presented in Figure 2.3.

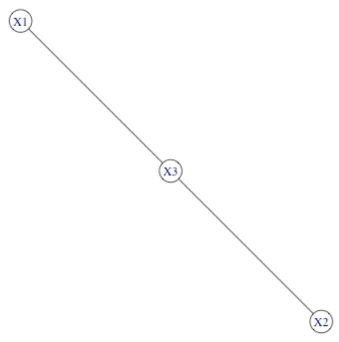


Figure 2.3: A grap corresponding to $K = \begin{pmatrix} 3 & 0 & 3 \\ 0 & 10 & 5 \\ 3 & 5 & 8 \end{pmatrix}$

This graph represents that X_1 and X_2 are conditionally independent given X_3 because X_3 lies on the 'path' between the two. To get from X_1 to X_2 following edges in the graph one needs to pass through X_3. X_3 separates X_1 and X_2 in the graph and this means that given the value of X_3 the information about X_1 does not change the distribution of X_2.

Creating these graphs would be straightforward if we have the precision matrix with exact zeros. However, in real-world scenarios, our knowledge is based on $\hat{K}$ estimated from the available data, in which it is very unlikely to observe exactly zero value. In the next section we show how the parameters of the Gaussian distribution and in particular how the precision matrix can be estimated.

GGMs use the Gaussian distribution due to its simplicity and the useful properties. Hence, when using these models to represent relationships in the data sets, it is crucial to ensure that the data is approximately Gaussian. Otherwise, the results may not be valid.

2.2.2. MLE of the Multivariate Gaussian Distribution

Consider n i.i.d. observations $Y = (Y_1, Y_2, ..., Y_n)^T$ of k-dimensional vector X, giving us a $n \times k$ dimensional data matrix.

The derivations of the likelihood and log-likelihood functions require careful handling of the matrix calculations involved. The derivations of the MLEs involve taking partial derivatives with respect to matrices and vectors which is not difficult in this case. For a a quadratic form $Q = y^T K y$ we have the following two derivatives that will be needed:

$$\frac{\partial Q}{\partial K} = yy^T$$
$$\frac{\partial Q}{\partial y} = 2Ky$$

We also use that the derivative of the log determinant of a matrix is equal to the transpose of the inverse matrix.

We get:

$$\begin{aligned} \log L(\mu, \Sigma; Y_1, Y_2, \dots, Y_n) &= \log\left(\prod_{i=1}^{n} \frac{1}{(2\pi)^{k/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(Y_i-\mu)^T\Sigma^{-1}(Y_i-\mu)\right)\right) \\ &= -\frac{nk}{2}\log(2\pi) - \frac{n}{2}\log|\Sigma| - \frac{1}{2}\sum_{i=1}^{n}(Y_i-\mu)^T\Sigma^{-1}(Y_i-\mu) \end{aligned} \tag{2.6}$$

First we will calculate the partial derivative with respect to μ:

$$\begin{aligned}\frac{\partial}{\partial\mu}\left(-\frac{nk}{2}log(2\pi)-\frac{n}{2}log(|\Sigma|)-\frac{1}{2}\sum_{i=1}^{n}(Y_i-\mu)^T\Sigma^{-1}(Y_i-\mu)\right)&=0\\ \sum_{i=1}^{n}\frac{\partial}{\partial\mu}(Y_i-\mu)^T\Sigma^{-1}(Y_i-\mu)&=0\\ \sum_{i=1}^{n}\Sigma^{-1}(Y_i-\mu)&=0\\ \Sigma^{-1}n\mu&=\Sigma^{-1}\sum_{i=1}^{n}Y_i\\ \hat{\mu}&=\frac{1}{n}\sum_{i=1}^{n}Y_i\end{aligned} \tag{2.7}$$

To simplify calculations we will rewrite the log likelihood function where we take the inverse of the covariance matrix Σ, which is the precision matrix K.

Now we will calculate the partial derivative with respect to K:

$$\begin{aligned}\frac{\partial}{\partial K}\left(-\frac{nk}{2}log(2\pi)+\frac{n}{2}log(|K|)-\frac{1}{2}\sum_{i=1}^{n}(Y_i-\hat{\mu})^TK(Y_i-\hat{\mu})\right)&=0\\ \frac{n}{2}(K^{-1})^T-\frac{1}{2}\sum_{i=1}^{n}(Y_i-\hat{\mu})(Y_i-\hat{\mu})^T&=0\\ (\Sigma)^T&=\frac{1}{n}\sum_{i=1}^{n}(Y_i-\hat{\mu})(Y_i-\hat{\mu})^T\\ \hat{\Sigma}&=\frac{1}{n}\sum_{i=1}^{n}(Y_i-\hat{\mu})(Y_i-\hat{\mu})^T\end{aligned} \tag{2.8}$$

The last step follows from the symmetry of Σ.

So the estimators of μ and Σ are:

$$\begin{aligned}\hat{\mu}&=\frac{1}{n}\sum_{i=1}^{n}Y_i\\ \hat{\Sigma}&=\frac{1}{n}\sum_{i=1}^{n}(Y_i-\hat{\mu})(Y_i-\hat{\mu})^T\end{aligned} \tag{2.9}$$

In further considerations we will assume $\mu = 0$ and denote:

$$W=\sum_{i=1}^{n}Y_iY_i^T=Y^TY \tag{2.10}$$

This gives:

$$\hat{\Sigma}=\frac{W}{n}$$

This maximum likelihood estimate of the covariance matrix exists if and only if W is symmetric and positive definite. This happens with probability one if $n \geq k$ and probability zero if $n < k$. This means that we need the number of observations to be equal or larger than the number of nodes or variables.

Using K and W we can now simplify the log likelihood equation:

$$\begin{aligned}\textbf{Maximize}: \log L(K) &= -\frac{nk}{2}\log(2\pi) - \frac{n}{2}\log|\Sigma| - \frac{1}{2}\sum_{i=1}^{n}(Y_i-\mu)^T\Sigma^{-1}(Y_i-\mu)\\ &= -\frac{n}{2}\log|\Sigma| - \frac{1}{2}\sum_{i=1}^{n}(Y_i-\mu)^T\Sigma^{-1}(Y_i-\mu) \qquad (2.11)\\ &= \frac{n}{2}\log|K| - \frac{1}{2}\operatorname{tr}(KW)\end{aligned}$$

The constant at the beginning is left out when maximizing this value. The Σ becomes K and makes the negative sign positive. Using our knowledge of traces we see that the last term becomes the trace of the product of K and W when $\mu = 0$.

Example 2. *To estimate the covariance matrix using estimator 2.9, we have simulated 250 observations from the Gaussian distribution with mean $(0,0,0)^T$ and precision matrix 2.5.*

Using equation 2.9 we get the following estimate of Σ presented with an accuracy to 3 decimal places:

$$\hat{\Sigma} = \begin{pmatrix} 0.812 & 0.217 & -0.412\\ 0.217 & 0.183 & -0.199\\ -0.412 & -0.199 & 0.383 \end{pmatrix}$$

The estimated precision matrix will thus be the inverse of $\hat{\Sigma}$:

$$\hat{K} = \begin{pmatrix} 2.707 & -0.124 & 2.842\\ -0.124 & 12.503 & 6.343\\ 2.842 & 6.343 & 8.946 \end{pmatrix}. \qquad (2.12)$$

To keep all the information together the matrix $\hat{\mathcal{K}}$ is included:

$$\hat{\mathcal{K}} = \begin{pmatrix} 1.645 & -0.124 & 2.842\\ 0.021 & 3.536 & 6.343\\ -0.578 & -0.600 & 2.991 \end{pmatrix} \qquad (2.13)$$

We see that element $(1,2)$ of $\hat{K}$ is not zero now, but still relatively small compared to the other values. Moreover, the partial correlation of X_1 and X_2,given X_3, is very small, indicating the possible conditional independence between these variables.

In the next section we will investigate if the element of the precision matrix should be fixed to be equal to zero.

2.3. Estimation of zeros in the Precision Matrix

We will investigate whether it is possible to set certain elements of the precision matrix to zero. Moreover, we will estimate the parameters of the simplified model and test if this model is supported by the data. This can be done with the likelihood ratio test as the considered models are nested.

2.3.1. Iterative Partial Maximization

To be able to set certain elements to zero we will use an algorithm that maximizes the log-likelihood described in Equation 2.11 for the vertices with fixed edges and then maximizes the equation for the edges with fixed vertices iteratively.

For k vertices and m edges we can store the vertices in a k-dimensional vector η and the edges in a m-dimensional vector δ. Then K is a function of η and δ, $K(\eta,\delta)$. Say we have a 3-variate Gaussian distribution with the partial correlation between X_1 and X_2 being zero, hence having 2 edges, we get the following K:

$$K = \begin{pmatrix} \eta_1 & 0 & \delta_1 \\ 0 & \eta_2 & \delta_2 \\ \delta_1 & \delta_2 & \eta_3 \end{pmatrix}$$

To estimate the values of η and δ we cannot use the maximum likelihood estimator as in Equation 2.9. To be able to constrain the precision matrix to have certain elements set to zero the method described below is used.

First, let us introduce a notation T^u as a $k \times k$ matrix of zeros and ones. This matrix will be used to indicate which elements in the matrix A or C are the same or can be fixed to zero in C.

The element

$$T^u_{ii} = \{1 \text{ if } i \in u, \text{ else } 0\}$$

resulting in T^u being a matrix with a single 1 on the diagonal. Moreover, if

$$T^u_{ij} = \{1 \text{ if } \{i,j\} \in u, \text{ else } 0\}$$

we get an adjacency matrix as T^u, with a 1 symmetrically placed.

Next, we merge all vertices and edges into a single $k+m$-dimensional vector θ. For example, in the 3-dimensional case, $\theta = (\eta_1, \eta_2, \eta_3, \delta_1, \delta_2)^T$. Then K can be expressed as a function of θ, $K(\theta) = \sum_u \theta_u T^u$.

Example 3. *For the matrix 2.12 we have the decomposition as follows:*

$$\begin{aligned} \hat{K} &= \begin{pmatrix} 2.707 & -0.124 & 2.842 \\ -0.124 & 12.503 & 6.343 \\ 2.842 & 6.343 & 8.946 \end{pmatrix} \\ &= 2.707 \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} + 12.503 \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} + 8.946 \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \\ &- 0.124 \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} + 2.842 \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} + 6.343 \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \end{aligned}$$

If we want to maximise the log-likelihood functions we cannot do so simultaneously for the vertices in A and edges in C, because their joint likelihood function is not in general concave [1].

To solve this problem, the likelihood is maximized in turn for matrix A and for matrix C. Both log-likelihood functions are strictly concave in their arguments. Hence the process iteratively updates the η and δ vector to maximize the appropriate likelihood. This is called iterative partial maximization.

The iteration step, where we maximize the log likelihood with respect to θ is as follows:

For the edges, the equation that maximizes the likelihood is:

$$tr(T^u AWA) = ntr(T^u C^{-1}), u \in E \tag{2.14}$$

This equation is solved iteratively with Newton's method, which needs a starting value which is taken as the value derived from the original maximum likelihood estimation.

The Newton iteration updates δ_u for every u $\in E$ in the following way:

$$\delta_u \leftarrow \delta_u + \frac{\Delta_u}{tr(T^u \hat{C}^{-1} tr(T^u \hat{C}^{-1}) + \Delta_u^2/2} \tag{2.15}$$

until convergence with $\Delta_u = tr(T^u \hat{C}^{-1}) - tr(T^u AWA)/n$.

Then η_u that maximizes the likelihood for given $u \in V$ while keeping δ and other values from the η-vector fixed is obtained by solving:

$$tr(T^u CAWA) = ntr(T^u), u \in V \tag{2.16}$$

To get a solution we define the Hadamard product of C and W with entries $Q_{ij} = C_{ij}W_{ij}$, which is positive definite since C and W are positive definite. The update step for η_u will be as follows:

$$\eta_u = \frac{-B + \sqrt{B^2 - 4n|V_u|D}}{2D} \tag{2.17}$$

where

$$B = \sum_{v_i \in V_u} \sum_{v_j \notin V_u} Q_{ij}\sqrt{k_{jj}}, D = \sum_{i \in V_u, j \in V_u} Q_{ij} \tag{2.18}$$

The process is repeated until convergence for u before moving on to the next node in $V \cup E$. However, in most cases it is sufficient to make only a few steps, ensuring the likelihood has increased. Thus the algorithm consists of two nested loops: an outer loop running over the elements of $V \cup E$, and an inner loop maximizing the log-likelihood with respect to θ_u while keeping all other parameters fixed. We repeatedly loop through all elements $u \in V \cup E$ until convergence.

These update iterations have been chosen such that the new matrix stays positive definite and the log likelihood is monotonically increasing, with a threshold value chosen to stop the iteration.[2] It has been shown in the Appendix of Lauritzen (1996) [9] that if there is a unique solution to the problem this method will converge to this unique solution. Additionally, in the appendix of Drton and Eichler (2006) [10] the proof of the convergence of the procedure is presented when it is assumed that there are a finite number of solutions. The process will converge to one of them.

This iterative method is implemented by the 'gRc' package in R. For a detailed derivation of the equations and their solutions (2.14 - 2.18) we refer to Appendix B. We can now use IPM to find the precision matrix that maximizes the log likelihood given constraints.

2.3.2. Model Selection

In the previous section we showed how the estimates of the precision matrix can be obtained when the constraints are known. However, it is not obvious how to decide which constraints are needed. There is a distinction between setting all elements close to zero to be zero simultaneously or setting them to zero one at the time. When the process is carried out one at a time, an element is set to zero and others are re-estimated. This changes the values of other parameters, and it could be that before the update they are small, but after they increase in magnitude.

Many different methods to set elements to zero can be considered (of which an overview can be found in Højsgaard et al. (2012b) [6]). There is no universally best-performing method, and since this thesis mainly focuses on the symmetry constraints (which will be treated in the next chapter), we opted to perform a stepwise selection process in fixing elements to zero.

The stepwise selection model starts from a fully connected graph and attempts to delete edges one at a time. When an edge is chosen for deletion, the precision matrix is updated with the optimization scheme discussed in the previous section. This method is implemented in gRbase, part of the 'gR' environment (a collection of packages created in R to work with GGMs). This model uses a different optimization scheme called Iterative Proportional Scaling. We have tested and found that both approaches lead to the same results.

After estimating the parameters of the model with a deleted edge, the Likelihood Ratio Test is used to test whether to accept or reject the deletion at a chosen significance level. If the model is not rejected, the process continues with a new candidate edge. The model compares each deletion with the previous deletions and not with the full model. The simulation study in Chapter 3 will analyze the performance of this process. A specified number of iterations must be given to the model, which we set to 1000. A higher number of repetitions has not resulted in improvements.

We also use a headlong search, which randomly traverses the edges instead of deleting edges starting with the edge deletion corresponding to the highest p-value. This is faster and results in more edge deletions in most cases. If we use the search order that is not random, certain edges don't get deleted because of the order it follows. In contrast, the headlong approach gives slightly different results each

[2]It has been shown through empirical evidence that K will stay positive definite with the newton iteration[1].

time it is run. This variation occurs because certain partial correlations may change in magnitude when one is set to zero, often preventing further edge deletions after this change.

The test used in the stepwise selection model is the Likelihood Ratio test. The likelihood ratio test is a statistical method used to compare the goodness of fit of two nested models based on their likelihood functions. In the context of our analysis, we use the likelihood ratio test to assess the fit of various restrictions in our graphical model. A pseudocode for the stepwise selection model can be found below.

Algorithm 1 Stepwise Selection with Random Search Order and Likelihood Ratio Test

```
Require: Data, pvalue, num_repetitions
Ensure: List of Deleted Edges
 1: Initialize FullGraph with all possible edges
 2: Initialize Edges as the list of all edges in FullGraph
 3: Initialize DeletedEdges as an empty list
 4: for rep in 1 to num_repetitions do
 5:     Shuffle(Edges) to randomize the search order
 6:     AnyDeletion $\leftarrow$ False
 7:     for each edge in Edges do
 8:         Backup the current PrecisionMatrix
 9:         Delete edge from FullGraph
10:         Update PrecisionMatrix with respect to the deleted edge
11:         if LikelihoodRatioTest(FullGraph, PrecisionMatrix) > pvalue then
12:             Append edge to DeletedEdges
13:             AnyDeletion $\leftarrow$ True
14:         else
15:             Restore the previous PrecisionMatrix
16:             Restore the edge in FullGraph
17:         end if
18:         if AnyDeletion then
19:             break
20:         end if
21:     end for
22:     if not AnyDeletion then
23:         break
24:     end if
25: end for
        return FullGraph, DeletedEdges
```

In the test the null hypothesis H_0 is that both distributions explain the observations similarly, while the alternative hypothesis H_1 is that the distributions are significantly different.

Let $L(\theta)$ denote the likelihood function of the unrestricted model, and $L_0(\theta_0)$ denote the likelihood function of the restricted model, where θ represents the parameters of the unrestricted model and θ_0 represents the parameters of the restricted model. The parameters correspond not only to the nodes, but also the edges that are not constrained to zero. The likelihood ratio test statistic is given by:

$$\lambda = -2\log\left(\frac{L_0(\hat{\theta}_0)}{L(\hat{\theta})}\right) = -2\left(\log L_0(\hat{\theta}_0) - \log L(\hat{\theta})\right)$$

where $\hat{\theta}$ and $\hat{\theta}_0$ are the maximum likelihood estimates of the parameters under the unrestricted and restricted models, respectively.

The likelihood ratio test statistic λ follows a chi-squared distribution with degrees of freedom equal to the difference in the number of parameters between the two models. We can use this distribution to determine the statistical significance of the test and decide whether the restrictions imposed on the model are justified.

Example 4. *Using the backwards stepwise selection method with a random search order (headlong) on our example yields a deletion of edge (X_1, X_2) with a p-value of 0.7355.*

We can check this result by calculating the log-likelihood of both models with Equation 2.11. For the unrestricted model this yields a value of 184.2455. For the restricted model the 'gRc' package returns the log likelihood value of 184.1887. So the likelihood ratio test returns:

$$\begin{aligned}\lambda &= -2\left(\log L_0(\hat{\theta}_0) - \log L(\hat{\theta})\right)\\ &= -2\,(184.1887 - 184.2455)\\ &\approx 0.1136\end{aligned} \tag{2.19}$$

This statistic is then Chi-squared with 1 degree of freedom, since the difference in parameters between the two models in the one index that was set to zero. This gives the p-value of 0.7355, being the same result as the stepwise selection model.

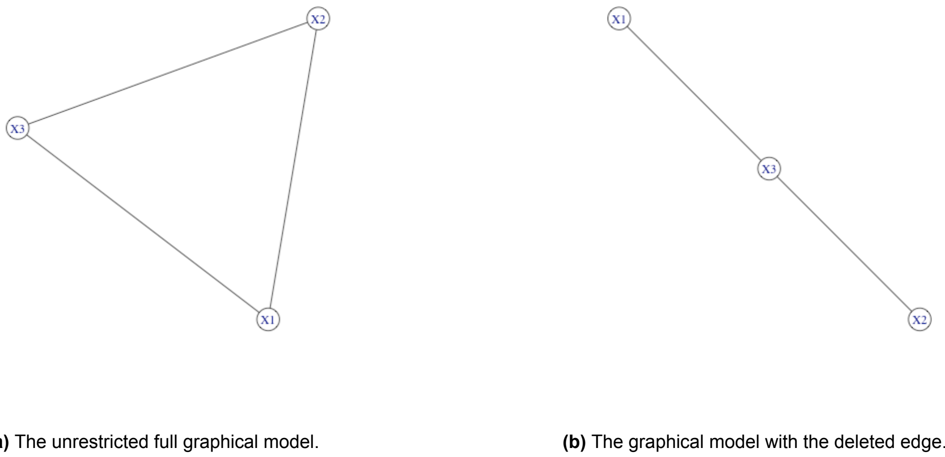
The stepwise selection model will now also look at a possible deletion of the next edge after accepting this deletion. When we try to delete $X1 : X_3$ we get a log-likelihood of 81.266. This gives a test statistic of $-2(86.255 - 184.1887) = 195.868$, which will give a p-value of 0.0, rejecting the possibility of deleting this edge. We will then look at the final edge left, which is $X_2 : X_3$, which gives a log-likelihood of 81.266 when deleting of the edge. This value is even smaller than the deletion of the other edge and will thus also have a p-value of 0.0. Since we have checked every edge now and there is no edge deletion yielding a p-value larger than 0.01, we can stop and get the final model.

Now we have deleted the edge between X_1 and X_2 we can find the updated precision matrix:

$$\hat{\mathcal{K}} = \begin{pmatrix} 1.645 & 0 & 2.904 \\ 0 & 3.535 & 6.473 \\ -0.586 & -0.608 & 3.013 \end{pmatrix}$$

The 'gRc' package returns the matrix in the format of $\mathcal{K}$. If we compare these values to before the restriction, we see that the values of the precision matrix have changed quite a bit.

The model with the deleted edge between X_1 and X_2 is shown in Figure 2.4.



(a) The unrestricted full graphical model. **(b)** The graphical model with the deleted edge.

Figure 2.4: The deletion of an edge in the graphical model.

3

Symmetry in Gaussian Graphical Models

Setting certain values in the precision matrix to zero, as discussed in the previous chapter, enhances interpretability by removing edges from the graphical model. Building on this, we can also introduce symmetry constraints to the precision matrix. This adjustment not only improves interpretability but also simplifies the model by reducing the number of parameters, thereby increasing parsimony. This approach is particularly beneficial in situations involving a small number of observations relative to the number of parameters. These symmetries will be introduced by specifying classes of nodes or edges, hence by defining colored graphs.

3.1. Graph Colouring

Consider a graph $G = (V, E)$. Let us partition the vertices of G where $r \leq |V|$ into disjoint sets $V_1, \dots, V_r$ referred to as **vertex colour classes (VCCs)**. Similarly, colouring the edges of G with $s \leq |E|$ is a partition of the edges into disjoint sets $E_1, \dots, E_S$, referred to as **edge colour classes (ECCs)**. Thus, $(\mathcal{V}, \mathcal{E})$ represents a **coloured graph** corresponding to $G = (V, E)$ with $V = V_1 \cup \dots \cup V_r$ and $E = E_1 \cup \dots \cup E_r$.

Note that we do not require that the adjacent vertices are not in the same color class as is often assumed in graph theory.

A colour class with a single element is called **atomic**, while a colour class with more than one element is called **composite**. When visualising a coloured graph, atomic colour classes are displayed using neutral colours (black or white), whereas composite colour classes are shown with other distinct colours.

Where we had the k-dimensional vector η and the m-dimensional vector δ in Chapter 2, we can now reduce the size of these vectors to a r-dimensional vector η and s-dimensional vector δ.

Example 5. *We can now impose constraints as in the following example:*

- X_1 *and* X_5 *are in VCC 1,*
- X_3*,* X_4*, and* X_6 *are in VCC 2,*
- X_2 *is atomic in VCC 3.*
- *Edge between* X_1, X_2 *and* X_2, X_5 *are in ECC 1,*
- *Edge between* X_1, X_3 *and* X_2, X_6 *are in ECC 2,*
- *Edge between* X_2, X_3 *and* X_4, X_5 *are in ECC 3,*

There are no atomic edge classes and the edges not in the list are assumed to have a partial correlation of zero. This leaves the edges out of the precision matrix and the Graphical model.

This makes K *look like this:*

$$K = \begin{pmatrix} \eta_1 & \delta_1 & \delta_2 & 0 & 0 & 0 \\ \delta_1 & \eta_2 & \delta_3 & 0 & \delta_1 & \delta_2 \\ \delta_2 & \delta_3 & \eta_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & \eta_3 & \delta_3 & 0 \\ 0 & \delta_1 & 0 & \delta_3 & \eta_1 & 0 \\ 0 & \delta_2 & 0 & 0 & 0 & \eta_3 \end{pmatrix}$$

These symmetric restrictions significantly reduce the number of parameters and thus also increases the accuracy of an estimation. This graphical model represented by the precision matrix with symmetries is presented in Figure 3.1, where different colors representing different edge and node symmetries can be seen.

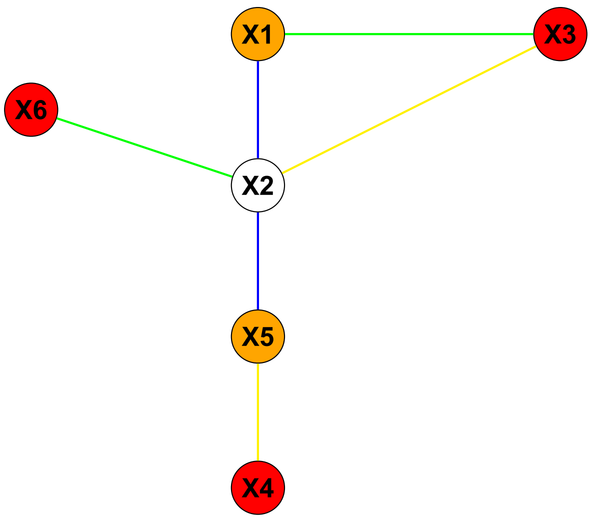


Figure 3.1: A 6-dimensional Graphical model with symmetry constraints.

3.2. Estimation of Symmetries in the Precision Matrix

To be able to impose that certain elements in the precision matrix are the same we have to define when elements are added to the same symmetry classes. A symmetry class is a set of elements in V or E with elements that are equal.

After the edges have been deleted, we will estimate the symmetries from the resulting model.

For a given graph (possibly with some edges removed), the estimation of parameters of the model under symmetry constraints follows steps similar to setting zero's in equation 2.14 - 2.18. In those steps, we iterated through the vertex and edge classes stored in η and δ, while keeping the log likelihood maximized for θ. For symmetry classes, the only difference is that we iterate through the symmetry classes $\mathcal{V}, \mathcal{E}$ instead of each element of the vertex and edge classes V, E.

The Newton iteration starts by updating δ_u for every $u \in \mathcal{E}$.

We iterate

$$\delta_u \leftarrow \delta_u + \frac{\Delta_u}{tr(T^u\hat{C}^{-1}tr(T^u\hat{C}^{-1}) + \Delta_u^2/2} \tag{3.1}$$

until convergence with $\Delta_u = tr(T^u\hat{C}^{-1}) - tr(T^uAWA)/n$.

We can then maximize η_u for given $u \in \mathcal{V}$ while keeping δ and other values from the η-vector fixed by solving the following system:

$$tr(T^uCAWA) = ntr(T^u), u \in \mathcal{V} \tag{3.2}$$

which has as solution:

$$\eta_u = \frac{-B + \sqrt{B^2 - 4n|V_u|D}}{2D} \tag{3.3}$$

where
$$B = \sum_{v_i \in V_u} \sum_{v_j \notin V_u} Q_{ij}\sqrt{k_{jj}}, D = \sum_{v_i \in V_u, v_j \in V_u} Q_{ij} \tag{3.4}$$

Example 6. *Let us go back to the estimations of A and C done in the previous section, after we had set the zero's:*
$$\hat{\mathcal{K}} = \begin{pmatrix} 1.645 & 0 & 2.904 \\ 0 & 3.535 & 6.473 \\ -0.586 & -0.608 & 3.0132 \end{pmatrix}$$

we see that the values $\hat{\mathcal{K}}_{13}$ and $\hat{\mathcal{K}}_{23}$ are similar. The diagonal elements are not that close to each other. If we assume that $\mathcal{K}_{13}$ and $\mathcal{K}_{23}$ are equal we get the estimate of the precision matrix with the additional edge symmetry $\hat{K}$:
$$\hat{K} = \begin{pmatrix} \eta_1 & 0 & \delta_1 \\ 0 & \eta_2 & \delta_1 \\ \delta_1 & \delta_1 & \eta_3 \end{pmatrix} \tag{3.5}$$

Where the starting value for δ_1 in IPM can be chosen as an average of the two values in C.

Adding this symmetry constraint gives the following estimation of $\hat{K}$:

$$\hat{\mathcal{K}} = \begin{pmatrix} 1.656 & 0 & 2.977 \\ 0 & 3.510 & 6.310 \\ -0.597 & -0.597 & 3.012 \end{pmatrix}$$

For the unrestricted model the log likelihood yielded a value of 184.1887. With the added symmetry constraint the log-likelihood value is 184.1263. So the likelihood ratio test returns:

$$\begin{aligned} \lambda &= -2\left(\log L_0(\hat{\theta}_0) - \log L(\hat{\theta})\right) \\ &= -2\,(184.1263 - 184.1887) \\ &\approx 0.1248 \end{aligned} \tag{3.6}$$

This statistic is then Chi-squared with 1 degree of freedom, since the difference in parameters between the two models is the single symmetry constraint. This gives a p-value of 0.724. In the case we would have set significance level $\alpha = 0.01$, we would not reject the model with the symmetry constraint against the model with the deleted edge.

We can also test our model with symmetry constraints against the full model.

$$\begin{aligned} \lambda &= -2\left(\log L_0(\hat{\theta}_0) - \log L(\hat{\theta})\right) \\ &= -2\,(184.1263 - 184.2455) \\ &\approx 0.2384 \end{aligned} \tag{3.7}$$

Under 2 degrees of freedom this test statistic has a p-value of 0.888, further proving the overall fit of our model.

3.3. Finding symmetry classes

In high-dimensional data analysis, identifying patterns and structures within the data can be challenging, particularly when dealing with complex dependencies and large datasets. While the gRc package offers a method to identify symmetry classes that maximize the log-likelihood by iteratively joining and splitting color classes, this approach can be computationally inefficient. For k vertices and a $k \times k$ precision matrix, we have $\frac{k(k-1)}{2} - e$ edges, where e is the number of deleted edges. The total number of possible symmetry classes is:

$$\text{Total number of possible symmetry classes} = 2^{k+\frac{k(k-1)}{2}-e}.$$

For a 3-dimensional graph with 1 deleted edge, this results in 32 possible symmetries. For a 14-dimensional problem with 5 deleted edges, it results in approximately 1.27×10^{30}. Manually exploring all these possibilities is impossible, which means we have to look for a faster, more efficient, and automated method.

Hierarchical clustering [11][12] is a powerful tool for identifying elements that are close to each other according to some metric, hence in this section we will apply hierarchical clustering to detect symmetry constraints in GGMs.

The primary motivation for using hierarchical clustering in the context of GGMs lies in its ability to organize groups of elements that are 'similar' systematically. Hierarchical clustering is flexible and can be applied to various types of data without the need for prior specification of the number of clusters. This adaptability makes it suitable for different datasets and applications, allowing for scalable analysis as the data size grows.

3.3.1. Methodology of Hierarchical Clustering

Hierarchical clustering begins with each data point as its own cluster and iteratively merges the closest pairs of clusters based on a chosen distance metric. In this thesis we opted for Ward metric to be applied in the hierarchical clustering.

Let D be a data set with n elements. Denote as D_i and D_j two disjoint subsets of elements of D, also called clusters. The Ward distance between clusters D_i (with n_i elements) and D_j (with n_j elements) is defined as:

$$\Delta(D_i, D_j) = \frac{n_i n_j}{n_i + n_j} ||\overline{D_i} - \overline{D_j}||^2 \tag{3.8}$$

Here, $\overline{D'}$ is the centroid or average of subset $D' \subseteq D$, calculated as $\overline{D'} = \frac{1}{|D'|} \sum_{d \in D'} d$. The term $\frac{n_i n_j}{n_i + n_j}$ is a weight that accounts for the sizes of the clusters.

The following pseudocode outlines the process of hierarchical clustering:

Algorithm 2 Hierarchical Clustering with Ward's Linkage

```
Require: List of points
Ensure: Linkage matrix
 1: Initialize each point as a separate cluster
 2: Compute the initial distance matrix
 3: while more than one cluster exists do
 4:     Find the two closest clusters
 5:     Merge the two closest clusters
 6:     Compute the centroid of the new cluster
 7:     Update distances using Ward's method
 8:     Record the merge in the linkage matrix
 9: end while
        return Linkage matrix
```

This process continues until all points are merged into a single cluster. The result is a dendrogram, a tree-like diagram illustrating the arrangement of clusters and the distances between each merge. An example dendrogram, created using a sample of 10 random values between 0 and 1, is shown in Figure 3.2.

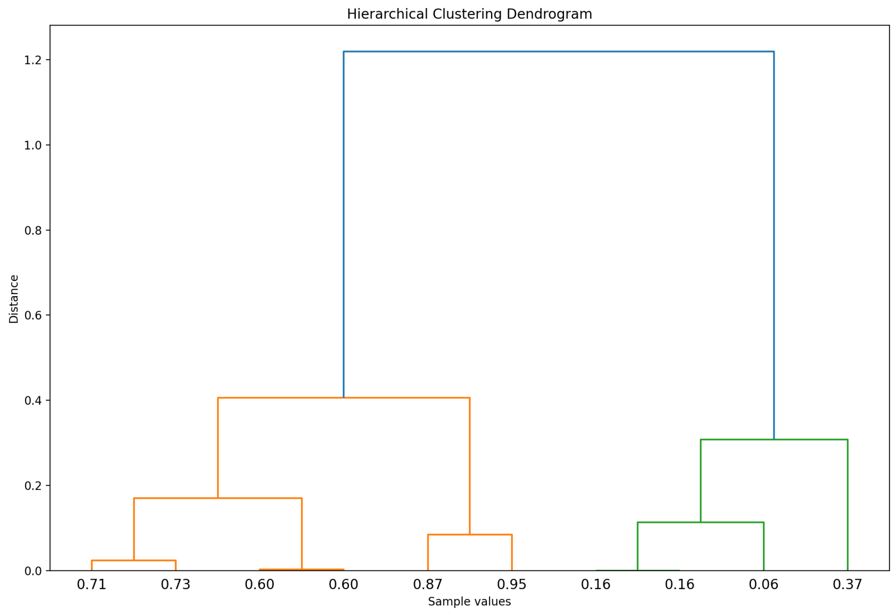


Figure 3.2: Hierarchical Clustering employed on a sample of 10 random values between 0 and 1

In Figure 3.2 the values of the sample are shown on the x-axis. On the y-axis the values of the Ward distance corresponding to the merge of clusters are indicated. Below we show in detail one update step of the hierarchical clustering algorithm for the example illustrated by the dendrogram in Figure 3.2.

Example 7. *First the initial distance matrix is calculated. The Ward distances between points are just Euclidian distances. Since 0.60 and 0.16 appear in the data twice, their distance is zero and they are merged.*

Table 3.1: Distance Matrix for Sample Values

	0.71	**0.73**	**0.60**	**0.87**	**0.95**	**0.16**	**0.06**	**0.37**
0.71	0.00	0.02	0.11	0.16	0.24	0.55	0.65	0.34
0.73	0.02	0.00	0.13	0.14	0.22	0.57	0.67	0.36
0.60	0.11	0.13	0.00	0.27	0.35	0.44	0.54	0.23
0.87	0.16	0.14	0.27	0.00	0.08	0.71	0.81	0.50
0.95	0.24	0.22	0.35	0.08	0.00	0.79	0.89	0.58
0.16	0.55	0.57	0.44	0.71	0.79	0.00	0.10	0.21
0.06	0.65	0.67	0.54	0.81	0.89	0.10	0.00	0.31
0.37	0.34	0.36	0.23	0.50	0.58	0.21	0.31	0.00

Since the smallest off-diagonal value is 0.02, we will merge the points corresponding to this distance, which are 0.71 and 0.73. The centroid of this cluster is $\frac{1}{2}(0.71 + 0.73) = 0.72$. The distances from this centroid to all other clusters (other single points) are calculated with formula 3.8 and are included in Table 3.2. Moreover, we can observe that indeed at level 0.02 elements 0.71 and 0.73 are merged in the dendrogram.

Table 3.2: New Distance Matrix After One Step of Clustering Using Ward's Distance Metric

	Merged (0.72)	**0.60**	**0.87**	**0.95**	**0.16**	**0.06**	**0.37**
Merged (0.72)	0.00	0.1386	0.1720	0.2093	0.5115	0.5986	0.3106
0.60	0.1386	0.00	0.27	0.35	0.44	0.54	0.23
0.87	0.1720	0.27	0.00	0.08	0.71	0.81	0.50
0.95	0.2093	0.35	0.08	0.00	0.79	0.89	0.58
0.16	0.5115	0.44	0.71	0.79	0.00	0.10	0.21
0.06	0.5986	0.54	0.81	0.89	0.10	0.00	0.31
0.37	0.3106	0.23	0.50	0.58	0.21	0.31	0.00

The process is repeated and the distance matrix updated, until we have the final matrix where all clusters have been merged to a single cluster.

The choice of distance metric is critical in hierarchical clustering. Common distances include Euclidean, Manhattan and Ward. The selection of an appropriate metric depends on the nature of the data and the specific objectives of the analysis. The selection of Ward's distance metric for hierarchical clustering is motivated by its ability to minimize within-cluster variance and create compact and interpretable clusters. We employ an agglomerative (bottom-up) approach, starting with each vertex and edge as individual clusters and merging them iteratively. This approach ensures that the most similar elements are grouped first, creating a hierarchical structure that reflects the data's similarities. Determining when to stop merging clusters is essential for maintaining meaningful and interpretable results. One effective stopping criterion is based on the likelihood values obtained from the model. By monitoring the changes in the likelihood as clusters are merged, we can identify the point at which further merging no longer significantly improves the model fit.

In this thesis, we implement hierarchical clustering using the 'sklearn.cluster' module in Python.

3.3.2. Application to GGMs

Applying hierarchical clustering to GGMs involves several steps:

We begin with an atomic clustering of vertices based on values of partial variances found in matrix A and partial correlations corresponding to edges found in matrix C. Next, we iteratively merge the closest clusters based on the Ward distance until a certain threshold value is reached. Once the clustering is complete, we apply the identified symmetry constraints to estimate the reduced model.

To evaluate the fit and performance of the constrained model, we use the likelihood ratio test.

By integrating hierarchical clustering into the modeling process, we can find symmetries in the data faster and more effective.

We will create dendrograms for the elements in A and C separately and use these dendrograms to determine which elements should be considered to be equal. For small models (not too many nodes) it is possible to update the precision matrix and calculate the p-value for our hypothesis test at each step in the dendrogram. However, as the dataset grows, this approach becomes computationally expensive. To work around this problem we will implement a threshold approach and stops the merging of the clustering when the distance becomes larger than the chosen threshold value. We have chosen 0.15 as the default threshold value. We will investigate the choice of this value further.

Example 8. *We will use Hierarchical Clustering on the matrix A and C from the example presented in Chapter 2.*

$$\hat{A} = \begin{pmatrix} 1.645 & 0 & 0 \\ 0 & 3.535 & 0 \\ 0 & 0 & 3.013 \end{pmatrix} \hat{C} = \begin{pmatrix} 1 & 0 & 0.586 \\ 0 & 1 & 0.608 \\ 0.586 & 0.608 & 1 \end{pmatrix} \tag{3.9}$$

The Hierarchical Clustering applied for these matrices yields the following dendrogram shown in Figure 3.3

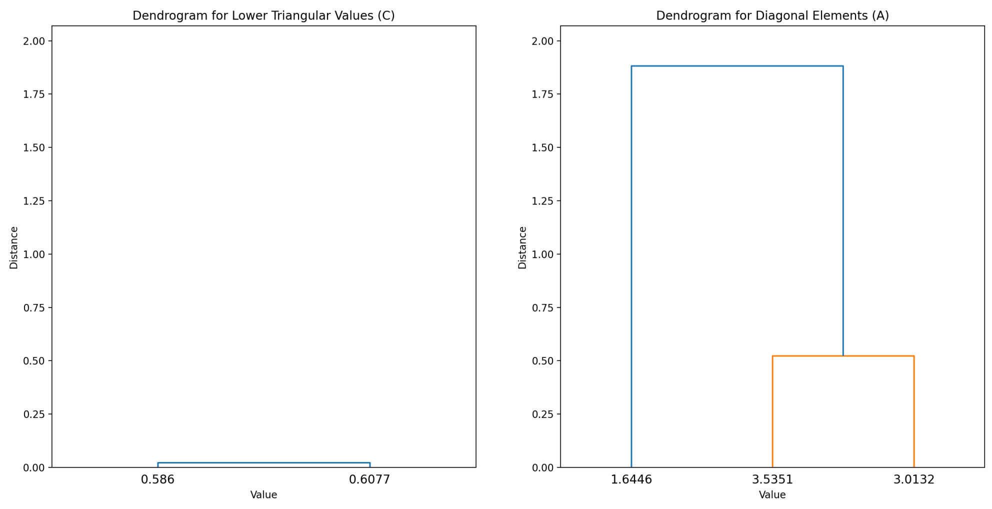


Figure 3.3: Hierarchical Clustering employed on matrix A and C.

When the threshold is chosen to be 0.15, the following clusters shown in Table 3.3 are obtained.

Cluster ID	Elements (Row, Column)
ECCs	
ECC1	(3, 1), (3, 2)
VCCs	
VCC1	1
VCC2	2
VCC3	3

Table 3.3: Grouped Elements with threshold level 0.15.

We observe that the distance between the two partial correlations is below our threshold value, which is why both edges are considered to be equally strong (edge symmetry). Although the distance between partial variances is also close, it does not meet the threshold value we have chosen. We have already tested this symmetry constraint of edges in Chapter 3 with the p-value found by Equation 3.6, being 0.724.

If we choose a higher distance, we could merge vertex X_2 and X_3. This model would lead to a log-likelihood of 177.3205. Testing this model against the model where all nodes are in different color classes and both edges in one color class yields a p-value of 0.0002. With significance level 5% we reject this symmetry constraint.

In the next chapter a simulation study is presented where different threshold values are considered. Increasing the threshold value decreases the number of clusters, resulting in fewer parameters to estimate. Therefore, the threshold value can be selected based on the number of observations and the desired accuracy of the estimation.

4 Simulation Study Results

In this simulation study, we analyze a 6-dimensional example to observe how varying thresholds in hierarchical clustering and sample sizes affect the accuracy of fitting GGMs. The main goal is to determine the impact of these factors on the acceptance rate and accuracy of estimating the number of correct edges and ECCs of the models.

4.1. Simulation Parameters

- **Distance Thresholds**: 0.05, 0.15, and 0.25
- **Sample Sizes**: 50, 500, 1000, and 10.000, with 100 repetitions each.
- **Stepwise selection iterations**: 1000 iterations using headlong stepwise selection based on the Likelihood Ratio Test with a p-value of 0.01 to delete edges.
- **Acceptance Rate 1, 2 and 3**: A model is accepted if the p-value of testing with the Likelihood Ratio Test is greater than 0.01. AR1: Full model against the model with deleted edges, AR2: Model with deleted edges against the model with symmetries, AR3: Full model against the model with symmetries.
- **Correct VCC Criteria**: A model has the same VCC if the clusters are the same for the calculated model and the original model.
- **Correct Edges Criteria**: A model has the same edges if the model at least includes the edges of the original model.
- **Correct ECC Criteria**: A model has the same ECCs if the clusters are the same for the calculated model and the original model.

4.2. Key Simulation Matrix $\mathcal{K}$

The matrix $\mathcal{K}$ is parameterized as follows:

$$\mathcal{K} = \begin{pmatrix} \sqrt{14} & 3.07 & 0 & -7.26 & -3.89 & 0 \\ -0.3 & \sqrt{7.5} & 4.28 & 0 & -1.42 & -1.13 \\ 0 & -0.57 & \sqrt{7.5} & -2.85 & 0 & 0 \\ 0.56 & 0 & 0.3 & \sqrt{12} & -1.8 & 0 \\ 0.3 & 0.15 & 0 & 0.15 & \sqrt{12} & 5.41 \\ 0 & 0.15 & 0 & 0 & -0.57 & \sqrt{7.5} \end{pmatrix}$$

This matrix represents the dependencies and partial correlations among the six variables in the GGM.

The following symmetries are imposed in the matrix:

- K_{11}: $\sqrt{14}$
- $K_{22} = K_{33} = K_{66}$: $\sqrt{7.5}$

- $K_{44} = K_{55}$: $\sqrt{12}$
- ρ_{12}: -0.3
- ρ_{14}: 0.56
- ρ_{15} and ρ_{34}: 0.3
- ρ_{25}, ρ_{26}, and ρ_{45}: 0.15
- ρ_{23} and ρ_{56}: -0.57

4.3. Results Analysis

In this section, we will analyze the results of simulations. We focus on the impact of threshold levels and sample sizes on the number of correctly identified edges, symmetry classes, and the acceptance rates of the model.

Table 4.1: Number of edges counted correctly and total number of edges kept in the model.

Sample Size	Correct Edges found (out of 900)	Total Edges kept (out of 1500)
10.000	900	1035
1000	895	1044
500	878	1033
50	517	616

Table 4.2: Sample sizes, threshold levels, and number of correctly identified symmetry classes.

Sample Size	Threshold Level	Correct ECCs (out of 500)	# Samples with correct VCCs
10.000	0.05	500	80
10.000	0.15	500	100
10.000	0.25	388	100
1000	0.05	441	5
1000	0.15	476	62
1000	0.25	343	72
500	0.05	394	2
500	0.15	443	38
500	0.25	298	50
50	0.05	179	0
50	0.15	214	0
50	0.25	195	3

Table 4.3: Acceptance Rates 1 (full model against deleted edges), 2 (deleted edges against symmetries), and 3 (full model against symmetries) for different sample sizes and threshold levels.

Sample Size	Threshold level	AR1	AR2	AR3
10.000	0.05	99	99	100
10.000	0.15	99	99	100
10.000	0.25	99	0	0
1000	0.05	99	99	100
1000	0.15	99	83	86
1000	0.25	99	6	9
500	0.05	100	100	100
500	0.15	100	84	84
500	0.25	100	16	22
50	0.05	83	94	89
50	0.15	83	100	95
50	0.25	83	100	91

Observations

In the results presented above, we observe that as the sample size grows, the number of correctly identified edges also increases. For the smallest sample size, we find only 517 of the correct edges, while the number of correctly identified edges rises to 878, 895, and 900 for the medium, large, and very large sample size, respectively. In almost all cases, it is not exclusively the correct edges that are identified; there are often edges left in the model that should not be there. This is likely due to the step-wise selection method that was used. Other edge deletion methods might perform better. However, we also see that the total number of edges remains roughly the same for the medium, large, and very large sample size. Only for the small sample size a significant difference in the total number of edges found is observed, indicating that too many edges are deleted in this case. This is most likely due to the tests accepting a deletion very easily for the small number of observations, thus resulting in a lot of edge deletions.

As far as ECC symmetries are concerned, we first recall that there are 5 ECCs to discover per simulation. With 100 simulations, that adds up to 500 ECCs per threshold level. We find that in the case of large sample sizes and a low threshold all ECCs are captured. Naturally, when the threshold increases, the number of correctly identified ECCs decreases. This decrease is due to merging ECCs that should not be merged. For the medium sample size, we find the highest number of correctly identified ECCs for threshold level 0.15, while 0.05 and 0.25 give slightly worse results (worst in the case of threshold 0.25). In all cases, we see that 0.15 yields the highest number of correct ECCs, suggesting that the optimal threshold level for the ECCs is around 0.15. For the small sample size the results are not satisfactory due to estimation variability.

In the case of VCCs, we see that the most correct VCCs are found at higher threshold levels, indicating that we should consider the ECCs and VCCs separately in the clustering process. Moreover, we observe that the thresholds chosen for the edges are not suitable for nodes, indicating the need for higher thresholds for VCCs. Once again, we conclude that the very large sample size yields the best results. In this case, we see an even larger drop in performance as the sample size decreases, leading us to believe that VCC estimation needs higher threshold levels to achieve accurately estimations.

Looking at the acceptance rates calculated throughout the simulation process, most seem to perform well. Only for small sample sizes the results are not satisfactory.

For AR1, we see very high acceptance rates in all sample sizes, meaning that the stepwise performs well in deleting edges without resulting in overall statistical insignificance.

AR2 also shows us that testing the model with deleted edges against the model with symmetry constraints decreases in a similar way as AR3 did, which makes sense, as the symmetries are the contributing factor to the change in likelihood.

We also see that the AR3 is 100% in the very large sample size as long as we don't set our threshold level too high. A similar decrease in acceptance rate occurs when the large and medium sized data are used. The performance of the method is not consistent in the case of small sample sizes. This was to be expected, as the estimates of the parameters are unreliable for small data sets.

Conclusion

To conclude, we observe that varying the threshold levels can greatly impact the number of correctly identified ECCs and improve the acceptance rate 2 and 3 drastically. To choose the appropriate threshold levels, it is necessary to analyze the dendrogram. Starting from a high threshold level will likely merge too many elements and the model will be rejected using the likelihood ratio test (as seen in AR2 and AR3). Reducing the threshold leads to decoupling some of the incorrectly merged groups. We will use this approach in the analysis of EEG data.

We have observed that a sufficiently large sample size is necessary to get satisfactory results. In our six dimensional example, the very large sample of 10000 is more than enough to obtain good results but 500 is not too bad, given we choose a somewhat correct threshold level. All sample sizes, except for the small sample size, are also very similar in the number of correct edges found relative to the total number of edges found. The large sample size of 1000 observations gets close to the correct number of ECC symmetries with acceptable acceptance rates. The only downside to not using the very large sample size is the decrease in correct VCCs.

These results highlight the importance of choosing appropriate thresholds and sample sizes in hierarchical clustering for GGMs, ensuring model accuracy and robustness. The effectiveness of the threshold level depends significantly on variability of the real data. Additionally, the sample size plays a crucial role in determining the model's performance, with larger sample sizes generally providing more reliable and precise estimations. Overall, these insights show the necessity of careful selection of threshold levels and adequate sample sizes to enhance model performance and ensure the robustness of the hierarchical clustering process.

5
EEG Signals

In this chapter, we explore data of EEG signals in different brain regions. Electroencephalography (EEG) is a non-invasive technique used to measure electrical activity in the brain, which is crucial for various applications such as clinical diagnosis, neuroscience research, and Brain-Computer Interface (BCI) development.

We start with an overview of the different parts of the brain that are typically monitored using EEG. Figure 1.1 provided in the introduction illustrates where the electrodes should be placed according to the internationally recognized 10-20 system. Our dataset, which will be described in the next section, utilizes an extended version of this 10-20 system called the 10-10 system, shown in Figure 5.1. The electrodes are placed on a patients head according to the pattern presented below.

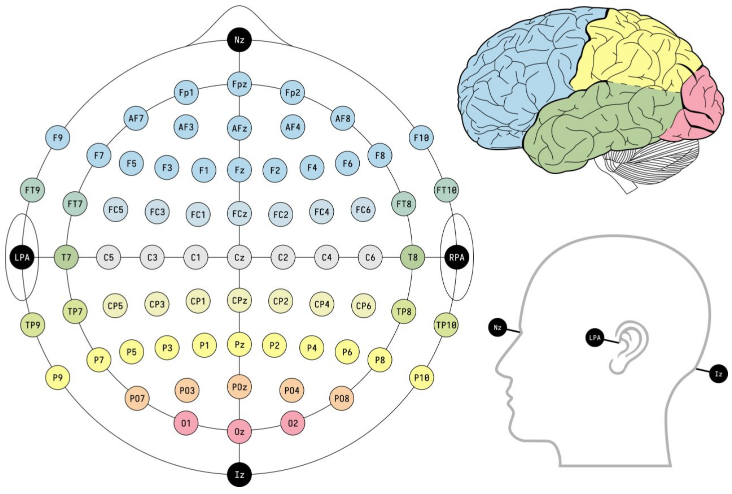


Figure 5.1: Extended 10-10 System.

5.1. MindBigData: A Comprehensive EEG Dataset

Our analysis is based on the MindBigData dataset, which is a large-scale project initiated to provide an extensive collection of brain signal data for research and development in neuroscience and machine learning. The MindBigData project includes various sub-datasets, each capturing different aspects of brain activity through different EEG headsets.

The dataset used in this chapter is the EPOC dataset, which consists of EEG recordings captured using a 14-channel EEG device.

Sample Characteristics The dataset contains thousands of EEG recordings from a single subject, each 2 seconds long, recorded at a sampling rate of 128 Hz. The signals were captured while the subject responded to seeing the numbers 0 through 9, providing a large dataset for analysis. The data set is large indeed but is unfortunately not ideal for this project as it records results of only one participant.

Channel Configuration The EPOC headset includes electrodes positioned at AF3, AF4, F3, F4, F7, F8, FC5, FC6, T7, T8, P7, P8, O1, and O2. Table 5.1 shows an overview of the locations of these electrodes and the function of the corresponding brain regions.

Brain Region	Electrodes	Functions
Frontal Lobe	F3, F4, F7, F8, FC5, FC6	Involved in cognitive functions such as decision making, problem-solving, and planning.
Temporal Lobe	T7, T8	Important for auditory processing and memory formation.
Parietal Lobe	P7, P8	Processes sensory information from the body, including touch, temperature, and pain.
Occipital Lobe	O1, O2	Primary center for visual processing.
Prefrontal Cortex	AF3, AF4	Associated with higher cognitive functions such as attention, emotion regulation, and executive functions.

Table 5.1: Brain regions and corresponding electrode placements in the Emotiv EPOC system.

Figure 5.2 shows the full model of the EPOC dataset, where we have colored the locations of measurements in the data set corresponding to the same brain region.

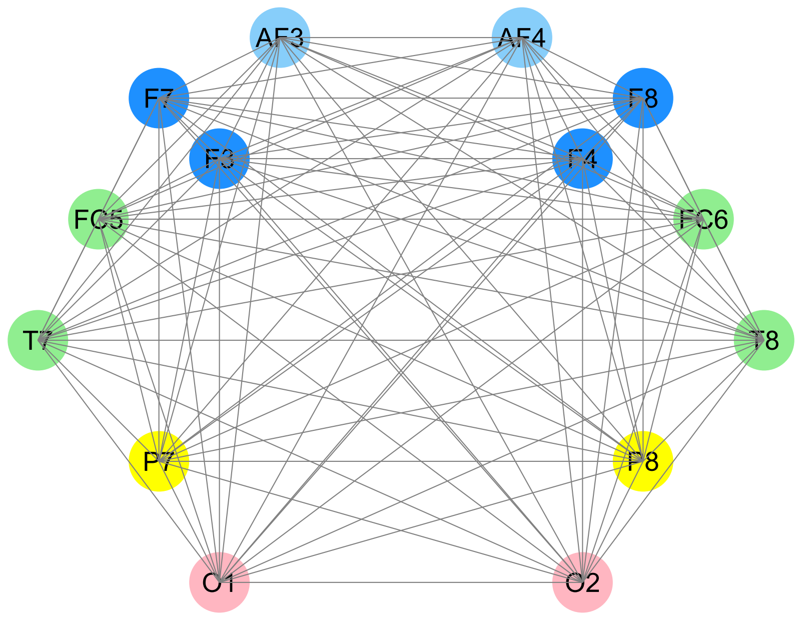


Figure 5.2: Fully Connected EEG Electrode Placement

5.1.1. Symmetries and Restrictions in EEG Data

Understanding of brain anatomy suggest certain symmetries and restrictions in EEG data.

A study (Carreiras et al., 2014) [13] investigated the activation of the brain while processing numbers, letters, or symbols. They concluded, through fMRI data, that there is a preference for number strings compared with letter strings in the right lateral occipital cortex (a part of the back of the brain involved in visual processing). Conversely, there is a preference for letters compared with numbers in the left fusiform cortex (located on the underside of the brain, involved in recognizing shapes and objects). In the early moments of processing, word processing follows a left lateralized pathway (primarily on the left side of the brain) and visual number processing follows a right lateralized pathway (primarily on the right side of the brain), including right parietal areas (located on the upper side of the brain, involved in spatial and numerical processing).

Number strings show increased activation in several brain regions, including the left inferior and middle occipital gyrus (back of the brain), the right fusiform (underside of the brain), the right inferior and middle temporal gyrus (side of the brain, involved in processing sensory input), the right inferior and superior parietal cortex (upper side of the brain), and the right supramarginal gyrus (upper side of the brain, involved in language processing). This suggests the involvement of different neural pathways in processing numbers compared to letters.

This study suggests that we could primarily find high EEG activity in the right hemisphere, with a primary focus on the occipital cortex (activity read from O2) and the parietal cortex (activity read from P8). Additionally, the middle temporal gyrus and the right supramarginal gyrus (activity read from electrode T8) are found to have high activity for number processing. The information deep within the brain cannot be easily read through our current EEG data, as we place our electrodes on top of the brain.

A different study (S. Dehaene, L. Cohen, 1995) [14] investigated the processing of digits in split-brain patients to understand how both hemispheres interact with digit processing. They concluded that a left hemispherectomy resulted in a loss of arithmetic function, while the right hemisphere was used for number identification and the ability to perceive magnitudes in numbers. A right hemispherectomy did not significantly change digit processing, as the left hemisphere was able to retain a copy of the right hemisphere's functions.

For our study, this suggests that most of our brain function related to number processing would be observed in the right hemisphere, since we are not performing any arithmetic tasks. This study agrees with the previous study.

In the next sections, we will apply GGM to the EEG dataset, exploring the potential of these models to enhance our understanding of brain activity and its applications in modern neuroscience and technology.

5.2. EEG Data Preprocessing

The total dataset comprises of 910,476 readings, resulting in 65,034 observations or events when 14 channels are taken into account. The parameters in the dataset are 'id', 'event', 'device', 'channel', 'code', 'size', and 'data'. For the purpose of this analysis, we focus on the 'event', 'channel', and 'data'.

Parameter	Explanation
id	A numeric identifier used only for reference purposes. Each entry in the dataset has a unique ID.
event	An integer used to distinguish the same event captured at different brain locations, primarily used by multichannel devices (all except MW).
device	A 2-character string identifying the device used to capture the signals. "EP" for Emotive Epoc
channel	A string identifying the 10-10 brain location of the signal. "AF3", "F7", "F3", "FC5", "T7", "P7", "O1", "O2", "P8", "T8", "FC6", "F4", "F8", "AF4"
code	An integer identifying the digit being thought/seen, with possible values 0-9, or -1 for random captured signals not related to any of the digits.
size	An integer representing the size in number of values captured in the 2 seconds of this signal. This number is 260 for EPOC.
data	A set of numbers representing the time-series amplitude of the signal. The precision of these numbers are real numbers for EP.

Table 5.2: Description of Key Parameters in the EEG Dataset

For example, one line of the EPOC device could be:

```
[id]    [event] [device] [channel] [code] [size] [data]
67650   67636   EP       F7        7      260    4482.564102,4477.435897,4484.102564…….
```

We will take the average value of the data as our measurement corresponding to the event.

5.2.1. Independence and Testing Normality

To be able to apply GGMs to this dataset we need independent observations. We will assume that each event is treated as an independent observation. This assumption is based on the experimental setup, where each event corresponds to a distinct trial or measurement. However, since all data were collected from a single subject, there may be underlying dependencies due to temporal proximity or other factors such as the subject's physiological state or experimental conditions. This assumption of independence is a simplification necessary for applying multivariate normality tests and GGMs. Future analyses could benefit from exploring potential dependencies between events.

Testing for Multivariate Normality

We will use the Mardia test [15] to test the multivariate normality of our data. Mardia's test is based on the measurement of the skewness and kurtosis of the given data. The skewness relates to how symmetric or asymmetric a distribution is, with the Gaussian distribution of course being symmetric. Figure 5.3 gives an example of a positively and negatively skewed distribution.

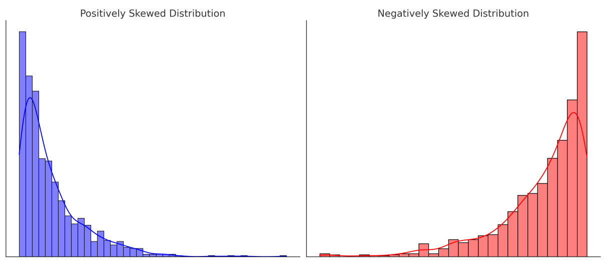


Figure 5.3: Skewness of a distribution

Kurtosis is a way to measure how heavy the tails are. A high kurtosis means that the tails are very thin, while a low kurtosis means the tails are very fat. Figure 5.4 gives an example of a high and low kurtosis.

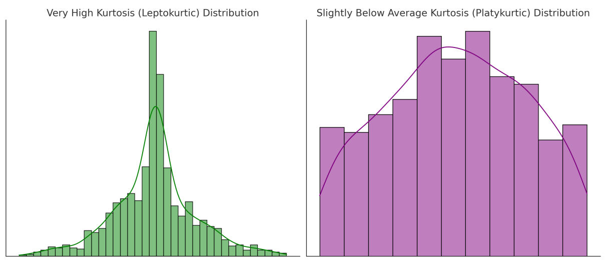


Figure 5.4: Kurtosis of a distribution

We can measure the skewness (γ_1) and kurtosis (γ_2) for k variables and n observations in the following way:

$$\gamma_1 = \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} m_{ij}^3 \text{ and } \gamma_2 = \frac{1}{n}\sum_{i=1}^{n} m_{ii}^2 \tag{5.1}$$

where $m_{ij} = (Y_i - \overline{Y})^T K (Y_j - \overline{Y})$, which is the squared Mahalanobis distance between two observations (measure to calculate distance from the mean in a multivariate setting with correlated variables available in a covariance matrix).

We can then test the null hypothesis that the data comes from a multivariate gaussian distribution against the alternative hypothesis that it is not multivariate normal. We perform two tests; one for the skewness and one for the kurtosis. The skewness statistic is $\frac{n}{6}\gamma_1$, which approximately follows a χ^2 distribution with $k(k+1)(k+2)/6$ degrees of freedom. The test statistic for the kurtosis is γ_2, which approximately follows a normal distribution with mean $k(k+2)$ and variance $8k(k+2)/n$.

Non-normality of the EEG data: When performing EEG data analysis, it is important to use a sufficiently large sample size to ensure the reliability and validity of the results. Larger sample sizes provide more accurate estimates and increase the confidence in the tests. However, in our analysis, normality tests reject the null hypothesis as the sample size becomes larger, indicating that the data most likely is not Gaussian. We will do our analysis on 1000 random samples and perform the Mardia test.

Figure 5.5 shows box plots of channels in 1000 events of EEG data. The data has been centered to have a mean of zero. While some of these channels appear to be somewhat normally distributed others are clearly not normal, as quite significant asymmetries and fat tails can be observed.

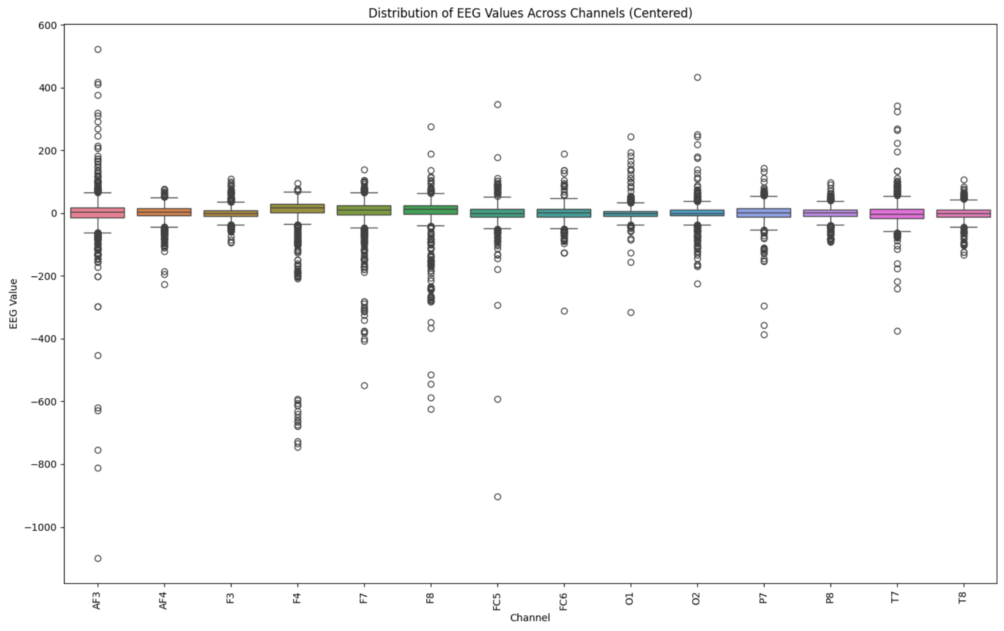


Figure 5.5: Distribution of EEG values across channels for the selected 1000 events.

A summary of all the scatter plots, marginal distributions and correlation coefficients can be found in Figure 5.6. We see that distributions of some channels do not depart a lot from the normal distribution, while the others are very skewed. The histograms of the marginal distributions of each channel are included in Appendix D.

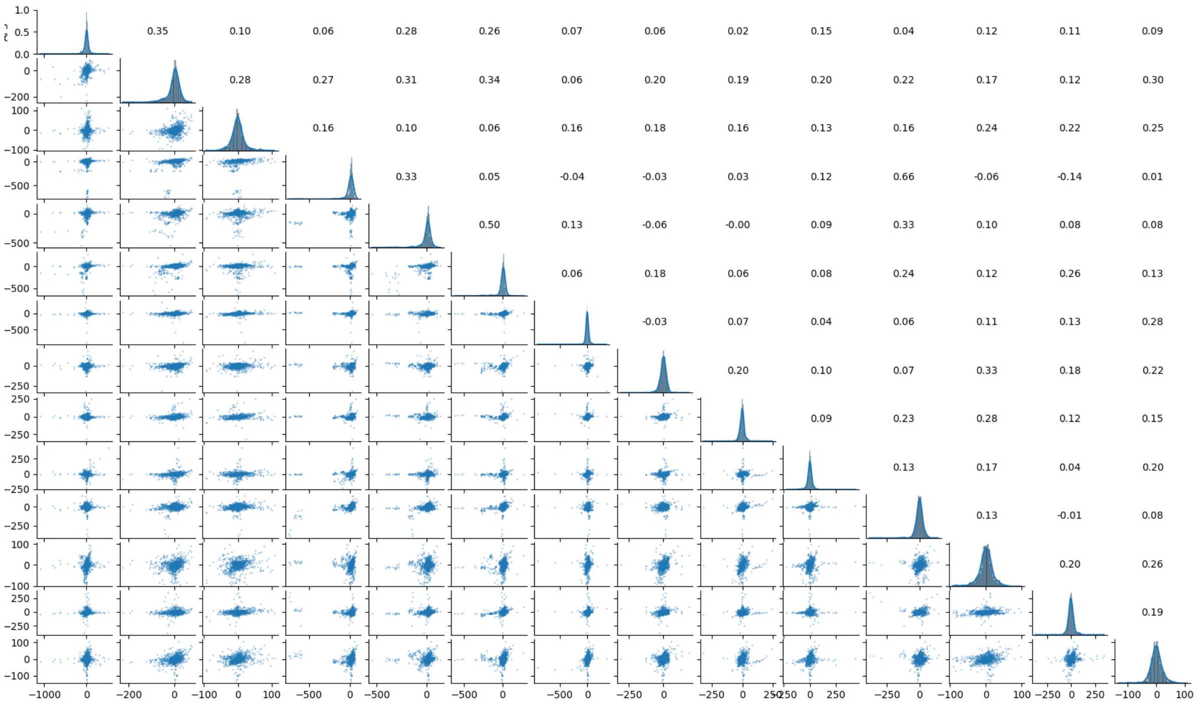


Figure 5.6: Summary of the distributions for 1000 events.

If the data was Gaussian we would see close to elliptical shapes in the scatter plots. This is clearly not the case for our current data. We will still try to test for multivariate normality with the Mardia test using the current data. The results can be found in Table 5.3.

Measure	Test Statistic	p value
Skewness	52994.437	0
Kurtosis	438.871	0

Table 5.3: Results of the Mardia test for the data.

The normality is strongly rejected in this data.

To address this, we will transform the data in a way that ensures the marginal distributions (the distributions of individual channels) are Gaussian. This transformation allows us to apply GGMs, while the dependence structure of data is maintained.

Firstly, we will compute the variances for each channel in the EEG data. Then, we will convert each margin to a uniform distribution. Following this, we will apply an inverse normal transformation to the uniformly distributed data. This transformation maps the uniform data onto a standard normal distribution, resulting in Gaussian marginal distributions.

This transformation focuses on the marginal distributions without altering the dependence structure between channels.

Figure 5.7 shows the summary of the data after transforming their marginal distributions to uniform distributions.

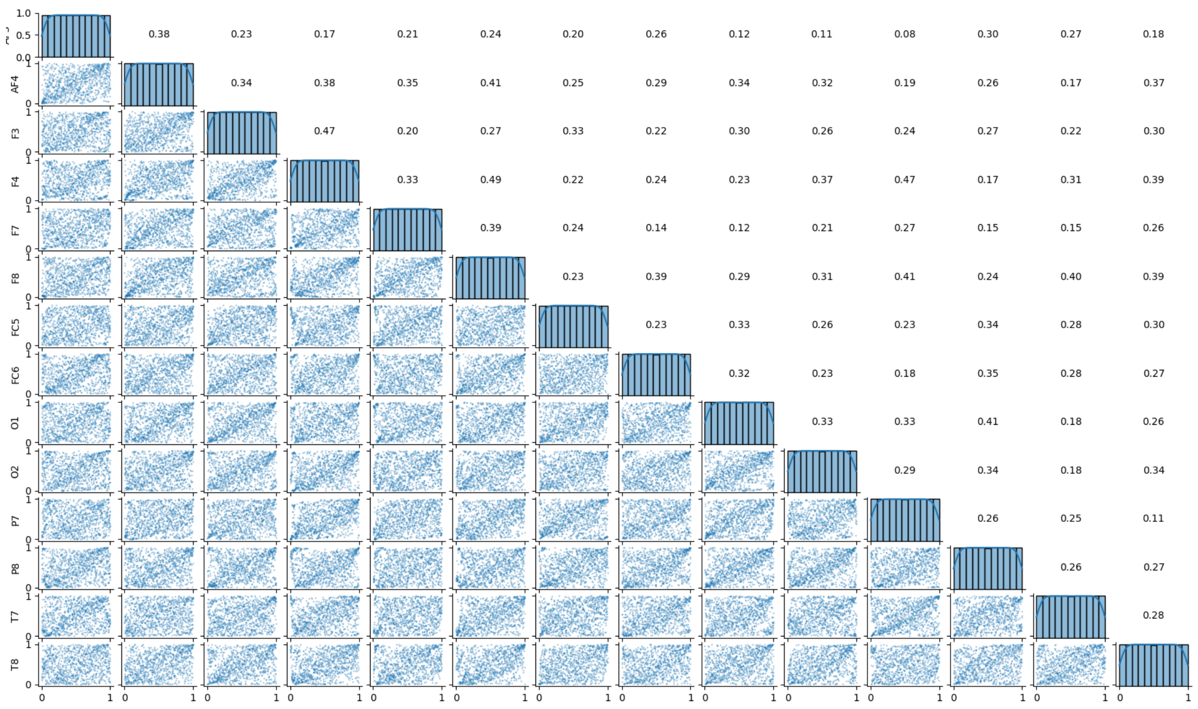


Figure 5.7: Summary of the uniform distributions for 1000 events.

The data after transforming to Gaussian margins is presented in Figure 5.8.

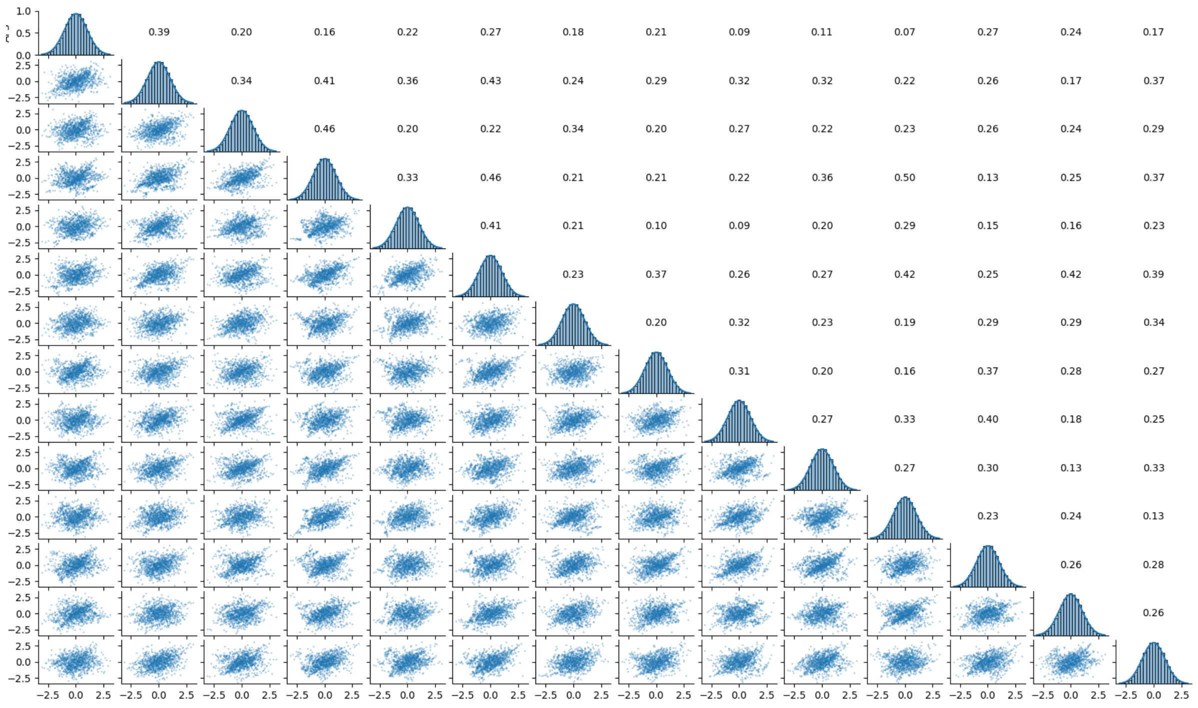


Figure 5.8: Summary of the transformed distributions for 1000 events.

The scatter plots appear more elliptical but after performing the Mardia test, the results still remain unsatisfactory as shown in Table 5.4.

Measure	Test Statistic	p value
Skewness	4047.842	0
Kurtosis	19.864	0

Table 5.4: Results of the Mardia test for the transformed data.

Our transformation has decreased the skewness and kurtosis by a lot, but has not made enough difference for us to accept multivariate normality. The further analysis concerning edge related constraints will be performed on the transformed data even though the Gaussianity is rejected.

5.3. Applying GGMs to EEG data

Let us examine the matrix containing elements of the precision matrix as well as its normalized of-diagonal elements shown in Figure 5.9. This matrix is presented in this heatmap form.

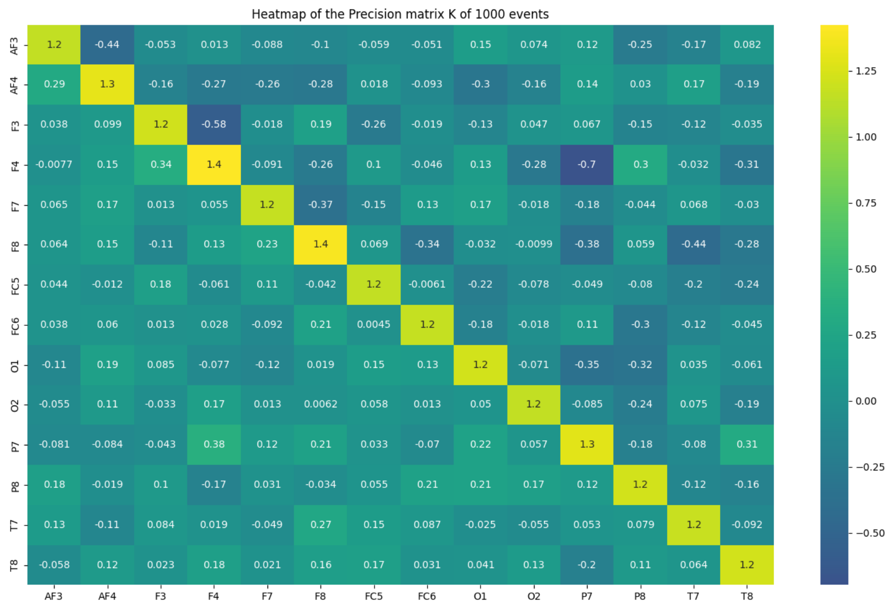


Figure 5.9: Heatmap of the precision matrix $\mathcal{K}$ with 1000 observations.

The full model has a log-likelihood of -5099.231. First we decide which edges can be deleted from the full model and then consider possible symmetry constraints.

5.3.1. Deleting Edges

Using the headlong stepwise selection model with 1000 iterations and a p-value of 0.01, the edges shown in Table 5.5 are deleted. In this table we also show the p-value of testing the model againts the full model for each step.

Different runs of the stepwise selection model result in different edges being deleted, as the order of deletion can change the values of other partial correlations. The number of deleted edges varied, and a histogram for 100 repetitions of the stepwise selection model can be seen in Figure 5.10. We chose to delete the maximum number of edges, which in this case is 35.

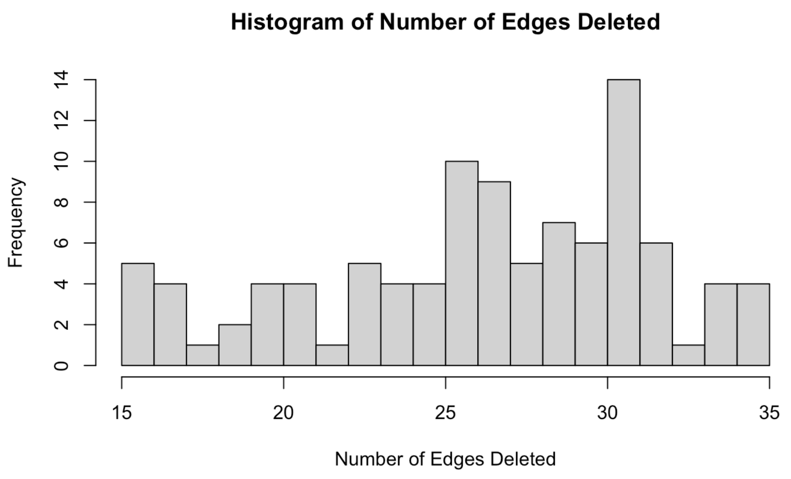


Figure 5.10: Histogram of number of deleted edges.

#	Edge Deleted	Incremental p-value	Overall p-value
1	FC6,F3	0.6699	0.6699
2	F3,AF3	0.2225	0.4343
3	F3,F7	0.6579	0.6011
4	F8,AF3	0.0555	0.2373
5	F3,T8	0.4907	0.306
6	FC5,AF3	0.1273	0.2531
7	FC5,FC6	0.7749	0.2982
8	T8,AF3	0.1971	0.2603
9	AF3,F4	0.9339	0.3444
10	FC6,T8	0.3190	0.3522
11	O1,T8	0.0891	0.2355
12	AF3,FC6	0.1154	0.1722
13	AF3,O2	0.0603	0.0963
14	AF3,F7	0.0113	0.0232
15	O2,F3	0.3050	0.0255
16	T7,T8	0.0517	0.0127
17	P7,F3	0.0971	0.0085
18	T8,F7	0.8889	0.0127
19	AF4,FC6	0.0102	0.0028
20	F4,FC6	0.0501	0.0013
21	P7,FC6	0.0137	0.0003
22	F7,F4	0.0408	0.0001
23	O1,F3	0.0210	3.8615e-5
24	AF3,P7	0.0505	1.8297e-5
25	FC6,O2	0.3082	2.1908e-5
26	F7,O2	0.4494	3.0296e-5
27	FC6,F7	0.0183	8.1800e-6
28	P8,F7	0.4475	1.1329e-5
29	O1,O2	0.0479	5.2679e-6
30	T7,F7	0.1794	4.8818e-6
31	O1,F4	0.2371	5.1445e-6
32	T7,O2	0.0610	2.7409e-6
33	P7,O2	0.0154	6.7513e-7
34	F8,O2	0.7309	1.0871e-7
35	O2,FC5	0.0718	6.3204e-7

Table 5.5: Steps of Edge Deletion with the stepwise selection model.

The graph after deleting 35 model is shown in Figure 5.11 with the heat map in Figure 5.12. The dimension of this graph (number of vertices and edges) is 70, following the deletion of 35 edges. This model has a log-likelihood of -5144.913. Performing a likelihood ratio test of this reduced model compared to full model gives a p-value of 6.32e-7. Testing of consecutive models in step-wise selection does not lead to the acceptable model for the data after the 18th edge deletion.

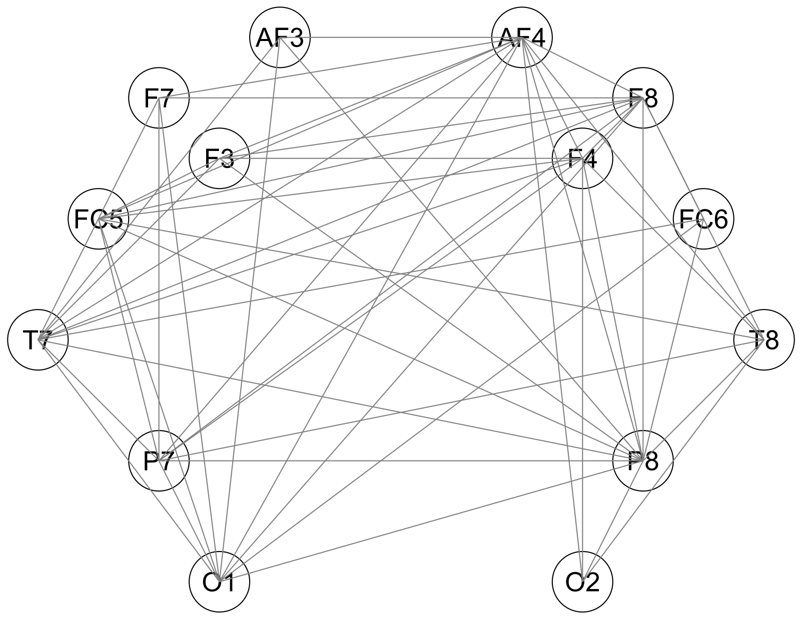


Figure 5.11: GGM representing the model with deleted edges.

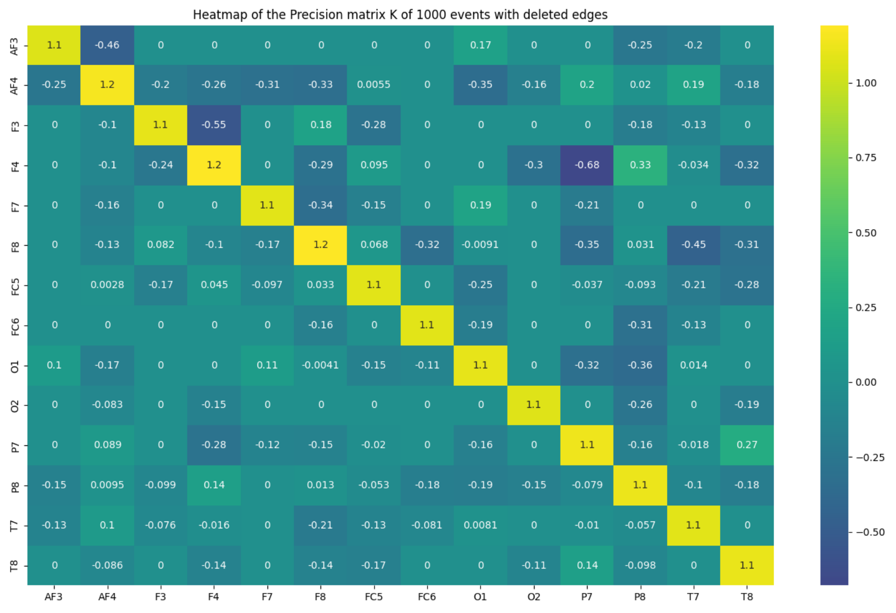


Figure 5.12: Precision matrix for the model with 35 deleted edges.

We created a Gaussian sample from the full precision matrix to check whether this problem occurred due to the non-Gaussianity of our model. However, the deleted edges found through the stepwise selection method also led to a rejection compared to the full model. So the non-Gaussianity is not the main reason causing the problem.

It is interesting to note that O2 only has connections with other channels in the right hemisphere, which aligns with our expectations. We also anticipated activity in the P8 channel, but it is surprising that this electrode is connected to all other electrodes except F7. The most significant result from the edge deletions is that most interactions in the right hemisphere have been preserved. However, the connections between O2 and FC6 and F8 are missing. Since these areas are part of the frontal lobe, it is not entirely unexpected that they are less correlated. The prefrontal cortex remains connected to O2, which could indicate a link between visual processing and the patient's attention.

5.3.2. Symmetry Constraints

Even though the stepwise deletion model didn't give us an acceptable p-value, for the sake of analysis and investigation of our EEG data, we will still perform symmetry analysis and test with respect to the model with deleted edges.

We will now apply hierarchical clustering to discover what the coloring of the graph is. The dendrogram for the vertices (the variances found in the marginal distributions of the original data) is shown in Figure 5.13. We can cluster based on variances, as making variances symmetric will result in symmetric partial variances, and the converse is also true. For the edges, the values of the negative partial correlations are shown in the dendrogram in Figure 5.14.

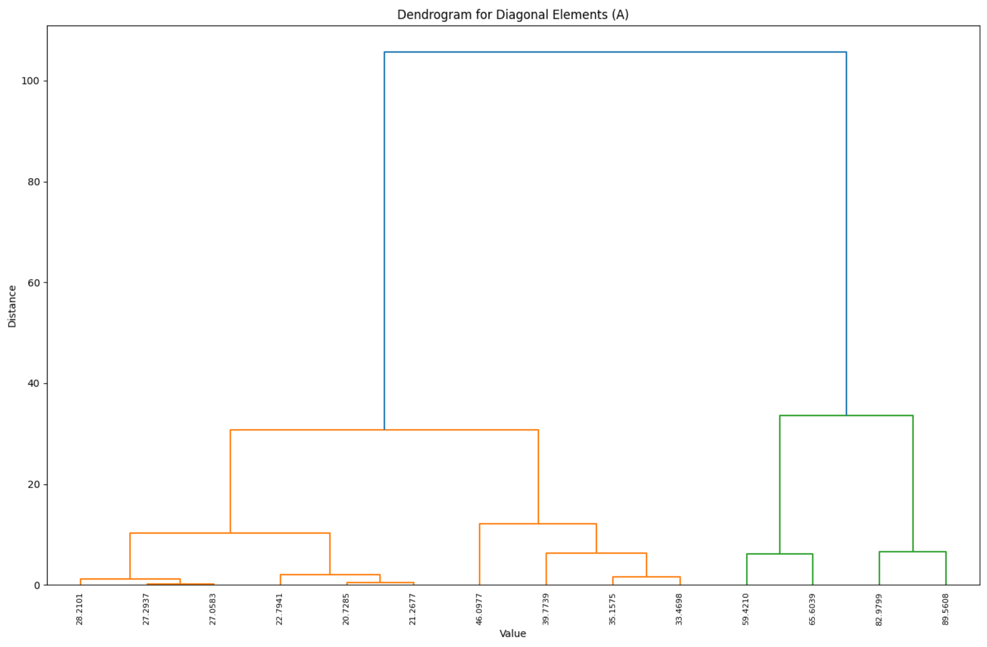


Figure 5.13: Dendrogram of the variances of the vertices.

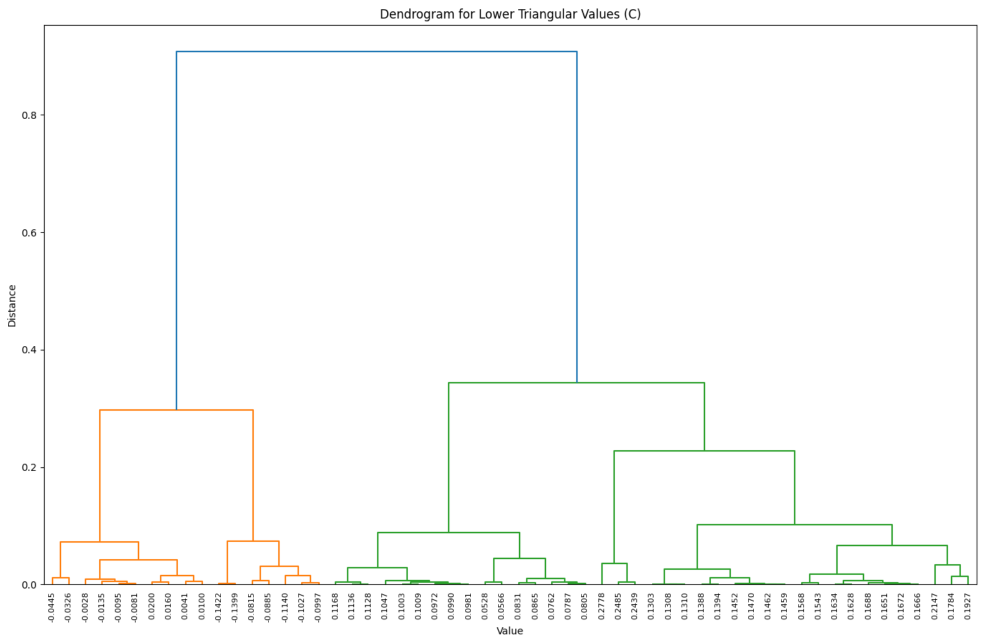


Figure 5.14: Dendrogram of the negative partial correlations.

To further analyze the hierarchical clustering results, we have summarized the effect of different threshold levels on the log-likelihood, dimension (number of VCC's and ECC's), and p-value of the Likelihood Ratio Test for the model with symmetry constraints against the model with deleted edges. These results are presented in Table 5.6. The table provides an overview of how varying the thresholds affects the results.

Threshold nodes	Threshold edges	logL	Dimension (VCC's/ECC's)	p value
1	0.05	-5168.002	22 (12/10)	0.548
1	0.07	-5169.631	21 (12/9)	0.456
5	0.04	-5173.089	21 (9/12)	0.22
2	0.04	-5172.922	22 (10/12)	0.199
1	0.08	-5179.838	19 (12/7)	0.041
5	0.05	-5184.452	19 (9/10)	0.007
1	0.09	-5188.658	18 (12/6)	0.002

Table 5.6: Results for Different Threshold Levels

By selecting appropriate thresholds, we can balance the model's complexity and its fit to the data.

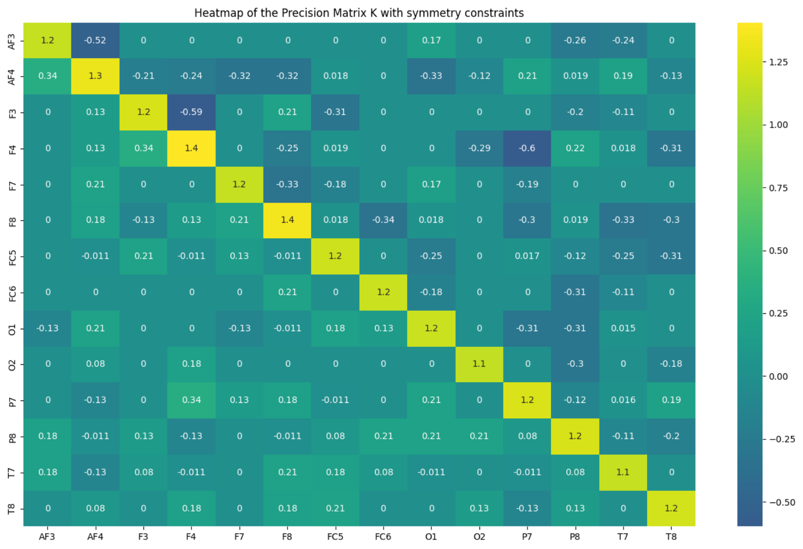
These results show the trade-off between model complexity and statistical significance. Increasing the threshold leads to a worse fit (by decreasing the log-likelihood), but does decrease the number of parameters.

If we choose the model that would not be rejected by the likelihood ratio test at the significance level $\alpha = 1\%$ the final result is a model with 12 node and 7 edge classes whose performance is included in the fifth row of Table 5.6. It should be noted that testing this model with symmetries against the full model returns a p-value of 1.6654e-6, which was to be expected given the low p-value for finding the model with deleted edges. The symmetries in this model are included in detail in Table 5.7, where we also already specify what colors will be used in the colored graph.

Cluster ID	Clusters
ECCs	
ECC 1 (light blue)	(AF4, AF3), (F4, F3), (P7, F4)
ECC 2 (light green)	(F3, AF4), (F4, AF4), (F8, F4), (FC5, F7), (O1, FC6), (P7, F7), (P8, F3), (T8, O2), (T8, P8)
ECC 3 (blue)	(F7, AF4), (F8, F7), (FC5, F3), (FC6, F8), (O1, AF4), (P7, O1), (P8, FC6), (P8, O1), (P8, O2), (T7, F8), (T8, FC5)
ECC 4 (brown)	(F8, AF4), (O1, FC5), (O2, F4), (P7, F8), (P8, AF3), (T7, AF3), (T7, FC5), (T8, F4), (T8, F8)
ECC 5 (red)	(F8, F3), (O1, AF3), (O1, F7), (P7, AF4), (P8, F4), (T7, AF4), (T8, P7)
ECC 6 (orange)	(FC5, AF4), (FC5, F4), (FC5, F8), (O1, F8), (P7, FC5), (P8, AF4), (P8, F8), (T7, F4), (T7, O1), (T7, P7)
ECC 7 (peach)	(O2, AF4), (P8, FC5), (P8, P7), (T7, F3), (T7, FC6), (T7, P8), (T8, AF4)
VCCs	
VCC 1	AF3
VCC 2	AF4
VCC 3 (red)	F3, P8
VCC 4	F4
VCC 5	F7
VCC 6	F8
VCC 7	FC5
VCC 8 (orange)	FC6, O1
VCC 9	O2
VCC 10	P7
VCC 11	T7
VCC 12	T8

Table 5.7: Grouped Elements with threshold level 1 for A and 0.08 for C.

The precision matrix with estimates for the chosen model is found in the heatmap in Figure 5.15. It is not possible to see all distinct color classes in the heatmap, since the values are close to each other. We present the colored graphical model in Figure 5.16, which allows to see the data in a more interpretable way. To make the results even easier to see, without the overlap in colors, we have created subgraphs where we took the edges of each color class separately in Figure 5.17.



	AF3	AF4	F3	F4	F7	F8	FC5	FC6	O1	O2	P7	P8	T7	T8
AF3	1.2	-0.52	0	0	0	0	0	0	0.17	0	0	-0.26	-0.24	0
AF4	0.34	1.3	-0.21	-0.24	-0.32	-0.32	0.018	0	-0.33	-0.12	0.21	0.019	0.19	-0.13
F3	0	0.13	1.2	-0.59	0	0.21	-0.31	0	0	0	0	-0.2	-0.11	0
F4	0	0.13	0.34	1.4	0	-0.25	0.019	0	0	-0.29	-0.6	0.22	0.018	-0.31
F7	0	0.21	0	0	1.2	-0.33	-0.18	0	0.17	0	-0.19	0	0	0
F8	0	0.18	-0.13	0.13	0.21	1.4	0.018	-0.34	0.018	0	-0.3	0.019	-0.33	-0.3
FC5	0	-0.011	0.21	-0.011	0.13	-0.011	1.2	0	-0.25	0	0.017	-0.12	-0.25	-0.31
FC6	0	0	0	0	0	0.21	0	1.2	-0.18	0	0	-0.31	-0.11	0
O1	-0.13	0.21	0	0	-0.13	-0.011	0.18	0.13	1.2	0	-0.31	-0.31	0.015	0
O2	0	0.08	0	0.18	0	0	0	0	0	1.1	0	-0.3	0	-0.18
P7	0	-0.13	0	0.34	0.13	0.18	-0.011	0	0.21	0	1.2	-0.12	0.016	0.19
P8	0.18	-0.011	0.13	-0.13	0	-0.011	0.08	0.21	0.21	0.21	0.08	1.2	-0.11	-0.2
T7	0.18	-0.13	0.08	-0.011	0	0.21	0.18	0.08	-0.011	0	-0.011	0.08	1.1	0
T8	0	0.08	0	0.18	0	0.18	0.21	0	0	0.13	-0.13	0.13	0	1.2

Figure 5.15: Heat Map of the precision matrix K with symmetry constraints applied.

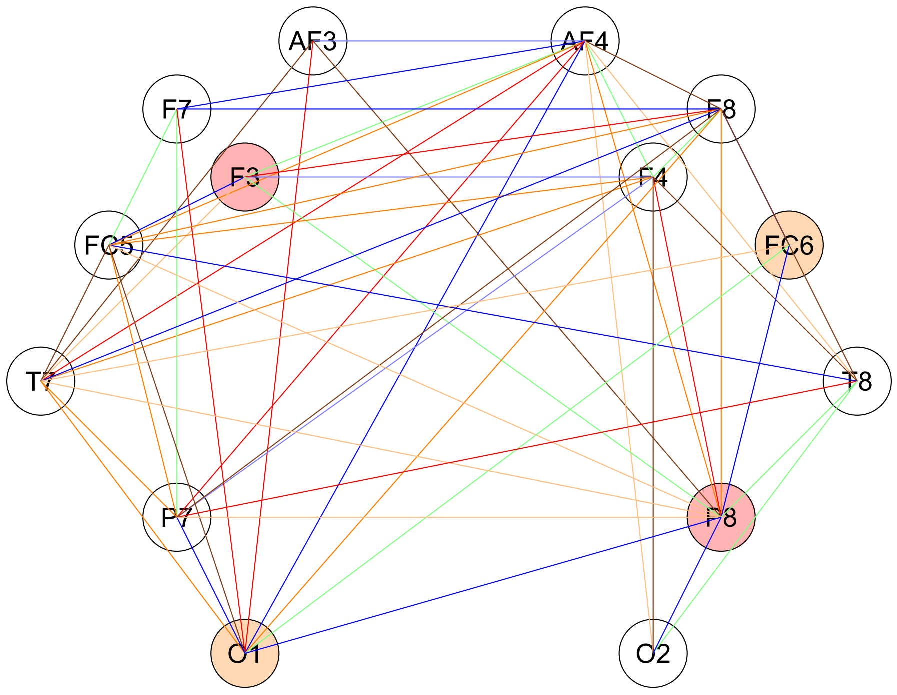
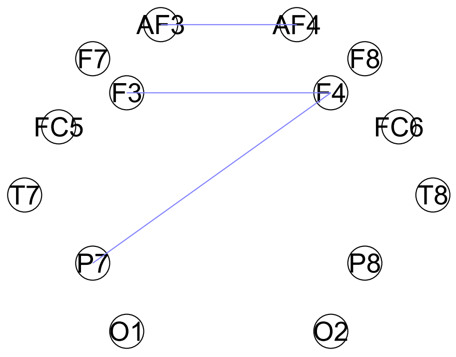
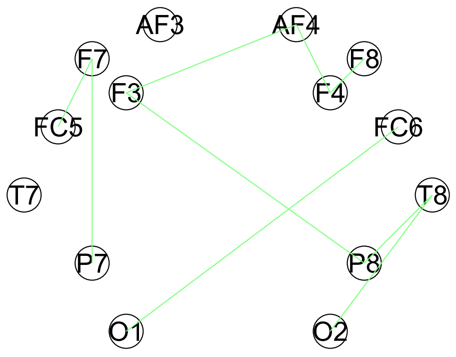


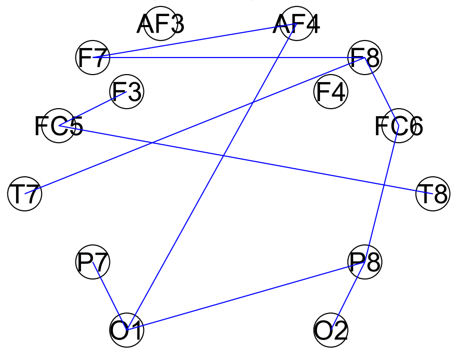
Figure 5.16: Final Graphical Model fitting our EEG Data.



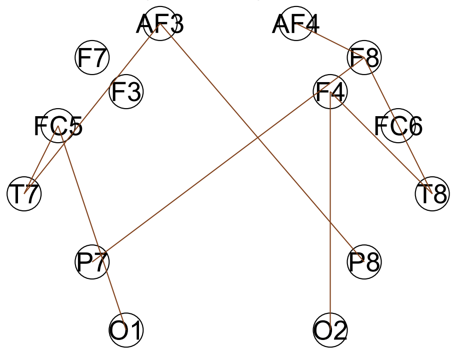
(a) Light Blue Edges (ECC 1)



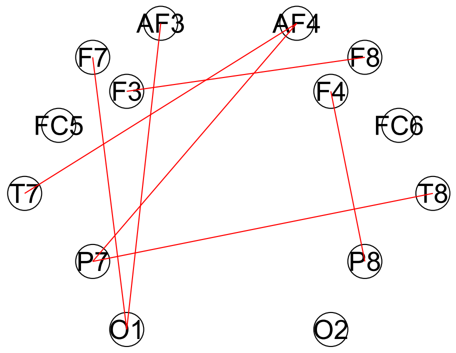
(b) Light Green Edges (ECC 2)



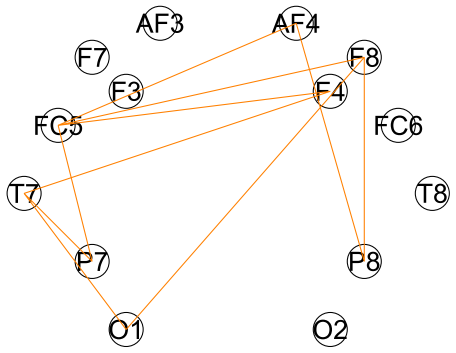
(c) Blue Edges (ECC 3)



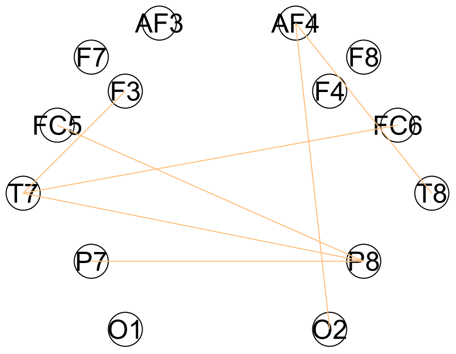
(d) Brown Edges (ECC 4)



(e) Red Edges (ECC 5)



(f) Orange Edges (ECC 6)



(g) Peach Edges (ECC 7)

Figure 5.17: Symmetry classes split per class.

Interpretation of symmetry classes

It is very encouraging that the results obtained in the performed in this thesis align with the understanding of the brain functions discussed in the previously mentioned biological studies.

In ECC 2, we observe a symmetry between O2:T8 and P8:T8, suggesting a connection in the strength of sending visual information to the memory formation and the memory formation of sensory information.

In ECC 3, we find that O2:P8, P8:FC6 and FC6:F8 have the same strength. While O2 does not directly connect to FC6 and F8, there appears to be a conditional correlation in this neural pathway, which aligns with the studies.

ECC 4 and ECC 7 also show symmetry patterns in the right hemisphere. These additional symmetries strengthen the idea that the right hemisphere has connectivity patterns that might be crucial for digit identification and processing. Further investigation into these patterns could provide insights into lateralization of brain functions and how different hemispheres contribute to various cognitive processes.

This demonstrates how GGMs can significantly impact advancements in neurology. By providing a detailed map of brain connectivity, GGMs can help in understanding the underlying mechanisms of brain function. To draw further conclusions, more background information on neurology is needed, which we do not possess. Collaborating with neurologists and neuroscientists could enhance the interpretation of these findings and lead to more comprehensive insights.

6 Conclusion

In this thesis, we have explored the use of GGMs to identify symmetries in EEG data. Our approach aimed to improve the interpretability and accuracy of GGMs by systematically incorporating symmetry constraints. This involved detecting symmetrical structures which we opted to detect through hierarchical clustering. The estimation of parameters of the GGMs with constraints were performed using Iterative Partial Maximization (IPM).

Summary of Key Findings

1. **Symmetry Detection and Simplification:** By applying hierarchical clustering to EEG data, we successfully identified symmetry classes, which allowed us to understand brain functions better. This reduced model complexity and enhanced interpretability of the model.

2. **Improved Model Performance:** The integration of symmetry constraints in GGMs resulted in improved accuracy, also when dealing with high-dimensional EEG data. This was shown with the simulation study and the real-world application to the MindBigData EEG dataset.

3. **Practical Utility in Neuroscience:** The application of these GGMs to the EEG dataset demonstrated their practical utility in neuroscience. By highlighting significant connections and dependencies between different brain regions, the models provided valuable insights into brain connectivity patterns.

4. **Robustness of IPM Implementation:** The implementation of IPM for parameter estimation proved crucial in optimizing the log-likelihood with the given edge deletions and symmetry constraints. This iterative approach ensured that the models remained accurate and reliable.

The findings from this thesis show the importance of incorporating symmetry constraints in GGMs to enhance their interpretability and accuracy. The methodology developed provides a robust framework for analyzing high-dimensional EEG data, offering significant potential for advancements in neuroscience and medical research.

Discussion

Future research should focus on exploring alternative statistical techniques that do not rely on normality assumptions, addressing the limitations of non-Gaussian EEG data. Additionally, further refinement of the hierarchical clustering scheme to test multiple thresholds at once will enhance the practicality of this method, leading to more accurate models capable of providing deeper insights into brain connectivity.

Another discussion could be made for the methods of deleting edges, as we have only discussed the stepwise selection model. A shortcoming of the stepwise selection model is that the update steps use a different optimization scheme and calculate the p-value for each step in the process, neglecting to compare the created model to the original model. This problem did not occur too much in our simulation study, but the EEG data analysis performed poorly in the overall testing. Other possibilities could create better results, such as implementing edge deletion in the hierarchical clustering process so that zeros are not treated as a separate step in the modeling process.

There is still room for improvement in setting up GGMs, as the software has not yet been fully optimized for ease of use in terms of integration between packages.

Moreover, the integration of GGMs with other imaging techniques, such as fMRI, could provide a more extensive view of brain activity. Combining structural and functional data would allow for a better understanding of how connections in the GGM translate into neural processes. This interdisciplinary approach could introduce new diagnostic tools and personalized treatment strategies in neurology.

In conclusion, this thesis has shown a systematic approach to detect and utilize symmetries in GGMs using gRc and hierarchical clustering. The application of these methods to EEG data has demonstrated their practical utility, paving the way for future innovations in both theoretical and applied neuroscience. By improving the interpretability and accuracy of GGMs, this work contributes to the ongoing efforts to understand and map brain connectivity.

References

[1] Søren Højsgaard and Steffen Lauritzen. “Graphical Gaussian Models with Edge and Vertex Symmetries”. In: *Journal of the Royal Statistical Society: Series B* 70.5 (2008), pp. 1005–1027. DOI: 10.1111/j.1467-9868.2008.00666.x.

[2] Jianqing Fan and Runze Li. *Statistical Challenges with High Dimensionality: Feature Selection in Knowledge Discovery*. 2006. arXiv: math/0602133 [math.ST].

[3] A. Goncalves, P. Ray, and B. et al. Soper. “Generation and evaluation of synthetic patient data”. In: *BMC Medical Research Methodology* 20.108 (2020). DOI: 10.1186/s12874-020-00977-1.

[4] U. Gather, M. Imhoff, and Fried. “Graphical models for multivariate time series from intensive care monitoring.” In: *Statistics in Medicine* 21.18 (2002), pp. 2685–2701. DOI: 10.1002/sim.1209.

[5] Mark Bodner et al. “Detecting Symmetric Patterns in EEG Data: A New Method of Analysis”. In: *Clinical Electroencephalography* 30.4 (1999), pp. 143–150. DOI: 10.1177/155005949903000406.

[6] Søren Højsgaard, David Edwards, and Steffen Lauritzen. *Graphical Models with R*. Springer, 2012.

[7] Søren Højsgaard and Steffen Lauritzen. “Inference in Graphical Gaussian Models with Edge and Vertex Symmetries with the gRc Package for R”. In: *Journal of Statistical Software* 23.6 (2007), pp. 1–26. DOI: 10.18637/jss.v023.i06.

[8] J. Whittaker. *Graphical models in applied multivariate statistics*. John Wiley & Sons, 1990.

[9] Lauritzen. *Graphical Models*. Clarendon Press, 1996.

[10] M Drton and M Eichler. “Maximum Likelihood Estimation in Gaussian Chain Graph Models under the Alternative Markov Property”. In: *Scandinavian Journal of Statistics* 33.2 (2006), pp. 247–257. DOI: 10.1111/j.1467-9469.2006.00482.x.

[11] S.C. Johnson. “Hierarchical clustering schemes.” In: *Psychometrika* 32 (1967), pp. 241–254. DOI: 10.1007/BF02289588.

[12] F. Nielsen. *Hierarchical Clustering. In: Introduction to HPC with MPI for Data Science. Undergraduate Topics in Computer Science.* Springer, 2016. DOI: 10.1007/978-3-319-21903-5_8.

[13] Manuel Carreiras et al. “Orthographic Coding: Brain Activation for Letters, Symbols, and Digits”. In: *Cerebral Cortex* 25.12 (2015), pp. 4748–4760. DOI: 10.1093/cercor/bhu163.

[14] L. Cohen S. Dehaene. “Towards an anatomical and functional model of number processing”. In: *Mathematical cognition* (1995). URL: http://www.unicog.org/publications/DehaeneCohen_TripleCodeModelNumberProcessing_MathCognition1995.pdf.

[15] K. V. Mardia. “Measures of Multivariate Skewness and Kurtosis with Applications.” In: *Biometrika* 57 (1970), pp. 519–530. DOI: 10.1093/biomet/57.3.519.

A
Calculations of the Univiariate Gaussian Distribution

Consider n samples of i.i.d. random variables $X_1, X_2, \dots, X_n$ with each $X_i \sim N(\mu, \sigma^2)$, with μ being the mean and σ the standard deviation.

The probability density function for the normal distribution is given by:

$$f_X(x_i) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_i-\mu)^2}{2\sigma^2}\right) \tag{A.1}$$

The univariate Gaussian distribution has the shape of a bell curve as can be seen in an example in Figure A.1

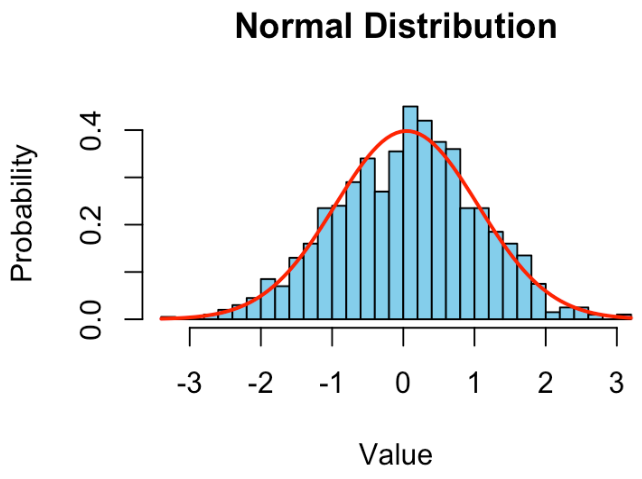


Figure A.1: An example of the Gaussian distribution for $\mu = 0$ and $\sigma = 1$

A.1. MLE of the univariate Gaussian distribution

Consider random variables $X_1, X_2, \dots, X_n$ that are independently and identically distributed, with each $X_i \sim N(\mu, \sigma^2)$ and the probability density function as in equation A.1.

To find the maximum likelihood estimators (MLEs) for μ and σ^2, we start by defining the likelihood function for the observed sample:

$$L(\mu, \sigma^2; x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_i - \mu)^2}{2\sigma^2}\right) \tag{A.2}$$

The log-likelihood function simplifies the calculations:

$$\begin{aligned} \log L(\mu, \sigma^2; x_1, x_2, \ldots, x_n) &= \sum_{i=1}^{n} \log\left(\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_i - \mu)^2}{2\sigma^2}\right)\right) \\ &= -\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i - \mu)^2 \\ &= -\frac{n}{2}\log(2\pi) - \frac{n}{2}\log(\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i - \mu)^2 \end{aligned} \tag{A.3}$$

To find the MLEs, we take the partial derivatives with respect to μ and σ^2 and set them to zero. For μ we get:

$$\begin{aligned} \frac{\partial}{\partial\mu}\left(-\frac{n}{2}log(2\pi) - \frac{n}{2}log(\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i - \mu)^2\right) &= 0 \\ \frac{1}{\sigma^2}\sum_{i=1}^{n}(x_i - \mu) &= 0 \\ \sum_{i=1}^{n}(x_i - \mu) &= 0 \\ \sum_{i=1}^{n}x_i - n\mu = 0 \\ \hat{\mu} = \frac{1}{n}\sum_{i=1}^{n}x_i \end{aligned} \tag{A.4}$$

For σ^2 we get:

$$\begin{aligned} \frac{\partial}{\partial\sigma^2}\left(-\frac{n}{2}log(2\pi) - \frac{n}{2}log(\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i - \mu)^2\right) &= 0 \\ -\frac{n}{2\sigma^2} + \frac{1}{2(\sigma^2)^2}\sum_{i=1}^{n}(x_i - \mu)^2 &= 0 \\ \frac{n}{2\sigma^2} &= \frac{1}{2(\sigma^2)^2}\sum_{i=1}^{n}(x_i - \mu)^2 \\ \hat{\sigma^2} &= \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2 \end{aligned} \tag{A.5}$$

So to conclude we have the maximum likelihood estimators for μ and σ^2:

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n}x_i$$

$$\hat{\sigma^2} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{\mu})^2$$

B

Calculations for solving the likelihood updates

B.1. Derivation of the Likelihood equations

The likelihood equations for the RCOR model implemented in gRc are derived from the log-likelihood function for a Gaussian graphical model. The log-likelihood function is given by:

$$\log L = \frac{n}{2} \log \det(K) - \frac{1}{2}\mathrm{tr}(KW)$$

where $K = ACA$ is the concentration matrix, W is the empirical sum of squares and products matrix. Here, A is a diagonal matrix with elements $a_\alpha = \sqrt{k_{\alpha\alpha}}$ and C is the correlation matrix with ones on the diagonal and negative partial correlations off the diagonal.

The likelihood equations are formed by taking the derivative of the log-likelihood function with respect to the parameters δ (off-diagonal elements) and η (diagonal elements).

B.1.1. Edge classes

Taking the derivative with respect to δ_u:

$$\frac{\partial \log L}{\partial \delta_u} = \frac{n}{2} \frac{\partial \log \det(K)}{\partial \delta_u} - \frac{1}{2} \frac{\partial \mathrm{tr}(KW)}{\partial \delta_u}$$

Since K is parameterized by δ_u:

$$\frac{\partial \log \det(K)}{\partial \delta_u} = \mathrm{tr}(K^{-1} \frac{\partial K}{\partial \delta_u})$$

$$\frac{\partial \mathrm{tr}(KW)}{\partial \delta_u} = \mathrm{tr}(W \frac{\partial K}{\partial \delta_u})$$

For δ_u, we have:

$$\frac{\partial K}{\partial \delta_u} = AT_uA$$

Thus, the first derivative becomes:

$$\frac{\partial \log L}{\partial \delta_u} = \frac{n}{2}\mathrm{tr}(\Sigma AT_uA) - \frac{1}{2}\mathrm{tr}(WAT_uA)$$

where $\Sigma = K^{-1}$.

To find the maximum likelihood estimates, we set the first derivative to zero:

$$\frac{n}{2}\mathrm{tr}(\Sigma AT_uA) - \frac{1}{2}\mathrm{tr}(WAT_uA) = 0$$

Solving for tr$(\Sigma A T_u A)$:

$$\text{tr}(\Sigma A T_u A) = \frac{\text{tr}(W A T_u A)}{n}$$

Where we can simplify:

$$\begin{aligned}\text{tr}(\Sigma A T_u A) &= \text{tr}(A^{-1} C^{-1} A^{-1} A T_u A)\\ &= \text{tr}(A A^{-1} C^{-1} T_u)\\ &= \text{tr}(T_u C^{-1})\end{aligned}$$

Edge Color Classes $u \in E$: So for edge color classes, we have:

$$\text{tr}(T_u A W A) = n\,\text{tr}(T_u C^{-1})$$

where T_u is the matrix indicating the edge color class u.

B.1.2. Vertex Classes

First define $\lambda_u = \log(\eta_u)$. Taking the derivative with respect to λ_u:

$$\frac{\partial \log L}{\partial \lambda_u} = \frac{n}{2}\frac{\partial \log \det(K)}{\partial \lambda_u} - \frac{1}{2}\frac{\partial \text{tr}(KW)}{\partial \lambda_u}$$

Since K is parameterized by λ_u:

$$\frac{\partial \log \det(K)}{\partial \lambda_u} = \text{tr}(K^{-1}\frac{\partial K}{\partial \lambda_u})$$
$$\frac{\partial \text{tr}(KW)}{\partial \lambda_u} = \text{tr}(W\frac{\partial K}{\partial \lambda_u})$$

For λ_u, we have using the product rule and $\frac{\partial A}{\partial \lambda_u} = T_u A$:

$$\frac{\partial K}{\partial \lambda_u} = T_u ACA + ACAT_u$$

Substituting the derivative of K gives:

$$\frac{\partial \log \det(K)}{\partial \lambda_u} = \text{tr}(K^{-1}(T_u ACA + ACAT_u))$$
$$\frac{\partial \text{tr}(KW)}{\partial \lambda_u} = \text{tr}(W(T_u ACA + ACAT_u))$$

Using $K^{-1} = \Sigma$:

$$\frac{\partial \log \det(K)}{\partial \lambda_u} = \text{tr}(\Sigma T_u ACA) + \text{tr}(\Sigma ACAT_u)$$
$$\frac{\partial \text{tr}(KW)}{\partial \lambda_u} = \text{tr}(W T_u ACA) + \text{tr}(W ACAT_u)$$

Since tr$(AB) =$ tr(BA), we can rewrite these as:

$$\frac{\partial \log \det(K)}{\partial \lambda_u} = 2\,\text{tr}(\Sigma T_u ACA)$$

$$\frac{\partial \text{tr}(KW)}{\partial \lambda_u} = 2\,\text{tr}(W T_u ACA)$$

To find the maximum likelihood estimates, we set the first derivative to zero:

$$\frac{n}{2}\cdot 2\,\text{tr}(\Sigma T_u ACA) - \frac{1}{2}\cdot 2\,\text{tr}(W T_u ACA) = 0$$

Simplifying:

$$n\,\text{tr}(\Sigma T_u ACA) - \text{tr}(WT_u ACA) = 0$$

Solving for tr$(\Sigma T_u ACA)$:

$$\text{tr}(\Sigma T_u ACA) = \frac{\text{tr}(WT_u ACA)}{n}$$

Since $\Sigma = K^{-1} = (ACA)^{-1} = A^{-1}C^{-1}A^{-1}$, we have:

$$\text{tr}(\Sigma T_u ACA) = \text{tr}(A^{-1}C^{-1}A^{-1}T_u ACA) = \text{tr}(T_u)$$

Thus, we get:

$$\text{tr}(T_u) = \frac{\text{tr}(T_u CAWA)}{n}$$

Vertex Color Classes $u \in V$: So for vertex color classes, we have:

$$\text{tr}(T_u CAWA) = n\,\text{tr}(T_u)$$

where T_u is the matrix indicating the vertex color class u.

B.2. Solving the Likelihood Equations

To solve these equations, we use an iterative partial maximization algorithm. The algorithm alternates between updating the parameters δ and η.

B.2.1. Updating δ_u

We solve δ_u using the Newton method, which is defined as:

$$\delta_u \leftarrow \delta_u - \frac{\partial L/\partial \delta_u}{\partial^2 L/\partial \delta_u^2 + \zeta}$$

, where ζ is a regularization parameter that is to be defined.

We know that $\partial L/\partial \delta_u = n/2\text{tr}(T_u C^{-1}) - 1/2\text{tr}(T_u AWA)$, which we simplify to be the difference in the expected trace and observed trace, which is called the discrepancy Δ_u:

$$\text{Expected Trace} = \text{tr}(T_u C^{-1})$$

$$\text{Observed Trace} = \frac{\text{tr}(T_u AWA)}{n}$$

$$\Delta_u = \text{tr}(T_u C^{-1}) - \frac{\text{tr}(T_u AWA)}{n}$$

And $\partial^2 L/\partial \delta_u^2 = -\text{tr}(T^u C^{-1} T^u C^{-1})$.

If we take the regularization term to be $\Delta_u^2/2$, we find our Newton iteration for δ_u:

$$\delta_u \leftarrow \delta_u + \frac{\Delta_u}{\text{tr}(T^u C^{-1} T^u C^{-1}) + \frac{\Delta_u^2}{2}}$$

with

$$\Delta_u = \text{tr}(T_u C^{-1}) - \frac{\text{tr}(T_u AWA)}{n}$$

B.2.2. Updating η_u

We solve η_u by rewriting it as a quadratic form.

First define the Hadamard product of C and W, $Q_{\alpha\beta} = C_{\alpha\beta}W_{\alpha\beta}$.

Now we have:

$$\begin{aligned} n\text{tr}(T_u) &= \text{tr}(T_u CAWA) \\ &= \sum_{\alpha\in u}\sum_{\beta\in V} a_\alpha C_{\alpha\beta} W_{\alpha\beta} a_\beta \\ &= \sum_{\alpha\in u}\sum_{\beta\in V} Q_{\alpha\beta} a_\alpha a_\beta \end{aligned}$$

Now we make a difference in observing $\beta \in u$ or $\beta \notin u$ and also that $a_\alpha = \eta_u \forall \alpha \in u$.

$$\begin{aligned} &= \sum_{\alpha\in u}\sum_{\beta\in u} Q_{\alpha\beta} a_\alpha a_\beta + \sum_{\alpha\in u}\sum_{\beta\notin u} Q_{\alpha\beta} a_\alpha a_\beta \\ &= \sum_{\alpha\in u}\sum_{\beta\in u} Q_{\alpha\beta} \eta_u^2 + \sum_{\alpha\in u}\sum_{\beta\notin u} Q_{\alpha\beta} \eta_u a_\beta \end{aligned}$$

For simplicity define:

$$D = \sum_{\alpha\in u}\sum_{\beta\in u} Q_{\alpha\beta} \text{ , } B = \sum_{\alpha\in u}\sum_{\beta\notin u} Q_{\alpha\beta} a_\beta \text{ , } n\text{tr}(T_u) = n|u|$$

Using these simplifications gives the quadratic equation:

$$\eta_u^2 D + \eta_u B = n|u|$$

, which has as a positive solution:

$$\eta_u = \frac{-B + \sqrt{B^2 - 4f|u|D}}{2D}$$

C
Code

The scripts used for the results obtained in this paper can be found at the following URL: https://github.com/DanielWolff2001/BachelorThesis_DWolff.

D

Histograms of distributions of the channels

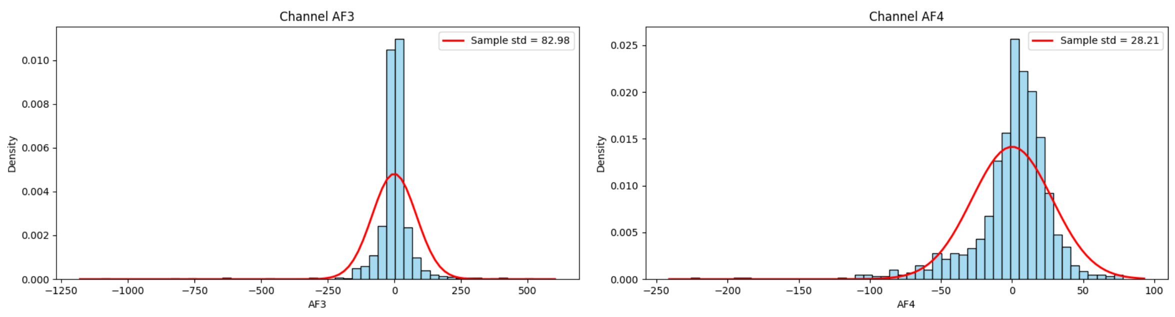


Figure D.1: Histogram of Channels AF3 and AF4

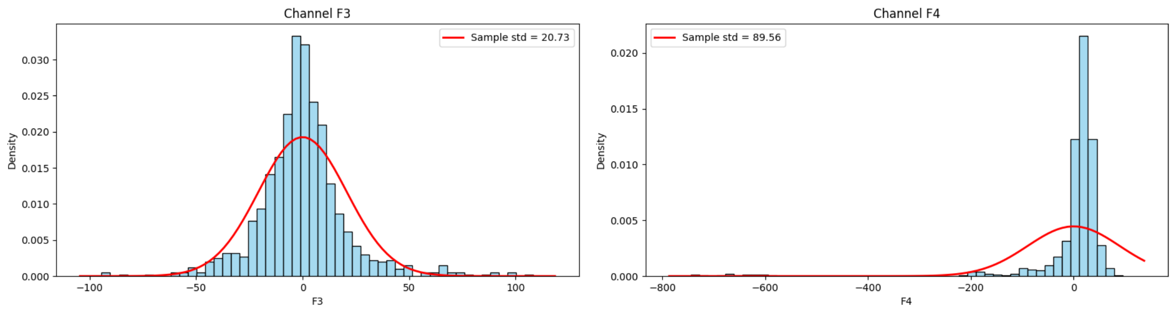


Figure D.2: Histogram of Channels F3 and F4

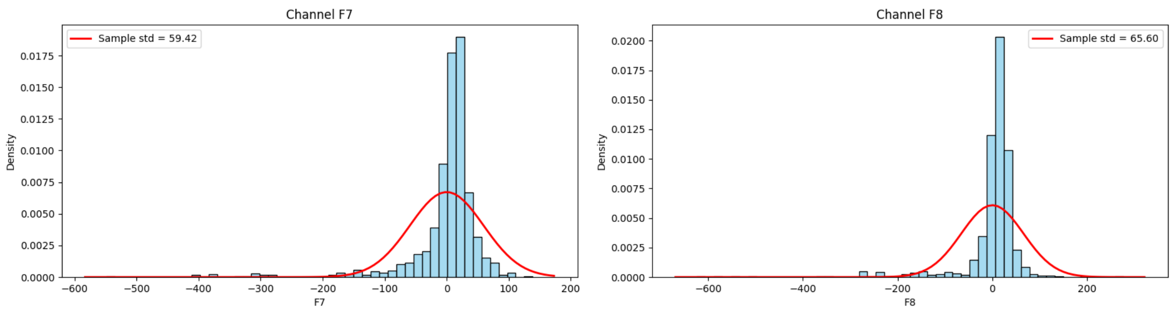


Figure D.3: Histogram of Channels F7 and F8

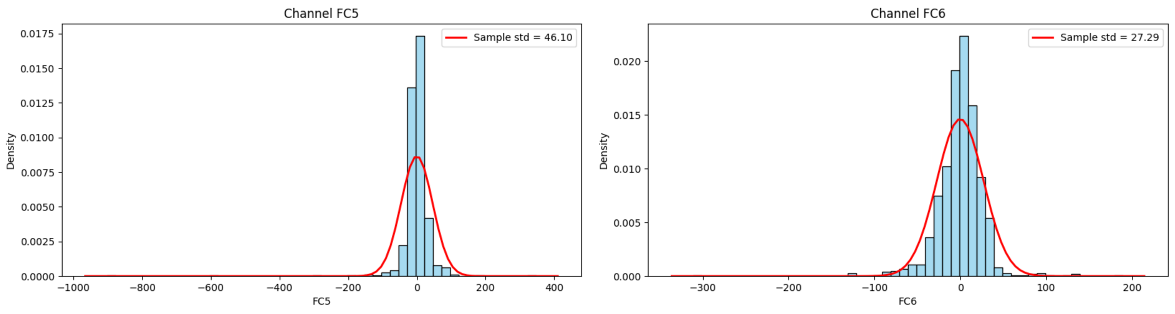


Figure D.4: Histogram of Channels FC5 and FC6

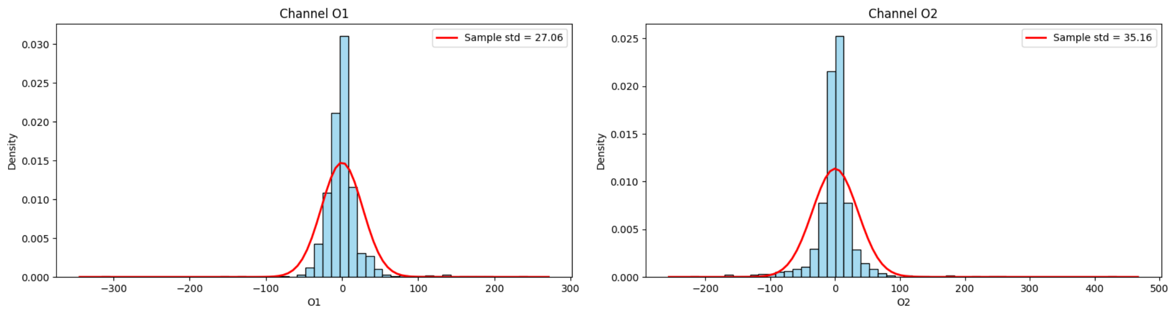


Figure D.5: Histogram of Channels O1 and O2

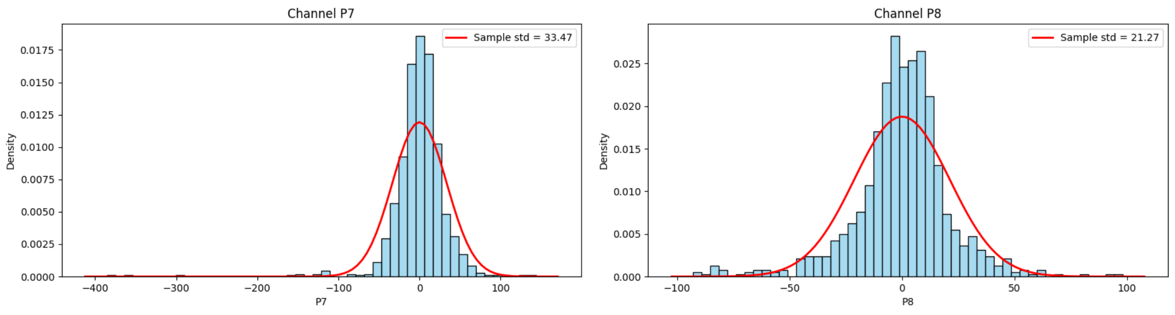


Figure D.6: Histogram of Channels P7 and P8

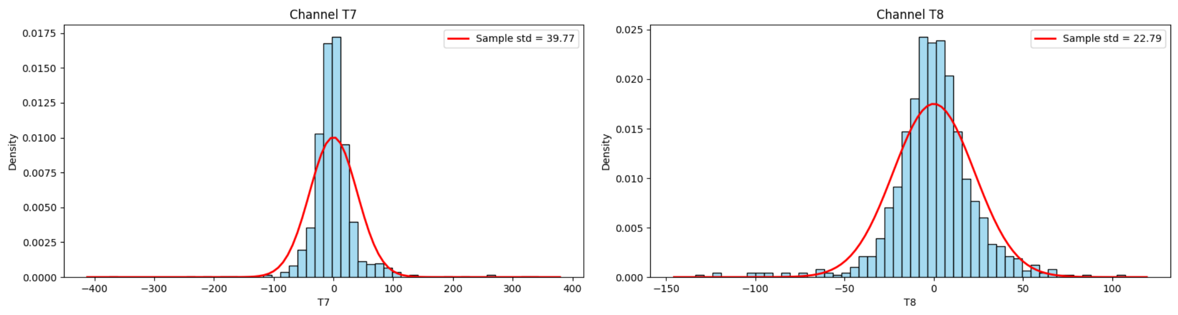


Figure D.7: Histogram of Channels T7 and T8