Computational aspects of differential networks

Ricardo Marques U16077131

WST 895 Mini-Dissertation

Submitted in partial fulfillment of the degree MSc (Advanced Data Analytics)

Supervisor: Prof. M. Arashi

Co - Supervisor: Prof. A. Bekker

Department of Statistics, University of Pretoria

December 15, 2021

Abstract

The need for statistical tools capable of addressing high-dimensional data is ever-growing. One such tool is that of differential networks, which have become increasing popular within various branches of science. The popularity of differential networks and their subsequent analysis is largely attributed to their ability to effectively represent the relationships between factors of complex systems over time, or over various experimental conditions. However, a differential network is not easily calculated, and in high dimensional settings common within biological sciences they must be estimated. Motivated by this, this dissertation comprehensively explores differential networks and the efficient estimation thereof through the use of a R package developed throughout the course of this research - dineR.

Declaration

I, *Ricardo Daniel Marques Salgado,* declare that this essay, submitted in partial fulfillment of the degree *MSc (Advanced Data Analytics),* at the University of Pretoria, is my own work and has not been previously submitted at this or any other tertiary institution.

_____

Ricardo Daniel Marques Salgado

_____

Prof. M. Arashi

_____

Prof. A. Bekker

December 15, 2021

_____

Date

Acknowledgements

The financial assistance of the National Research Foundation (NRF) towards this research is hereby acknowledged (SRUG190308422768 Grant Number 120839). Opinions expressed and conclusions arrived at, are those of the author and are not necessarily to be attributed to the NRF. Similarly, all data shown and used for the purposes of this dissertation are freely available (NAS124/2019).

Glossary, Abbreviations and Acronyms

ADMM Alternating direction method of multipliers. 21, 43, 46–49, 51, 53, 56, 66, 76

AIC Akaike information criterion. 57

BIC Bayesian information criterion. 57, 58

big data Extremely large or complex datasets, whose size or complexity prohibits the use of traditional statistical tools for analysis. 8, 10

block coordinate descent An optimization algorithm that finds the minimum of a function iteratively, by minimizing it in one coordinate plane at a time. 26, 31

CLIME Constrained ℓ_1 - minimization for inverse matrix estimation. 28

dineR Differential Network Estimation package in R developed during this research and available on CRAN. 2, 53, 54, 56–59, 62, 66, 68, 75, 76

EBIC Extended Bayesian information criterion. 57, 58, 60, 72

epidemiology The branch of medical sciences responsible for studying disease occurrence, progression and spread. 11

genomic The study of the genomes - the genetic material that makes up a organism or cell. 11, 16, 30

GGM Gaussian graphical model. 15, 16, 24, 51

high-dimensional Datasets in which the number of features recorded exceeds the number of observations present. That is $p > n$. 2, 11, 14, 16, 17, 22, 23, 25–27, 31, 34, 37, 42, 43, 49, 57–59

LASSO Least absolute shrinkage and selection operator. 20, 24–31, 48–51, 61, 62, 64, 72

MCP Minimax concave penalty. 20, 29, 30, 48, 56, 61, 62, 64, 66, 68

nonparanormal Distributions that can be represented as a semi-parametric Gaussian copula. 15, 32

SCAD Smoothly clipped absolute deviation. 20, 22, 25, 28–30, 48, 56, 61, 62, 64, 65

sparsity The proportion of elements within a matrix that are zero. That is the number of zero-valued elements over the total number of elements within the matrix. 12, 24, 29, 31, 33, 41, 43

TPR True positive rate. 59, 61, 62

ultra high-dimensional Datasets in which the number of features recorded *greatly* exceeds the number of observations present. That is $p >> n$. 29, 43, 47, 57, 59, 72, 75

Contents

List of Figures

List of Tables

1 Introduction

Data is a drastically growing resource, with more data having been generated in the last two years alone than in all of human history preceding that [29]. As the volume of data available explodes, so too has the average size of datasets. This is seen as data scientists were surveyed on the largest dataset they processed annually, with the median response growing from approximately 6 gigabytes in 2006 to as large as 30 gigabytes in 2015 [55]. The increasing frequency that these exceptionally large datasets are encountered has lead to the development and popularisation, as seen in Figure 1 [73], of the term big data.

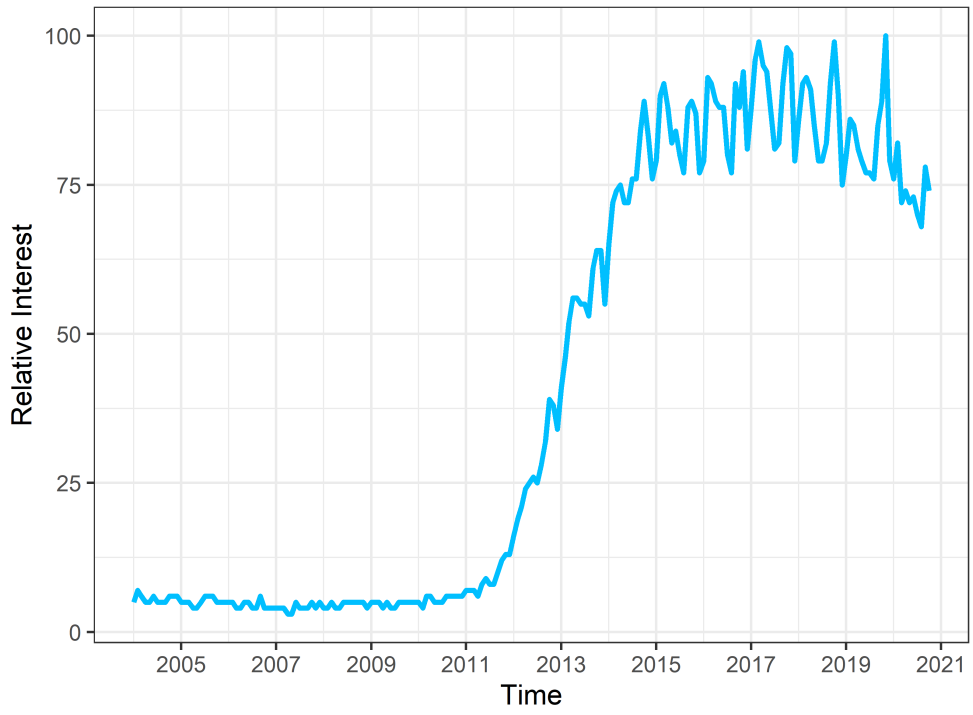

Figure 1: Google search trends on big data

Big data refers to any and all data, whose size or complexity prohibits the use of traditional statistical techniques to perform analyses. This prohibition has resulted in the development of several statistical methods, designed with the ability to specifically address such complexities. One such method is differential networks. Before further details on differential networks can be introduced, graphical modelling preliminaries are required. Reason being is graphical models form the foundation of all visualisation and interpretation of differential networks, as will be shown later [68].

1.1 Graphical Modelling Preliminaries

Graphical models, often referred to as networks in literature, are considered the union of probability theory and graph theory [51]. By design, graphical models aim to capture and represent the interactions between the various components of complex systems [68]. This ability, as well as the ability to clearly

illustrate the behaviour of each of the system's components in response to changes in external stimuli has seen the popularity of network theory grow exponentially in many sciences as of late [7]. Sciences in which network theory has seen this growth include social sciences, medical and biological sciences, physics as well as epidemiology [7].

Formally, a graphical model is defined as $G = \{V, E\}$ [39]. Where V is referred to as the set of nodes or vertices of the graph, which represent the variables in the given problem [51]. While, E is the edge set of the graph that consists of pairs of nodes, also called arcs or links, who characterise the interactions between nodes [68]. An introductory graphical model can be seen in the below figure:

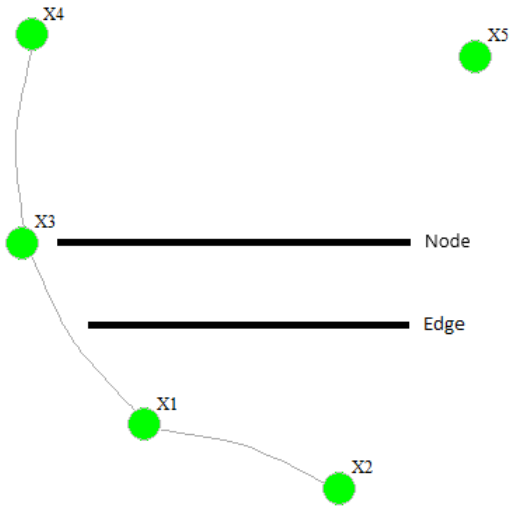

Figure 2: Introductory graphical model

The above graphical model, Figure 2, is derived from the following sets: $nodes = \{X_1, X_2, X_3, X_4, X_5\}$ and $edges = (\{X_1, X_2\}, \{X_1, X_3\}, \{X_3, X_4\})$. One main motivation for the use of graphical models, such as the above, is the ability to observe changes in any complex system represented by a graph, G, by observing changes in the its nodes, V, or its edges, E, or both [65]. Most systems such as those within computational biology, climate studies, genomics, finance and medical imaging, all of which are often high-dimensional, only undergo changes within their edge set [92]. That is, changes occur within the interactions of the system's components over time or in reaction to changes in environmental factors [92]. Changes in the node set, although less common, are still possible such as those frequently observed in communication and social networks, where the number of components within the network grow overtime [7, 65]. For the purposes of this dissertation, it will be assumed that the node set of any network shown here is fixed, and only the edges will be expected to undergo changes. This is a reasonable assumption, as all of the practical applications will be performed on clinical data in which the node set should almost never change [31]. Another structural difference that may be observed between graphical models is the

form of the edge joining any two nodes. There are two forms which edges may undertake, that is edges may be directed or undirected [40]. If edges are undirected, which is the most frequently observed case, any single edge is simply a pair of nodes, which indicate there is a non-zero association between the two variables represented by the nodes [68]. However, there are two main ways this association between variables can be determined. To illustrate this, consider a graph $G = \{V, E\}$ where $V = \{1, 2, ..., n\}$ represents the set of nodes and E describes the edges between said nodes, all related to a set of random variables $X_1, X_2, ..., X_n$. The first manner in which the edges may be determined is through the use of marginal inference procedures. That is, an edge is said to exist between variable X_i and variable X_j if and only if the variables are dependent on one another [65]. Mathematically, nodes i and j are connected if and only if $\rho(X_i, X_j) \neq 0$. There are however, several different means to examine ρ and thus whether a marginal dependence between any two nodes exists. One possible measure is the Pearson correlation coefficient. In this case, the network for the random variables corresponds to a set of non-diagonal components of the sample correlation matrix, $\mathbf{R}$ [65]. These components are selected through the use of hypothesis tests, with $H_0 : \rho(X_i, X_j) = 0$. Alternatively, the set of significant correlations can be obtained by the use of a thresholding parameter, λ, whereby all the correlations larger than λ are deemed significant [65]. λ here is a tuning parameter that allows the user to control the level of sparsity within the network, something which becomes of great importance later when estimating the network. One severe disadvantage of making use of the marginal inference procedures, despite their simplicity is that marginal inference procedures cannot differentiate between relationships which are direct from those which are indirect [65]. A simple example can illustrate this shortfall. Consider the variables $X_1, ..., X_5$ all of which follow a normal distribution. Assume the true underlying network G is categorised by three edges as in Figure 2. Thus, the edges are known and are namely $\{X_1 - X_2, X_1 - X_3, X_3 - X_4\}$. Secondly, assume $\rho(X_1, X_2) = 0.8 = \rho(X_1, X_3)$. However, this second assumption then implies that $\rho(X_2, X_3) = 0.64$, that in turn implies that the graph has an additional edge in the form $X_2 - X_3$, which is incorrect [65]. Thus, to address the above issue, an alternative to the marginal inference procedures is required.

Conditional association measures aim to address the above limitation by making use of a slightly different description as to that provided above. Under conditional inference procedures, an undirected edge exists between two nodes i and j if and only if the variable X_i and variable X_j are conditionally dependent, given all other variables [65]. This approach, although more computationally taxing than the marginal one, has been shown to provide results which may be far more scientifically valuable [65]. The use of conditional dependencies can also be shown to have addressed the issue introduced in the example above, as the partial correlation between variable X_2 and variable X_3 given variable X_1 is zero. Having shown that making use of conditional associations addresses the main issue arising from the use of marginal inference as previously described, there is another advantage from utilising the conditional

approach. That being there exist many well developed non-parametric approaches to obtain the network when considering conditional dependencies, however for the purposes of this dissertation it is important to note that these approaches all perform inadequately when the system consists of a large number of nodes [65]. Moreover, these techniques do not extend easily to a high-dimensional setting, which as will be discussed later, is the focus of the work here [65]. Taking into account the above, and the fact that parametric graphical models extend far easier to a high-dimensional setting, the focus for the remainder of this document is on that of parametric graphical models.

Thus far, only undirected graphs have been considered however there exists another popular sub-category of graphical models. In this case, edges are directed and consist of two components namely a parent node and a child node which describes the direction of the edge between the nodes [68]. Statistically, a directed edge only exists between two variables if there is a causal relationship between said variables [68]. The parent node is then representative of the cause variable, and similarly the child node represents the effect variable [68]. In summary, most graphical models can then be considered to fall within one of the following sub-categories:

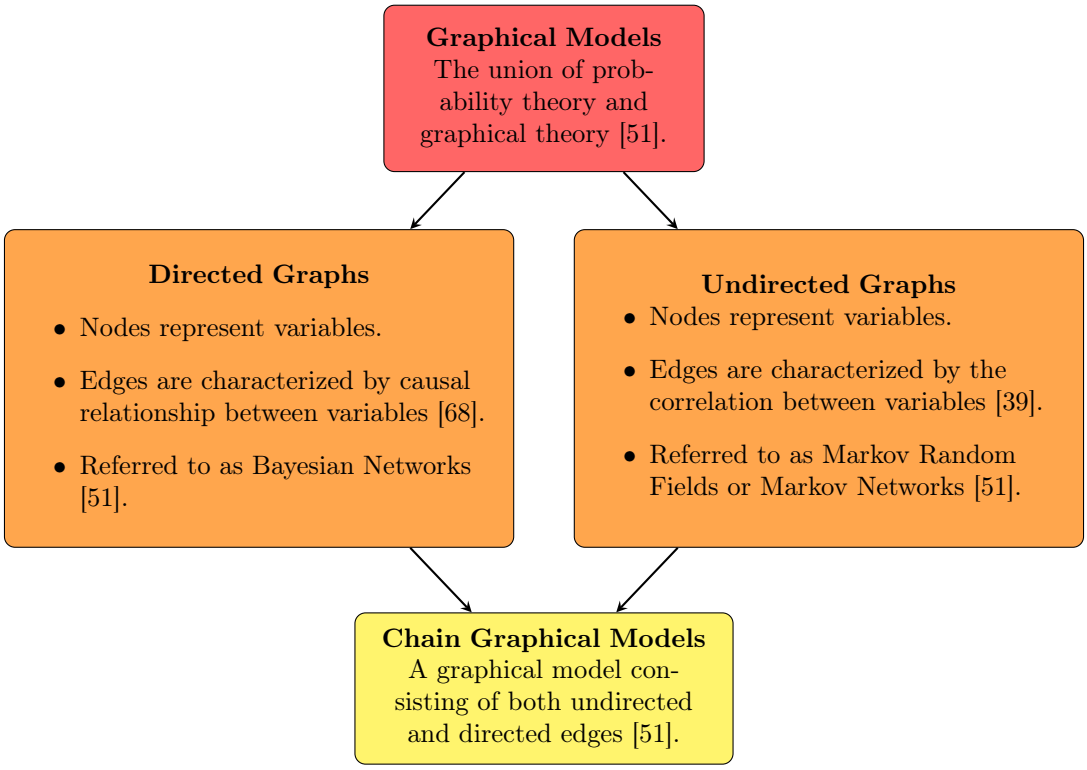

Figure 3: Overview of graphical models

For the purposes of this research, edges will assumed to be undirected and thus the work going forward is centered on undirected graphical networks for three main reasons. Firstly, the available literature and subsequent applications of directed networks thereof is by far in the minority when compared with their

undirected counterparts [65]. Secondly, the determination of causal relationships between variables in high-dimensional settings is extremely computationally taxing. This makes the application of directed networks very rarely feasible in the event whereby the number of variables outnumbers the number of observations significantly which will be explored within the application section of this dissertation. Thirdly, the purpose of this dissertation is to examine differential networks and as such undirected graphs which are components therein.

In Figure 3, the types of graphical models listed is not exhaustive and several other structures for graphical models exist, such as factor graphs, directed cyclic graphs, directed acyclic graphs, forest graphs, moral graphs, join trees, clique trees, junctions trees and geographical networks [68]. To illustrate the power of graphical models as a whole, consider Figure 4 and Figure 5 which represent a geographical network and the corresponding map of Liverpool City that illustrates the number of new weekly reported COVID-19 cases during the 41st week of lockdown in England.

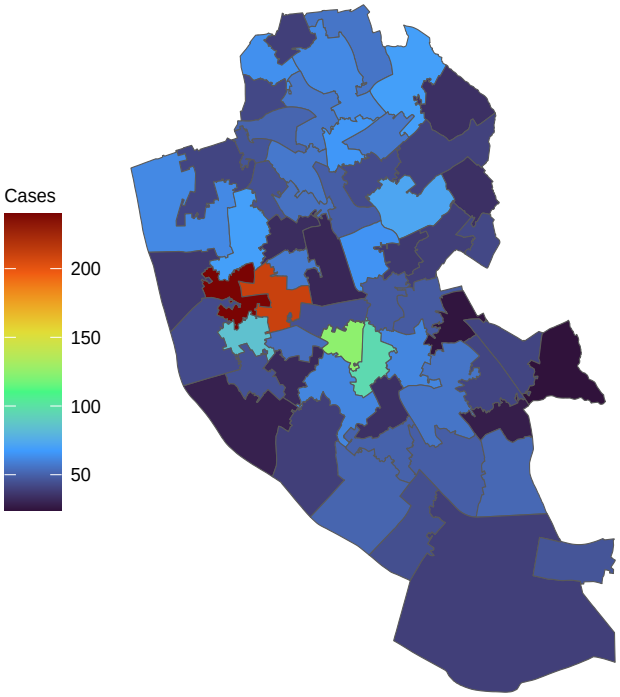

Figure 4: COVID-19 cases by municipality

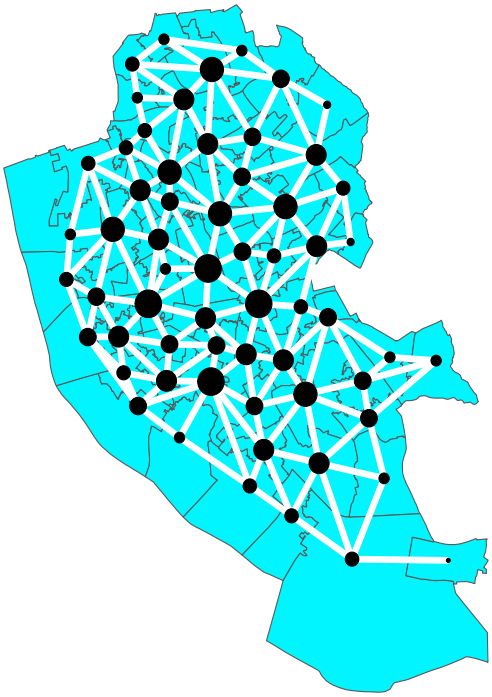

Figure 5: Geographical network of municipalities

Figures such as the above provide researchers and lawmakers with valuable information regarding the spread of COVID-19 and the subsequent areas that were most at risk given an outbreak in a particular region. Researchers could then anticipate a high number of cases in regions who are highly connected to the region currently experiencing an outbreak. Geographical networks such as that shown above can be greatly improved to include epidemiological factors, and even transportation networks that can then also be used to predict the disease transmission allowing for proactive measures, such as the region

lockdown implemented in Leicester City by the English government in June 2020. Despite the usefulness of geographical networks, there is however one such special case of graphical models that is the most extensively studied and applied approach within literature [43]. The special case is that of the Gaussian graphical model.

Gaussian graphical models, GGMs, follow the same general framework as all graphical models, that is the graph is defined as $G = \{V, E\}$ with V the set of nodes, and E the edges joining the nodes [43]. The defining trait of GGMs is that it is assumed the underlying distribution of the variables, captured as nodes, is normal [43]. The assumption of normality although a rather stringent one, and one that is known not to hold in almost all real-world scenarios, will be later shown to be easily relaxed through the use of several nonparanormal transformations that will allow the capturing of non-normal data as a Gaussian graphical model without loss of generality [42, 43]. As with any assumption, it is important to understand the motivation behind assuming normality here. The motivation is that if the objective is to make use of a p-dimensional vector $\mathbf{x}$, with $\mathbf{x} \sim N_p(\underline{\mu}, \mathbf{\Sigma})$, and obtain the underlying graphical model encoded within the data then the complete graphical model is described by the inverse covariance matrix of $\mathbf{x}$. The use of GGMs has become common practice in many fields such as speech processing, image analysis, bioinformatics, gene regulation and macroeconomics [17, 39, 90, 92]. The reason for GGMs popularity within such fields has to do with the scale of problems regularly observed within these fields. That is, problems within these disciplines routinely involve thousands or even millions of random variables that are all related to one another [39]. These relationships between variables are more often than not extremely complex, and as such cannot intuitively be visualized through the use of traditional statistical approaches [39]. GGMs however provide a robust framework for addressing such scenarios as shown below.

Figure 6: Heatmap of a correlation matrix

Figure 6 visualises a heatmap of the 10×10 correlation matrix shown in Table 9 which is a figure commonly examined within literature. However, despite there being only 10 variables present within the above example, identifying the various relationships that exist is not a trivial task and grows exponentially in complexity as the number of variables considered increases.

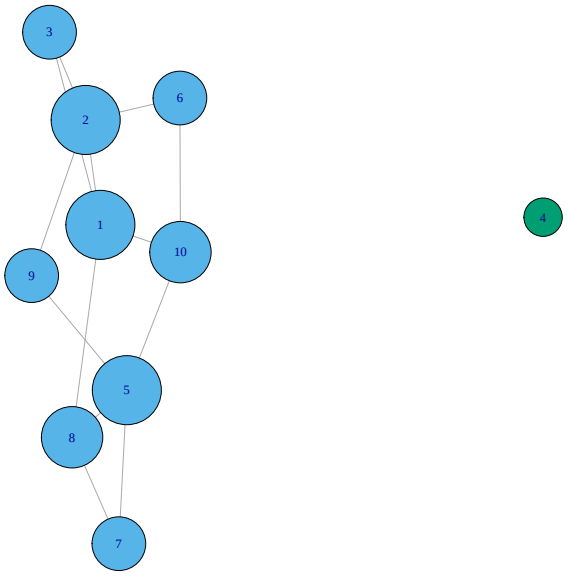

Figure 7: Graphical model representation of correlation matrix

Figure 7, however depicts the graphical model obtained from Figure 6, and when inspected the interactions between variables becomes far clearer. This clearly illustrates the significant benefit of applying GGMs, specifically in a high-dimensional setting. As such, while also in an attempt to remain consistent with existing practical applications in biological sciences, it will be assumed that the underlying distribution of any set of variables examined here is normal [31]. This will make the findings of the work here applicable to a wide variety of biological settings. Lastly, until this point networks have only been mentioned at a single point in time, or at a particular state but it is often of interest to examine a complex system over time, or in response to external stimuli [37]. To extend the theory introduced thus far, an adaption is to compare networks over time or networks under various states such as pre-treatment against post-treatment which gives rise to differential networks [65].

1.2 Differential Networks

In sciences such as genomics, or medical sciences it is not of considerable interest to examine any single network as the relationships that exist between components are rather well documented [65]. Within

these sciences, and several other fields it is of greater interest to examine differences between networks [65]. Differences between networks can be a single network observed at different points in time, or the same network observed in response to different experimental conditions [17]. One example of this is the comparison of a network of medical measurements for individuals before a cancer treatment to a network of medical measurements for individuals during cancer treatment. Such a comparison provides insight into cancer progression and the response to the selected treatment [65]. Thus, making use of the assumptions provided in Section 1.1, a differential network is defined as the difference between two inverse covariance matrices, also referred to as precision matrices [69]. Mathematically, a differential network is represented as follows:

$$\boldsymbol{\Delta} = \boldsymbol{\Sigma}_1^{-1} - \boldsymbol{\Sigma}_2^{-1}, \tag{1}$$

where $\boldsymbol{\Sigma_1}$ and $\boldsymbol{\Sigma_2}$ are the covariance matrices of experimental state 1 and 2 respectively. The motivation behind calculating $\boldsymbol{\Delta}$ is to be able to capture and hopefully explain any changes between the conditional correlations across the two states [69]. One such use case whereby differential networks and their ensuing analysis is commonly deployed is genetics [94]. An example of this is whereby gene expression levels are represented as vertices in a graphical model whose edges are characterized as the conditional dependency relationships between genes [94]. Within gene expression level studies it is common however to have this data available under various conditions. It is then possible to derive a differential network, $\boldsymbol{\Delta}$, for any combination of two of the conditions present. Any element of $\boldsymbol{\Delta}$ is then the difference between the partial covariance for a pair of genes across the prescribed conditions [94]. In the above example, allowance was only made to consider two conditions, however this is not a necessary requirement to derive a valid differential network. If the aim is to study gene expression levels over K different conditions, with $K > 2$, one possible way to address this would be to determine all possible pairwise differential networks [94]. This approach, although valid, would be both time consuming and computationally taxing. Thus, the possibly more favourable approach would be to evaluate the pairwise difference between each condition's relevant precision matrix and a common precision matrix, such as a pooled covariance matrix across all K conditions [94].

At first glance, it appears rather simple to obtain a differential network, but there is one key challenge. That is, the determination of differential networks relies upon two inverse covariance matrices [69]. These inverse covariance matrices become increasingly problematic to determine as the number of covariates considered increases [69]. In a high-dimensional setting, whereby the number of variables, p, is larger than the number of observations, n, these covariance matrices become singular and as such their inverses do not exist [5]. However, in many of the disciplines mentioned thus far, the data is frequently high-dimensional [65]. It is then vitally important to possess tools to overcome the singularity of the covariance matrices.

Currently the most popular techniques that allow the issue of non-invertible covariance matrices to be circumvented are to either estimate the differential network as a whole, or the network components, precision matrices, individually [69]. Depending on the approach taken from those above, the steps to estimate $\mathbf{\Delta}$ differ greatly, but so too do the assumptions regarding the individual precision matrices and indeed the differential network itself. Hence, it is important to consider the purpose of the results one hopes to obtain and decide upon an appropriate approach with the relevant assumptions attached.

1.3 Optimisation and Estimation

As it will be later shown within this dissertation, all of the processes currently available for differential network estimation consist of two broad steps. That is, in order to estimate a differential network, literature states an appropriate loss function must first be derived or selected [3, 69, 90, 94]. The target of this loss function can then either be the differential network as a whole, or the individual precision matrices that make up the differential network [94, 90, 69], with the differences between these two approaches further explored in Section 1.4. Once a loss function has been selected, an applicable optimization scheme based on the characteristics of the loss function can then be implemented to optimize the loss function and arrive at the differential network estimate. Both of these two steps are common throughout many branches of physics, mathematics and statistics. However, before providing specific details regarding each of these steps in the context of this dissertation, it is important to consider loss functions and optimization in a universal setting.

Consider the common transportation problem as defined by Pedregal [59]. That is, a company has an in-demand product which they must distribute globally from m of their manufacturing plants to n different distributions points. The quantities shipped from each manufacturing plant can be represented mathematically as $s_1, s_2, ..., s_m$, with the amounts to be received at each center being $r_1, r_2, ..., r_n$. The cost of transportation for a single unit of product from plant j to the distribution center i is known in advance as c_{ji}. It then becomes of great interest to the company to determine the quantity of product that can be sent between destination j and i while maintaining transportation costs that are as low as possible [59].

To solve the above problem, it is first represented mathematically. That is, if the quantity of product transported from plant j to point i is represented by x_{ji} then the optimization problem is clearly to minimize the following: $\sum_{j,i} c_{ji}x_{ji}$. However, there are several restrictions on the amount of product that may be shipped. Namely, any one manufacturing plant has a finite amount of product available to ship. Thus the following inequality must hold: $\sum_i x_{ji} = s_j$. Similarly, any distribution point only has a finite amount of demand, and as such a second inequality must hold: $\sum_j x_{ji} = r_i$. One final restriction exists, that is no quantity shipped may be negative and as such $x_{ji} \geq 0$ must hold. This mathematical

problem, as represented in Figure 8, is then one which is facing many companies globally.

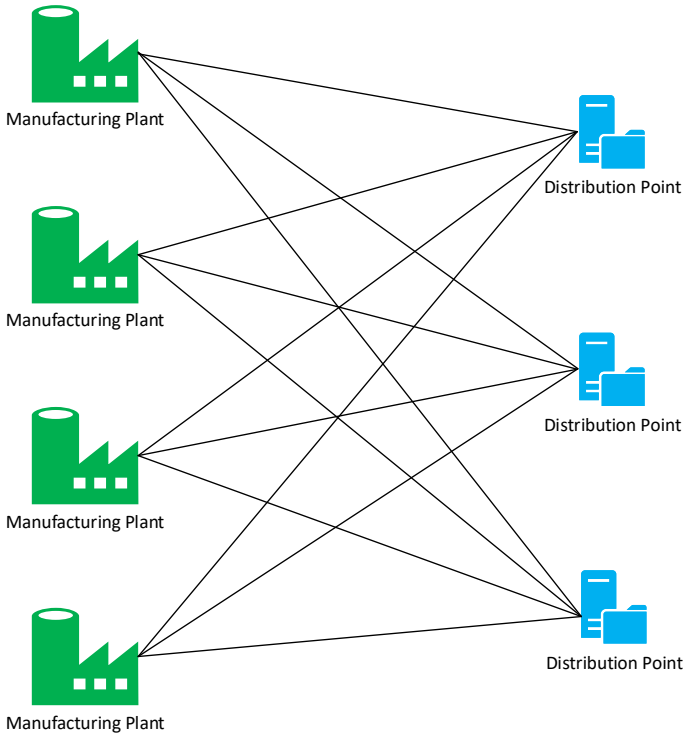

Figure 8: Common transportation problem as a geographical network for $m = 4$ and $n = 3$

The above problem then contains an objective function, $\sum_{j,i} c_{ji}x_{ji}$, aptly named as it contains the objective of the problem, to solve for x_{ji}. As the above problem now has a well defined objective function, it can be minimized through an applicable optimization process to provide the product quantities for which the cost of transportation is a minimum. In this particular example, the optimization problem is then one more specifically of minimization and as such, the objective function can be referred to as a loss function or cost function [24], however often times the desire is to maximize some quantity. Under such circumstances, the objective function is then referred to as a utility or fitness function. For the purposes of this dissertation, only loss functions will be considered, as all current literature regarding the estimation of differential networks make use of minimization to arrive at the estimate [9]. The above loss function is a rather specific one, with little value to problems other than the transportation one described above, however throughout statistics, there are several well known and commonly used loss functions, such as:

- Mean Square Error Loss: MSE $= \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$
- Mean Absolute Error Loss: MAE $= \frac{1}{n}\sum_{i=1}^{n}|y_i - x_i|$

- Huber Loss: $\text{Huber} = \begin{cases} (y_i - \hat{y}_i)^2 & \text{for } |y_i - \hat{y}_i| \leq \gamma \\ 2\gamma|y_i - \hat{y}_i| - \gamma^2 & \text{otherwise} \end{cases}$

where n is the number of observations, y_i are the observed values, $\hat{y}_i$ are the predicted values, and γ is a hyper-parameter that allows one to control the influence extreme values have on the model. The above mathematical functions, can then be represented graphically as is shown in Figure 9.

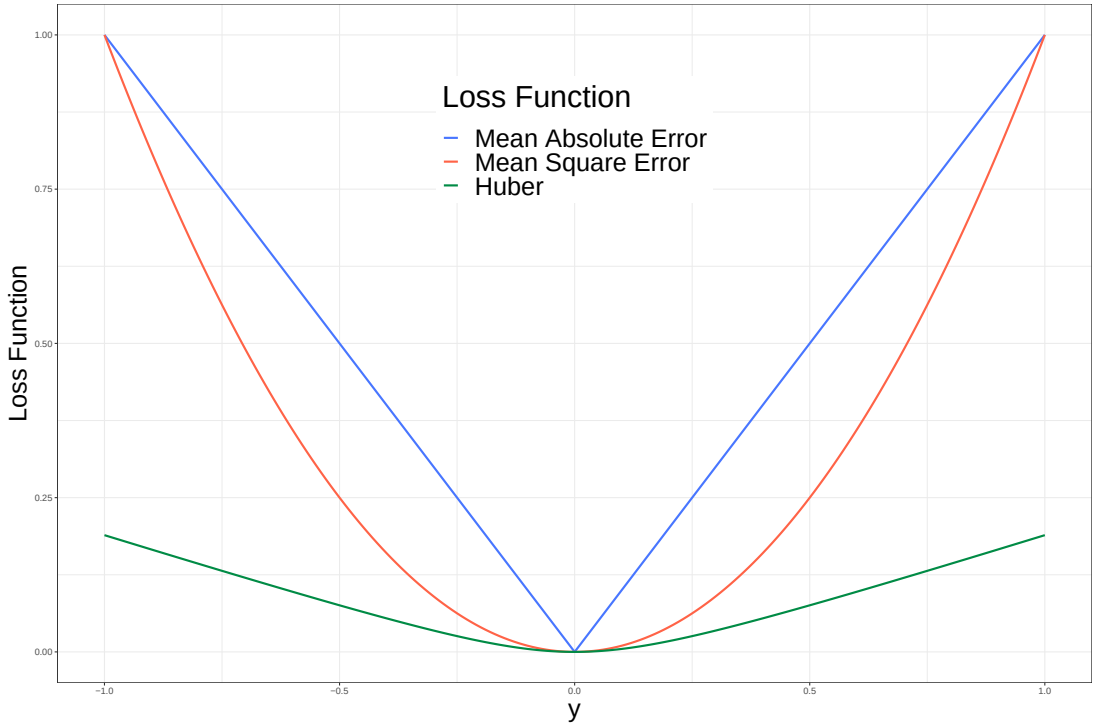

Figure 9: Common loss functions for regression

These loss functions although powerful, are not always applicable. For example, when performing classification, one may be tempted to deploy the above loss functions, but these loss functions were specifically developed to aid in regression-type problems and as such perform poorly in classification settings. It is thus vitally important that the loss function is correctly chosen based on the circumstances of the problem at hand. Keeping this in mind, there are several well-defined loss functions that have been proven both theoretically and numerically to handle the task of differential network estimation extremely efficiently. These loss functions will be the focus of this dissertation, and are the graphical LASSO loss function of Friedman et al. [28], d-trace loss function of Yuan et al. [90], SCAD, and MCP loss functions [52].

The above loss functions, will then be minimized through the use of optimization. Optimization techniques usually involve a significant amount of non-trivial mathematics as matrix derivatives are utilized. At their core however, optimization techniques simply begin with an initial *guess* for the quantity of interest [59]. This guess is then plugged into the algorithm which then returns an update on the

quantity of interest, this process is then repeated several times until the returned quantity is equal to the supplied, indicating convergence [59]. For the purposes of this dissertation, the optimization scheme that will be used in conjugation with each of the previously mentioned loss functions is that of alternating direction method of multipliers, ADMM. ADMM is considered a robust, state-of-the-art optimization scheme capable of efficiently solving the loss functions of interest even in high-dimensional settings where other optimization schemes encounter difficulties [9]. Hence, both loss functions and optimization as a whole have been discussed briefly while the specific loss functions and optimization technique deployed throughout this dissertation have been introduced further specifics regarding their uses within differential network estimation are covered in Section 3.

1.4 Literature Review

This section acts as a description of the key sources which were consulted throughout this dissertation. The sources considered within this section are by no means exhaustive either in terms of the sources that were reviewed for this research, or in terms of the available literature. This section does however briefly describe the sources that make significant contributions to this dissertation.

Arguably one of the two most valuable sources that was consulted is that of the work by Shojaie [65]. Within his paper, Shojaie provides an extremely comprehensive overview of differential networks as a whole. This includes the definition of a differential network, and recognizes their subsequent use thereof within medicine to explore disease development and progression. Shojaie next provides a description of the two different associations that one may consider to determine undirected edges within a network [65]. Gaussian graphical models are also introduced, but so too are graphical models in which the underlying distribution is no longer assumed to be normal [65]. Graphical models with different underlying distributions are not explored here, but could form the foundation of beneficial future work. Lastly, Shojaie provides the framework of various statistical methods for performing differential network analysis. This includes how one can go about hypothesis testing whether differences between networks exist locally or even globally [65].

The second most valuable source reviewed is the paper by Tang et al. [69] whose paper introduces differential networks alongside the reasoning why a differential network can very rarely be directly determined and as such must be estimated instead [69]. This will be discussed in greater detail in Section 2. Tang et al. also provide a brief overview of existing estimation procedures, before introducing the mathematics and theory behind their own proposed algorithm. Their methodology, although new, still relies on the same optimization technique as preceding approaches [69]. That is, most differential network estimation frameworks make use of alternating direction method of multipliers, ADMM, as the optimization process with the difference between these frameworks arising from their use of different loss functions

[69]. Lastly, they provide a practical comparison between algorithms on simulated data which seem to support their hypothesis that their proposed method is indeed faster than previously defined methods [69].

To complement the above sources, several other articles were consulted for the relevant details regarding the estimation and optimization as discussed by Tang et al. [69]. These remaining sources can then be categorized as follows: covariance matrix estimation, differential network estimation, optimization and the application of these theories in practical settings.

1.4.1 Covariance Matrix Estimation

Before considering the estimation of differential networks, it is important to explore the estimation of covariance matrices as many of the estimation techniques for differential networks are merely extensions of methods for covariance matrix estimation.

The first source that was considered in this vein is the work of Bickel and Levina [5]. Their work focuses on the estimation of a single covariance matrix, such as through the use of the maximum likelihood estimate $\hat{\mathbf{\Sigma}} = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T$, with $\mathbf{x_i} = (x_1, ..., x_p)^T$ with $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$, commonly seen throughout literature [5]. The paper then provides details regarding regularizing the sample covariance matrix, or its inverse to allow one to overcome the difficulties experienced in the estimation of $\mathbf{\Sigma}$ when either $p \to \infty$ or $n \to \infty$. With n the number of observations and p the number of dimensions. Lastly, the paper shows that the estimates obtained by means of their regularization schemes are consistent given the covariance matrices are considered well-conditioned [5]. This is a concept that is deployed within the R package of Tang et al. [69].

The paper by Rothman et al. [62] builds on this idea. That is, Rothman et al. introduce the concept of generalized thresholding of large covariance matrices to ease the estimation process [62]. Generalized thresholding combines shrinkage methods with those of thresholding, with the objective to gain a more flexible approach than simply making use of either thresholding or shrinkage on their own [62]. Hence, one can apply any elementwise generalized thresholding that satisfies the conditions as given by Rothman et al. and obtain a consistent estimate of the covariance matrix that would otherwise be non-trivial in a high-dimensional setting. This idea follows closely to that of smoothly clipped absolute deviation, SCAD, regression which will be explored in Section 4. and as such is important to beware of. Lastly, within the application section Rothman et al. compare the accuracy of the covariance matrix estimates to the true covariance matrices. This is something that is not done by Tang et al. [69], and will be addressed in this dissertation.

One shortfall of the the work done by Rothman et al. [62] is that the generalized thresholding needs to be manually specified and as such is not adaptively tuned according to various properties of the

covariance matrix itself. Cai and Liu [12] were able to directly address this shortfall, with one additional condition. Adaptive thresholding is a technique that applies generalized thresholding to sparse covariance matrices, depending on the variability of the matrix elements [12]. Put simply, the generalized threshold as described by Rothman et al. applies a single threshold to every element of the covariance matrix, while adaptive thresholding allows for the threshold to vary over these same elements [12]. This results in an approach with greater flexibility which is more applicable to commonly seen heteroscedastic problems while still remaining theoretically valid but also lowering the computational cost of estimation [12]. Finally, an important consideration which is shown within this article that has thus far be excluded is the selection of any tuning parameters. That is, Cai and Liu [12] provide a robust manner which one can employ to tune parameters when performing covariance matrix estimation that will be extended to the estimation of differential networks.

The next two papers which were examined follow closely in nature. That is, both sources provide an overview of single covariance matrix estimation and various estimation schemes one can make use of rather than a single scheme as is the case in the above 3 sources. The first source is the work of Tong et al. [71] in which the estimation of several matrices in a high-dimensional context is reviewed. Tong et al. discuss the use of variance and covariance matrix estimates in several branches of statistics such as linear discriminant analysis, principal component analysis and for the construction of t-tests [71]. The use of these estimates prompts their discussion on the estimation of variances through various methods. The provided variance estimators fall within one of three families. Namely the estimators are either regression, Bayesian or shrinkage in nature [71]. Secondly, three further families of estimators specifically for covariance matrix estimation are also included. These families include sparse estimators, ridge-type estimators and lastly Stein-type estimators [71]. Finally, the families of ridge-type estimators and sparse estimators are extended to allow for the estimation of precision matrices [71]. It is however noted that the Stein-type estimators are also known to be easily extended to precision matrix estimation.

The other source consulted that excellently summarises the estimation of both covariance and precision matrices, is the work of Fan et al. [27] which acts as a great compliment to the theory of Tong et al. [71]. Here, Fan et al. provide an overview of thresholding in the form of adaptive thresholding, entry-dependent thresholding and generalized thresholding. Fan et al. [27] also introduce a *thresholding constant* that allows one to guarantee the estimate obtained is positive-definite [27]. As with Tong et al. [71], Fan et al. [27] then extend their theory on covariance matrices to precision matrices. This includes the estimation of precision matrices through column-wise or penalized likelihood approaches [27]. Arguably the most important section within this paper, is the component regarding tuning-insensitive approaches to precision matrix estimation, TIGER[1] and EPIC[2] [71]. This is extremely valuable as current methods

[1]The tuning-insensitive graph estimation and regression method
[2]The estimating precision matrix with calibration method

for differential network estimation rely on at least one tuning parameter, λ, that controls the level of sparsity of the estimate [69]. The selection of this λ is by no means straight forward and any adaptation of existing estimation procedures to one that is more tuning-insensitive would be welcomed as a more robust alternative.

All of the above sources mentioned within this sub-section only consider the estimation of a single matrix, and as is the case with differential networks it may be more beneficial to estimate various matrices simultaneously [94]. The above summarised work has been broadened to include the estimation of various matrices simultaneously. A particular occurrence of this, is that of Chiquet et al. [17]. Chiquet et al. critically evaluate the estimation of several matrices simultaneously in an attempt to overcome the assumption in a Gaussian graphical model that the underlying data arises from an independent and identically distributed sample [17]. Experience suggests however that this assumption will most likely not hold in many real-world scenarios and as such Chiquet et al. propose a remedy to overcome violations of this assumption. The suggested solution is to estimate multiple GGMs, with each GGM matching a different distribution that corresponds to a different experimental condition [17]. Chiquet et al. go on to then provide the details of three LASSO-type approaches which are shown to numerically perform at least as well as the neighbourhood selection of Meinshausen and Bühlmann [48].

Another valuable contribution to the estimation of multiple covariance matrices is that of Guo et al. [34]. The objective of the authors here was near identical to that of Chiquet et al. with the main difference being the estimation procedure itself. That is, Guo et al. derived a method to estimate multiple Gaussian graphical models by leveraging the expected common structure across each of the covariance matrices while still allowing for differences to exist between their elements [34]. This objective is achieved through the use of a hierarchical penalty function within the loss function that removes repeated zeros across the matrices to ease the computation [34]. Hence, the method of Guo et al. can be considered to exploit the sparsity pattern of covariance matrices, which will be commonly observed within differential network estimation.

The last source consulted on multiple covariance estimation is the work of Zhu and Li [96]. They introduce an important distinction to the estimation process that differentiates their proposal significantly from the above methods. In estimating a covariance matrix or its inverse if a loss function is utilized, it is almost always convex in nature. This is due to the fact that convex loss functions can be easily minimized through the use any one of the many well-documented optimization procedures. However, the distinction which Zhu and Li [96] make is that their suggested loss function is non-convex. This adaptation presents difficulties both computationally and theoretically, but Zhu and Li were able to prove mathematically that the minimum of their loss function is indeed the precision matrix of interest [96]. Lastly, Zhu and Li [96] specify an optimization scheme for their loss function and show numerically that their method

outperforms existing methods for precision matrix estimation. The idea of deploying a non-convex loss function is one that will also be applied to differential networks within this dissertation such as SCAD in the context of regression.

1.4.2 LASSO and Alternatives

Up until this point the method of estimation has differed greatly between each of the above sources, however there are several estimation techniques for both single matrices and differential networks as a whole that have been built on the work of Tibshirani [70]. This particular paper is widely considered within statistics as one of the most influential papers to have been published and has amassed nearly 40000 citations to date. The basic idea that Tibshirani introduced was to perform standard linear regression with the inclusion of a constraint on the regression coefficients [70]. The constraint being that if $\beta = (\beta_1, \beta_2, ..., \beta_p)^T$ are the regression coefficients then the following restriction must hold: $\sum_j |\beta_j| \leq t, t > 0$ [70]. The constraint could also be structured into a traditional regression model which then is considered the least absolute shrinkage and selection operator, LASSO, in the context of regression, with the following penalized loss function:

$$\sum_{i=1}^{n}(y_i - \sum_{j=1} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j|, \tag{2}$$

where y_i, $x_i = (x_{i1}, ..., x_{ip})^T, i = 1, ..., n$ are the response and predictor variables respectively with $\lambda > 0$ a tuning parameter that allows one to control the amount of shrinkage imposed on any β_j [70]. Depending on the choice of λ, it is very common that several of the regression coefficients are eliminated from the model, as their value is shrunken to zero. This results in a simpler model than originally had. The ability to *remove* coefficients from the model provides the ability to be able control both the variance of the parameter estimates but also their ensuing bias [70]. The power of Tibshirani's regression model is thus clearly evident but in its current state cannot be readily applied to estimate either precision or covariance matrices respectively.

Meinshausen and Bühlmann [48] did however extend LASSO regression as defined by Tibshirani to perform matrix estimation. They describe how the use of LASSO and neighbourhood selection can accurately estimate large networks. They also show that their combination of neighbourhood selection and LASSO is a viable strategy for variable selection, even in high-dimensional settings [48]. By deploying neighbourhood selection, the assumption that the underlying distribution of the data is normal is technically no longer required. But, Meinshausen and Bühlmann [48] were able to show that by including said assumption, their approach inherited some advantageous theoretical properties. The main one being that their estimator was proven to be consistent [48]. Finally, Meinshausen and Bühlmann were able to show that their proposed procedure achieved favourable results when compared to maximum likelihood

based estimation. Hence, while the method introduced by Meinshausen and Bühlmann is a valuable contribution to matrix estimation as a whole, their comparison is less than ideal. That is, the issues that stem from attempting to implement maximum likelihood estimators in high-dimensional settings are well-documented and a more useful contrast of approaches could have been included. One such example could have been to include the work done by Zhu and Li [96] or Guo et al. [34].

The next source inspected is that of Friedman et al. [28]. In this journal paper Friedman et al. first establish graphical LASSO which forms the foundation of many differential network estimation algorithms. Graphical LASSO provides a powerful, systematic approach to estimate sparse inverse covariance matrices [28]. The difference between the theory mentioned by Friedman et al. [28], and the content described by Rothman et al. [62] and Cai and Liu [12] is that Friedman et al. assumes the observations within the data follow a multivariate normal distribution with mean μ and covariance matrix $\mathbf{\Sigma}$ [28]. This assumption of normality provides a very convenient loss function in the form $\log(\det(\mathbf{\Omega})) - \text{tr}(\mathbf{S\Omega}) - p||\mathbf{\Omega}||_1$, with $\mathbf{\Omega} = \mathbf{\Sigma^{-1}}$, $\mathbf{S} = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(x_i - \bar{x})^T$ the sample covariance matrix, p the dimension of the data and $||\mathbf{\Omega}||_1 = \max_{1\leq j\leq n}\left(\sum_{i=1}^{n}|\Omega_{ij}|\right)$ [28]. This loss function is convex and as such can be efficiently solved through the use of block coordinate descent [28]. Thus, Friedman et al. described a model that is both computationally and theoretically attractive to the estimation of precision matrices such as those seen as the components of a differential network.

Despite the groundbreaking nature of graphical LASSO at the time of publication by Friedman et al. [28], it is not without fault. In 2012 Mazumder and Hastie [47] published a paper showing the there are several circumstances under which graphical LASSO fails to converge [47]. They were also able to illustrate that in situations whereby the algorithm converged, the resulting precision matrix was not the true inverse of the estimated covariance matrix [47]. The reason for such issues was discovered to be as a result of the choice of target in the optimization procedure [47]. That is, in graphical LASSO the target of the block coordinate descent optimization is $\mathbf{\Sigma}$ and not $\mathbf{\Sigma}^{-1}$ [47]. Mazumder and Hastie [47] proposed two modifications to circumvent theses issues seen in graphical LASSO. Both of the modifications involve the use of alternative optimization algorithms which make use of $\mathbf{\Sigma}^{-1}$ as the objective of the minimization [47]. The result was alternatives to standard graphical LASSO, within the same family of algorithms so to speak, that performed as least as well as graphical LASSO in most settings, but also outperforming it under other circumstances [47].

Friedman et al., Mazumder and Hastie are not the only authors that have published alternatives to traditional LASSO, in 2006 Zou [97] discovered a powerful manner to improve upon the work of Tibshirani [70]. Zou found that the LASSO, when deployed for the purposes of variable selection, is not always consistent. Zou was then able to derive a condition, necessary to guarantee the consistency of LASSO [97]. It is thus plausible to suspect their will be contexts within which this condition is violated

and a further alternative is required. Zou developed such an alternative. Namely, adaptive LASSO which derives its name as it extends LASSO to include weights which adaptively penalize each regression coefficient [97]. That is, taking the LASSO model as derived by Tibshirani, adaptive LASSO has the following loss function:

$$\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \hat{w}_j|\beta_j|, \tag{3}$$

with the weight vector being $\hat{w_j} = \frac{1}{|\hat{\beta_j}|}$, $\hat{\beta}_j$ being the jth element of any consistent estimator $\hat{\beta} = (\hat{\beta}_1, ..., \hat{\beta}_p)^T$ of $\beta = (\beta_1, ..., \beta_p)^T$ and γ simply chosen so that $\gamma > 0$ holds. The above model is shown by Zou to outperform standard LASSO in various conditions but not universally. As such, rather than invalidating LASSO, adaptive LASSO provides a powerful compliment that could possibly be extended beyond the purposes of variable selection to accomplish covariance precision matrix estimation [97].

In fact, Zhou et al. show adaptive LASSO can readily be utilized for covariance matrix estimation [95]. However, adaptive LASSO, as described above, is technically a two-stage process. The first step involves the determination of appropriate weights for the regression coefficients, with the second step being the implementation of said weights within the actual regression model itself [95]. Previously, Zou suggested making use of the estimated ordinary least squares regression coefficients to determine the initial weights. In high-dimensional settings, the calculation of the required inverse for ordinary least squares is by no means trivial so Zhou et al. suggested a substitute. The substitute is to first apply standard LASSO and obtain a set of regression coefficient estimates, these estimates can then be plugged in and used to determine the weight vector [95]. Adaptive LASSO can then be implemented, with the requirement of weaker conditions than those required by say graphical LASSO, but with a greater computational cost due to the two-step nature of the procedure [97]. One final challenge encountered with adaptive LASSO, is that there exists a small probability that the estimate obtained is not necessarily unique [95]. This is an issue that afflicts all optimization techniques when the loss function is convex in nature, but not strictly convex and as such is not a concern isolated to only adaptive LASSO.

All of the loss functions considered thus far within this section are either that of LASSO or direct relatives thereof. However, several other authors have developed their own loss functions to perform the same task. Zhang and Zou released a paper in 2014 which identified a weakness with the most commonly used loss functions for sparse precision matrix estimation, including graphical LASSO. The weakness being that these loss functions do not guarantee the final estimate is positive-definite [92]. Zhang and Zou then aimed to make use of the same assumptions as Friedman et al. and develop their own loss function which guarantees a positive-definite precision matrix estimate [92]. This gave rise to the convex D-trace loss function which improved upon the graphical LASSO in various aspects. Namely, the D-trace was mathematically simpler hence allowing for greater theoretical analysis but also easing the

computational cost despite relying on the same optimization architecture [92]. Zhang and Zou however concluded that in order to make use of their loss function, an additional condition over those required for graphical LASSO is necessary. This condition, named the *irrepresentability condition* does not always hold and as such it is possible for there to be circumstances in which the graphical LASSO loss function outperforms the D-trace loss function as is shown by Zhang and Zou in their numerical results. Thus, one cannot conclude one loss function dominates the other, but rather both loss functions have their own individual merits [92].

The above idea is further extended if the loss function of Cai et al. [13] is considered. Just as Zhang and Zou built upon the work of Friedman et al. so too did Cai et al. when developing their loss function. Cai et al. were able to derive a method for inverse covariance estimation, aptly named *constrained* ℓ_1 *- minimization for inverse matrix estimation*, CLIME. CLIME was then shown to have favourable rates of convergence under weaker assumptions than graphical LASSO [13]. These rates of convergence were derived theoretically and seen experimentally to hold when the data had an underlying distribution whose tail was polynomial or exponential [13]. CLIME does have a well-documented drawback that plagues covariance and precision matrix estimation. That is, the CLIME estimator is commonly solved for by iteratively stepping through each data column [13]. This results in a precision matrix estimate that is not necessary symmetric and subsequently requires a further step to symmetrize the estimate and obtain a valid precision matrix estimate. The added step can be done in several manners, but will always add to the computational cost taken to arrive at a valid estimate.

Although both of the above sources aimed to improve upon graphical LASSO, there is one matter upon which neither loss function improved. The matter being that the penalty portion of the loss function is linear with regards to the regression coefficients. This is easily verified by inspection of the penalty most commonly written as $\lambda \sum_j |\beta_j|$ in literature. As a result, LASSO results in biased estimates for large regression coefficients [26]. One way to address this bias is through the use of non-convex penalties such as the smoothly clipped absolute deviation penalty, SCAD, [26]. One paper which explores this, in the specific context of network estimation is that of Fan et al. [26]. Fan et al. implement and compare SCAD with another non-convex penalty already encountered - adaptive LASSO [26]. Within this comparison it is discovered that SCAD produces estimates that satisfy the three desirable properties of any network estimate. These properties are that the estimate is sparse, consistent and unbiased for large values [26]. The estimator of Fan et al. is obtained by adjusting adaptive LASSO to iteratively re-weight the regression coefficients through the use of the SCAD penalty [26]. The estimator itself is still however solved by the framework of graphical LASSO [26]. Despite the use of the same optimization approach the SCAD penalty performs better than both adaptive LASSO and standard LASSO in both the simulation studies and real-world example done by Fan et al. [26]. That is, SCAD produced the estimates with the

lowest bias, as it provided the estimates with the highest sparsity [26]. Indicating that SCAD was able to correctly set more of the spurious non-zero entries in the network estimate to zero [26].

The SCAD penalty is not the only non-convex penalty that has been the recipient of significant attention recently for performing network estimation. Another such penalty is that of the minimax concave penalty, MCP, introduced by Zhang [91]. MCP, like SCAD aims to address the issues of bias seen within implementations of LASSO [91]. According to Zhang, MCP does this by "minimizing the maximum concavity" [91]. Broadly speaking this means that MCP is the closest penalty function to being convex, while actually being non-convex.[3] Zhang provides derivations confirming that MCP maintains the favourable properties of nearly unbiased and consistent estimates, even in ultra high-dimensional settings, $p >> n$, while remaining computationally efficient [91]. There is a further distinction between the process described by Zhang and that described by Fan et al. for SCAD. Unlike SCAD, MCP does not make use of the graphical LASSO framework to minimize the loss function, but rather Zhang introduced an algorithm specifically for this task. The algorithm in question is that of PLUS, penalized linear unbiased selection [91]. The reason for the development and deployment of PLUS here is due to its unique ability to move the solution path between local minima where other algorithms would otherwise be stuck in a single local minimum [91]. This combination of MCP and PLUS indicate that Zhang's approach should obtain the global minimum rather than a local minimum more frequently than graphical LASSO. The phenomena of optimization methods getting caught in a local minimum is visualized below.

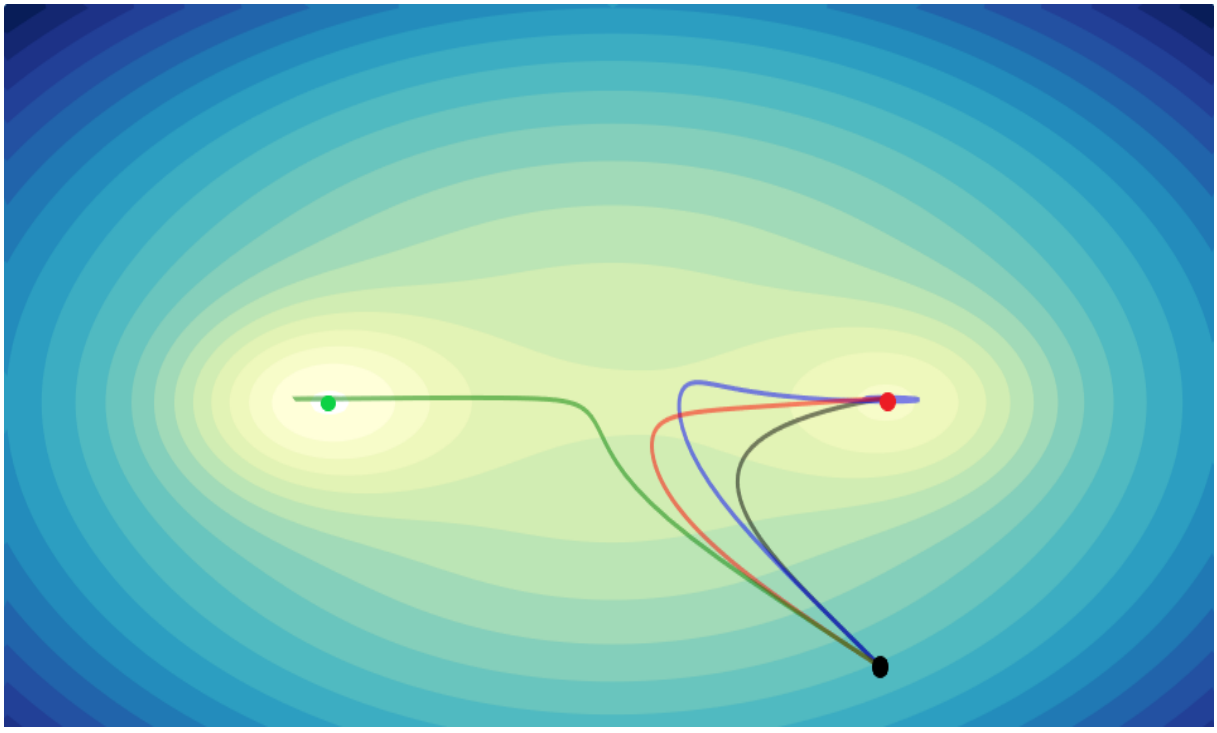

Figure 10: Contour plot with optimization methods

[3]Kenneth Tay, "The minimax concave penalty (MCP)", Statistical Odds & Ends, retrieved on April 23, 2021 from `https://statisticaloddsandends.wordpress.com/2019/12/09/the-minimax-concave-penalty-\gls{MCP}/`

Figure 10 represents a contour map to illustrate a 2-dimensional optimization problem. The lighter the colour, the lower the point in the contour. It is known that the global is minimum is situated at the green dot, while there is a local minimum situated at the red dot. Each of the algorithms have the same starting values, shown as the black dot. The objective of the optimization algorithms, represented by the 4 coloured lines, is to determine the absolute minimum. It is clear that 3 of the 4 algorithms would lead to erroneous thinking that the local minimum is the global minimum.. This clearly shows the importance of optimization algorithms requiring tools to exit local minima to avoid such errors.

To conclude, two sources in which multiple of the above techniques were simultaneously examined are considered. Namely the work of Pötscher and Leeb [60], and Ogutu and Piepho [52] respectively. Pötscher and Leeb investigated the distributional and asymptotic distributional properties of estimators obtained through LASSO, SCAD (also called soft thresholding) and hard thresholding [60]. Pötscher and Leeb confirm that each of the estimators is indeed consistent, while also providing the multimodal, and thus non-normal, finite-sample distributions [60]. Pötscher and Leeb also contradict many of the above papers. The contradiction being that previous derivations of the *oracle property*[4] may not necessarily hold as previously suspected [60]. Next Ogutu and Piepho implement MCP, SCAD and LASSO amongst various other regression models to perform genomic prediction. This paper is of particular interest as it focuses solely on the application aspect of each of the regression models, instead of the heavy focus on the theoretical components observed thus far. Ogutu and Piepho aim to compare the prediction accuracy of each of the models. An interesting aspect of this comparison is that the prediction accuracy greatly improved across most of the models when any tuning parameters were selected through the use cross validation rather than through the use of information theoretic criteria [52]. This is of particular interest as Tang et al. [69] only made use of such criteria to select their tuning parameter and may mean their methods could be readily improved upon. Lastly, Ogutu and Piepho concluded that none of the selected regression models consistently outperformed any other, with the significant difference between models only being observed in their computation time taken to arrive at the solution [52]. Thus, having explored in great detail the most popular approaches used for network estimation, the focus now shifts to those approaches for differential network estimation.

1.4.3 Differential Network Estimation

Differential network analysis is not a particularly new field however the estimation thereof is extremely new with very few specialized procedures currently documented. That is, differential network estimation can be broken into two main approaches. Either each individual precision matrix is estimated separately using any of the approaches discussed in Section 1.4 and the difference of the estimates is then taken, or the differential network as a whole is estimated. This subsection focuses upon the methods that aim to

[4]The penalized estimator has the same performance regardless of whether the true underlying model was known a-priori.

estimate the differential network as a whole. This will be the case throughout the dissertation, as the focus is on the differential networks, each of the condition specific networks do not necessarily need to be recovered.

In 2014, Zhao et al. developed the first procedure for the direct estimation of differential networks. Their proposed approach was to deploy a Dantzig-type[5] estimator that was uniquely equipped to handle high-dimensional problems [94]. Zhao et al. were also able to show that an advantage of the direct estimation of differential networks is that any common features observed across each of the precision matrices can be exploited to increase the sparsity and subsequently ease computation [94]. Direct estimation is not without its shortfalls. One such shortfall is that the estimate is not necessarily symmetric [94]. However, a differential network is simply the difference between two symmetric matrices and as such should always be symmetric. A further step is then introduced by Zhao et al. to guarantee the symmetry of the estimate. Zhao et al. like many of the authors mentioned here make use of block coordinate descent to optimize their loss function and the Akaike information criterion, AIC, to tune their model [94]. An important note here is that the technique here is not fundamentally built on single matrix estimation but was designed specifically with differential network estimation in mind.

In contrast to the above, there is an approach for differential network estimation that is entirely based upon existing covariance and precision matrix estimation. That is the D-trace loss function defined in [92] is extended to a differential network context by Yuan et al. [90]. As was the case before, the algorithm is built upon the LASSO framework first developed by Tibshirani [70] and then later extended by Friedman et al. [28]. As such, the optimization technique remains largely the same, however the optimization does become a two-step procedure that provides a symmetric estimate [69]. The only significant difference is then in the loss function, as the loss function must now undertake a new target [90]. Lastly, Yuan et al. opted to deploy the Bayesian information criterion in their practical applications to select their tuning parameter. In these practical applications it is shown that the D-trace loss function outperforms the Dantzig-type approach of Zhao et al. and even the fused graphical LASSO of [23]. Thus, arguably making the D-trace the most attractive estimation procedure at the time of its publication.

However, this did not remain the case for very long. In 2018, Jiang et al. derived a cutting edge technique to perform quadratic discriminant analysis. Their approach as is the case within discriminant analysis requires the calculation of a discriminant function [38]. This function contains one term of particular interest to this dissertation. Namely, $\boldsymbol{\Omega} = \boldsymbol{\Sigma}_2^{-1} - \boldsymbol{\Sigma}_1^{-1}$ [38]. This means that in order to execute quadratic discriminant analysis in a high-dimensional setting as described by Jiang et al., differential network estimation is required. Knowing this fact, Jiang et al. formulated their own approach for differential network estimation, employing block coordinate descent to minimize their specified loss function. Their approach holds considerable computational advantages over techniques such as those of Zhao et al. [94]

[5] A tribute to George Dantzig, as the loss function is convex in nature and essentially a variable selection procedure.

and even Yuan et al. [90]. This is a substantial feat, given Yuan et al. guaranteed a symmetric estimate, while as in the case of Zhao et al. [94], Jiang et al. required an additional step to ensure symmetry [38]. It is important to note that for each of the three differential network estimation algorithms mentioned above, the estimators were proven to be consistent [69]. From a theoretical perspective, each of the algorithms then appear to be on equal footing, with the main difference being observed across their computational costs.

To further explore the use and need of the above techniques two further sources are consulted. These sources both focus upon on applications in a biological setting, specifically protein or gene interactions and microarray[6] data. The sources consulted are that of Gill et al. [31] and Ideker and Krogan [37] respectively. Ideker and Krogan focus on the use of previously determined networks to illustrate the value of differential networks and their analyses for gene interactions. Ideker and Krogan conclude that as access to state of the art technologies increases, so too will the availability of protein and gene interaction data [37]. It is at this point in time that Ideker and Krogan suspect that differential network analysis will become commonplace as a result of being one of the few approaches uniquely equipped to such data [37]. Gill et al. provide a different perspective, in that their paper focuses upon one of the metrics which do exist to perform differential network analysis. Namely, a connectivity score that provides insights into the strength of network interactions should they exist [31]. This score can be calculated to explore the interaction between any two nodes, or aggregated to gain information on the entire network as a whole [31]. One fact evident from both of these sources is that the data in question for which differential networks may be of the most use is usually not normal and as such violates the assumption made by Gaussian graphical models [65]. The need for procedures to circumvent such scenarios is then vitally important.

Such procedures do exist, such as those formulated by Liu et al. [42], Liu et al. [43] or Xue and Zou [89]. Each of these papers introduce the concept of a nonparanormal distribution. This refers to any distribution, whose cumulative distribution function can have a set of unknown nonparametric transformations applied to make the cumulative distribution function equal to that of the standard normal distribution [42, 43, 89]. Simply, a transformation on the variables follows the normal distribution. The ability to make such a transformation relaxes the assumption that the underlying data must be normal, and as such provides the ability to extend Gaussian graphical modelling based estimation to non-normal data [42]. This adaptation implies that the differential network estimation discussed thus far within the dissertation is more robust, as there are more applicable scenarios for said procedures. Lastly, the details of these procedures although excluded here are later discussed in Section 3.3.

[6]The study of the interactions/states of multiple genes simultaneously. This typically involves examining thousands of genes concurrently.

1.5 Research Objectives

The objectives of the work shown within this dissertation can be categorised into one of two components. The first component is a thorough and comprehensive exploration of differential networks. This includes an examination of all the relevant theory on the definition and formulation of differential networks, as well as a brief overview of several real-world scenarios in which differential networks are currently deployed. In addition to this, a motivation as to why differential network estimation is often necessary will be provided, accompanied by a study of the various avenues of estimation one may undertake. This will be done such that an individual with no prior knowledge of networks, will be fully aware of their definition, usefulness as well as be equipped with the necessary knowledge to estimate a differential network through the use of the most appropriate estimation method given their specific problem characteristics.

The second component of the objectives within this dissertation, is to extend the currently available and documented estimation techniques. There are several aspects upon which it is hoped that the current techniques may be extended upon. These aspects are summarised below.

- There currently exists several works regarding the estimation of differential networks or the use thereof. However, within these works, there is often little to no discussion regarding the accuracy of the differential network estimate obtained. As such, the accuracy of the various estimates one can obtain through the different estimation procedures will be explored through the use of simulation studies, such that the exact analytical network will be available for comparison.

- Within the above objective, the sole focus will be to compare the estimation algorithms. Although, it is also of interest to determine the effect, if any, that the correlation between variables has upon the estimate acquired. This is important to be able to quantify, as the correlation of the variables is directly linked to the sparsity of the network, where the level of sparsity within the network estimate is often varied through the use of a tuning parameter to ease estimation [5].

- As will be shown in the application of this dissertation, Section 4, current implementations of the available estimation procedures place significant emphasis upon the solution path obtained for the different values undertaken by the tuning parameter, λ [69]. The optimal λ is most often selected by making use of either the Akaike information criterion or the Bayesian information criterion. One problem incurred when making use of such criteria is that, the choice of λ for the two criterion often does not coincide. As such, this dissertation will aim to provide a more robust approach for selecting the optimal λ.

- Lastly, in literature most applications regarding differential networks involve a stringent assumption of normality [42, 47, 65]. In real-world applications however, such an assumption will most likely be violated. It is thus of great importance to examine what impact such a violation has on the

estimation of differential networks. Investigation will also be performed regarding what influence the transformations currently available to relax the assumption of normality have on the estimation procedure [42, 43].

Having defined the objectives of this dissertation, the first component of these objectives is now addressed.

2 Differential Networks

2.1 Background

Differential networks are by no means a recent development within statistics, with their analysis being studied as far back as 2012 [37]. However, the need for tools such as differential networks has never been greater. This is due to the fact that the fourth industrial revolution, the digital revolution, is well and truly underway, and along with it some of humanities greatest technological advancements [63]. These advancements have brought the ability to capture and record more data than ever before. For example, Google the largest search engine in the world is estimated to receive more than 40,000 search queries every second, and holds an estimated 10-15 Exabytes of data, where a single Exabyte is a equal to a billion Gigabytes [49]. Similar patterns can be seen in industries like banking, or healthcare where the availability of data has exploded over the last decade [72]. As a result, it is vitally important for companies, and researchers to have access to robust, well-developed statistical tools that are capable of addressing larger and larger datasets to provide previously unknown insights. Differential networks are exactly that, with one main drawn back. That is, in high-dimensional settings, a differential network cannot be directly determined but must rather be estimated. The reason for this, follows as a result of their mathematical definition. A differential network, $\mathbf{\Delta}$, is as follows:

$$\mathbf{\Delta} = \mathbf{\Sigma}_1^{-1} - \mathbf{\Sigma}_2^{-1}, \tag{4}$$

where $\mathbf{\Sigma}_1^{-1}$ and $\mathbf{\Sigma}_2^{-1}$ are the inverse covariance matrices of state 1 and state 2 of interest respectively [65]. It is clear, that in order to obtain a differential network, one must be able to determine two inverse matrices. In low-dimensional settings this is trivial, but this is not the case in commonly seen high-dimensional scenarios [69]. In such scenarios, the number of features often exceeds the number of observations, and thus the matrices whose inverses are required are that of $\mathbf{\Sigma}_1 : n_1 \times p$ and $\mathbf{\Sigma}_2 : n_2 \times p$ where $n_1 < p$ and $n_2 < p$. It is then a well known algebraic result that $\mathbf{\Sigma_1}$ and $\mathbf{\Sigma_2}$ are both not of full rank, and are singular matrices, whose inverses do not exist [69]. This result is confirmed in the below lemma.

Lemma 2.1. *A matrix that is not of full rank, is non-invertible. That is, its inverse does not exist.*

Proof. First consider what it means for a matrix to be invertible, that is matrix $\mathbf{A}$ is said to be invertible if $\mathbf{AB} = \mathbf{BA} = \mathbf{I}$. Secondly assume $\mathbf{A}$ is not of full rank, and that its columns are $v_1, ..., v_n$ which are then linearly dependent. By definition of linear dependence, there are then constants $c_1, ..., c_n$ not all equal to zero, such that:

$$c_1\underline{v}_1 + ... + c_n\underline{v}_n = \underline{0} \tag{5}$$

Next, if $\underline{w}$ is defined as a vector containing all the entries $c_1, ..., c_n$, then $\underline{w} \neq \underline{0}$. Then rewriting equation 5 as follows:

$$\mathbf{A}\underline{w} = c_1\underline{v}_1 + ... + c_n\underline{v}_n = \underline{0} \tag{6}$$

But it is also known that:

$$\mathbf{A}\underline{0} = \underline{0} \tag{7}$$

Hence, if $\mathbf{A}^{-1}$ does exist then:

$$\underline{0} = \mathbf{A}^{-1}\underline{0}$$
$$\underline{w} = \mathbf{A}^{-1}\underline{0}$$

Which implies that $\underline{w} = \underline{0}$ which is not the case, and as such $\mathbf{A}^{-1}$ does not exist.

□

Hence, having illustrated the main issue encountered when attempting to determine a differential network, it is important to note that there are means to overcome such an issue. That is, in practice a differential network is almost never directly calculated but rather it must be estimated through the use of sophisticated mathematical and statistical techniques [69]. Further details of these techniques are provided in Section 3.

With the definition of a differential network as well as the sciences and industries wherein they are of great interest introduced previously, the motivation for why one might wish to obtain a differential network is now discussed. A differential network is a powerful tool which captures changes in the conditional correlations between two states [69]. Put more simply, a differential network gives insights and allows one to observe changes in the interactions amongst variables given some environmental stimuli has been altered [69]. This ability is showcased in the below figures.

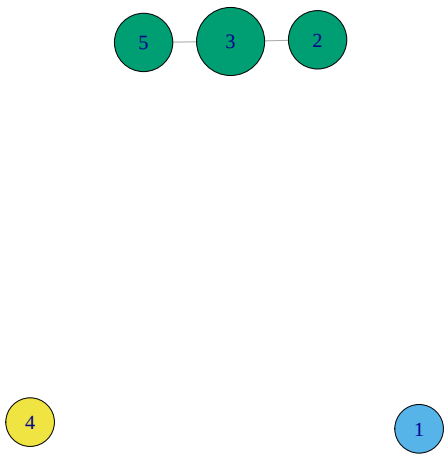

Figure 11: Graphical model for state 1

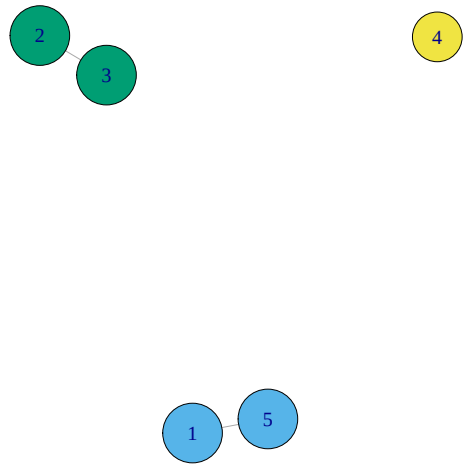

Figure 12: Graphical model for state 2

Figure 11 and Figure 12 represent two graphical models that illustrate the relationships between 5 different features for the same cohort under two different experimental conditions. Using the common biological settings for networks, assume state 1 represents the before treatment measurements for the cohort of subjects, all of which have a particular illness of disease. Then, assume state 2 represents the after treatment measurements for this same cohort. A differential network can then be obtained from these networks to clearly illustrate the differences between the interactions of features across these two states as a whole.

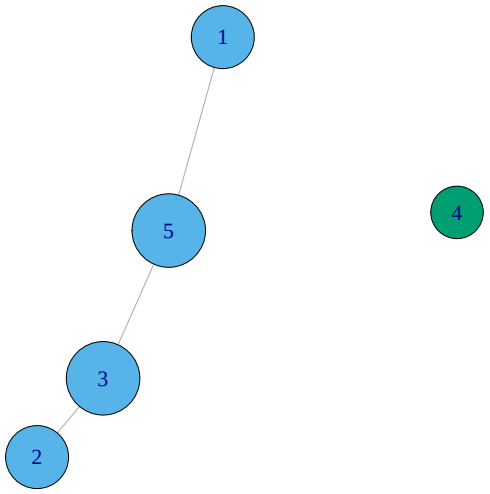

Figure 13: Differential network for the simulated states

Figure 13 is exactly that differential network. This differential network, can then be examined to find the relationship between features 3 and 5 has been altered in response to the treatment while so too has the relationship between feature 1 and 5, and that between feature 3 and 2. Each of these changes can actually be seen when performing visual inspection of Figure 12 and Figure 11, but any analysis done through the means of the visual inspection of two separate networks becomes exponentially harder as the number of features grows. It is also known that in many biological studies, the number of features available can easily be in the millions making such a comparison not feasible [65]. A differential network's value is thus its ability, particularly in high-dimensional settings whereby changes amongst relationships are not easily observable across multiple networks, to accurately and conveniently represent such changes [69]. This ability, combined with the ever-growing size of datasets, has made differential networks a cornerstone of statistical analysis in many areas of biological sciences where networks have been used to gain understanding of disease progression and initiation [65].

This popularity of differential networks is expected not only to grow further within biological sciences, but also to spread to other industries such as finance and social sciences whereby differential networks currently see little to no implementation [65]. For this reason, this dissertation aims to provide a robust R package that is freely available on CRAN that allows any and all researchers or companies to implement differential networks in a manner that is both easy to do so, and computationally efficient. Before specific details regarding the capabilities and usage of this package are provided, it is important to note that the network examples shown within this section have only been for illustration purposes, and do not reflect real-world scenarios. It is noteworthy to then include a real-world example.

2.2 Example Usage

A practical example of a graphical model derived from a differential network is now provided. The example below considers a estrogen microarray dataset available within R, specifically the Bioconductor package [36]. The dataset consists of the responses of various genes to estrogen simulation. The procedure was to first normalise the data, in accordance with the assumptions described previously. Secondly, to simplify any plotting and the further results obtained, simple linear regression was applied to determine the 25 most significant genes within the dataset. The objective is to then investigate how the interactions amongst genes change between the before estrogen simulation state and the after estrogen simulation state. This was done through means of a differential network, which in this particular case did not need to be estimated, as the data in question is not high-dimensional in nature, and as such the differential network was calculated directly in Figure 14.

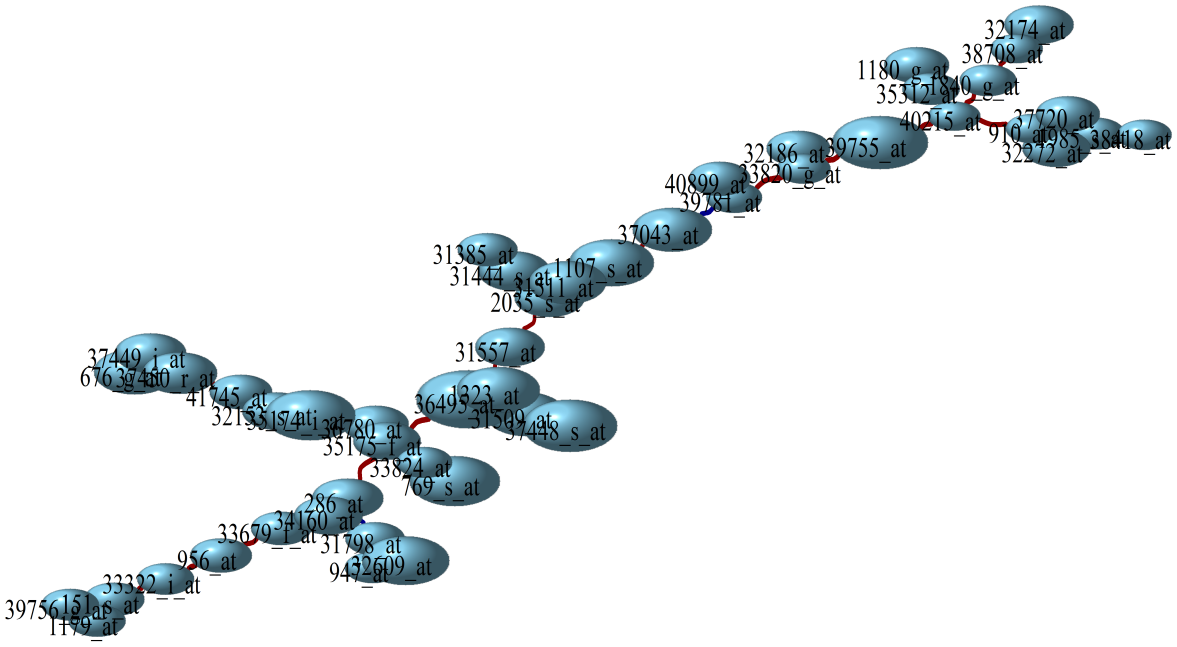

Figure 14: Initial differential network

Having acquired the above graphical model, another common analysis is performed. That is, any communities within the graphical model are clustered according to their edge betweenness. The results of this are shown below in Figure 15.

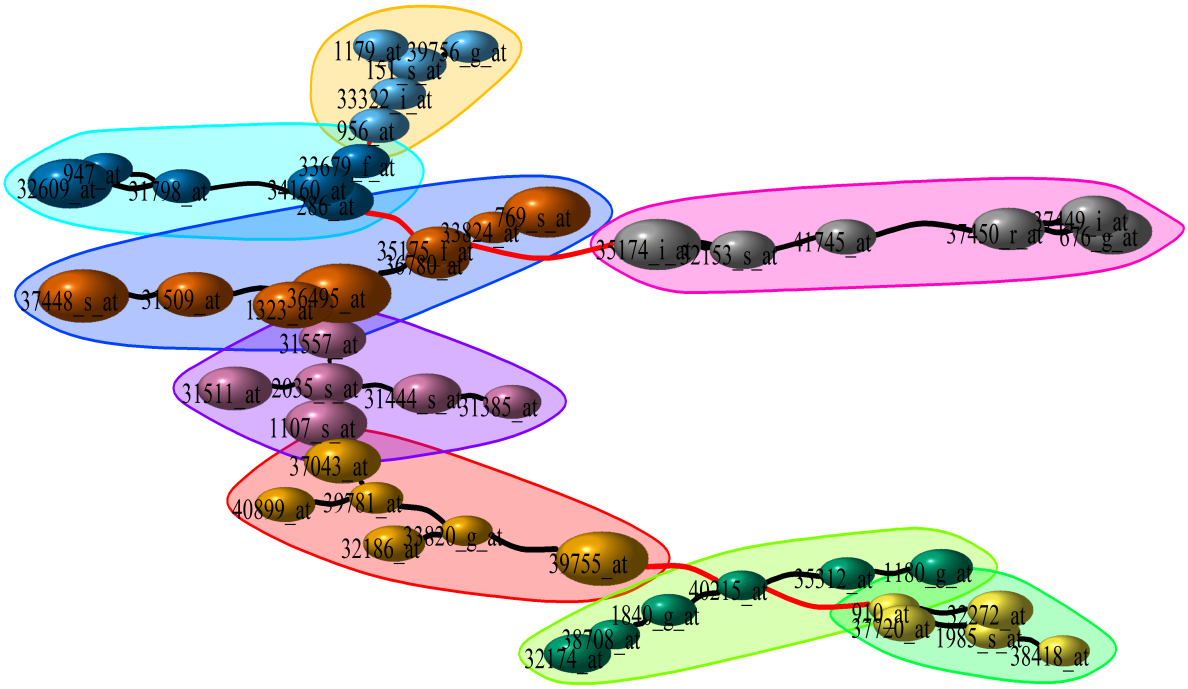

Figure 15: Clustered differential network

Having obtained the above differential networks, typically the next step would be to perform an analysis on the network [65]. This analysis would investigate the cause as to why possible changes in specific relationships occurred, or why other relationships did not change. In the case of a disease study, the network would be examined to determine whether the disease progression is being effected by the chosen treatment, and if so to what degree. Such analyses have been performed to great success in various papers, as shown in the below figure.

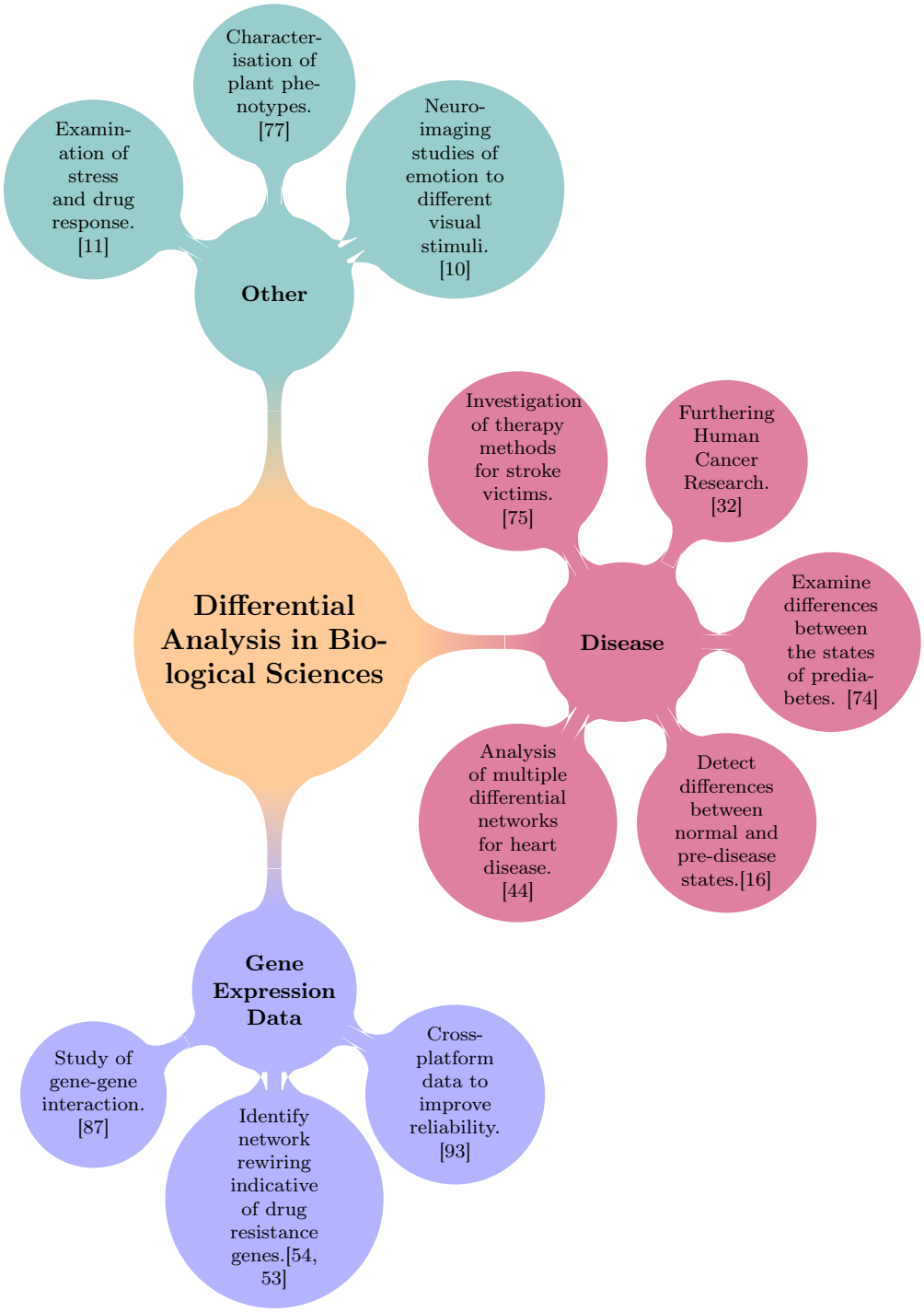

Figure 16: Applications of differential network and their analyses in biological sciences

Analyses as those done in each of the papers mentioned in Figure 16, rely heavily on background knowledge regarding the specific science of interest. For example, to draw analyses on the networks shown in Figure 14 or Figure 15, prior knowledge of genetics would be required. Within this dissertation, it is thus not possible to mention every method of analysis nor is it the intention. There are however, several *universal* methods of analysis that may be applied to any graphical model that is of interest.

2.3 Differential Network Analysis

Differential networks, like all graphical models, can be analysed and investigated through the use of a number of different statistical methods [65]. These methods can be categorized into two broad groups, those which examine the network globally, and those which examine local components of the network [65]. Global methods, are calculated over the entire network, and give a holistic summary of the relationships which characterize the network, while local methods are only determined for a particular node, neighborhood of nodes, or some subset of the network, sub-network, and provide insight into a snapshot of the network. Each of these measures of analysis have their own respective merits, however more often then not, they should be calculated in tandem to produce the most comprehensive analysis [65].

Arguably, the most popular global measure for analysing networks is the structural Hamming distance. The Hamming distance by definition is simply the count of the number of edges that differ between any two networks [65]. Hence, the Hamming distance, like all global measures, provides the answer to whether networks $\mathbf{N}_1$ and $\mathbf{N}_2$ are equal, and this is the case only when the Hamming distance is zero, as then there are no differences in the edges of the two networks. Other common global measures include exploring weighted differences [65] or even measures such as those which consider the degree of distribution, edge density, average degree, shortest path, diameter, average path length, average connectivity, clustering coefficient, cluster size, number of cluster and also the centrality of the network, for further details on how these measures can be calculated see the work of Shojaie and Sedaghat [66]. Having considered the above global measures, it is possible to extend them to obtain local equivalents. That is, each of the above global measures can be converted to a local measure by calculating the measure over only a particular subset of the nodes and or edges within the network. Local measures are extremely predominant within disease progression studies, as often only a few features are of interest as they are strongly related to the actual development and the ensuing treatment of the disease [65].

Thus far, the methods considered for analysing differential networks are all quantitative in nature. Recent works however, have shown that these measures can yield undesirable results, and produce false positives when aiming to identify differences within the network structures [65]. Thus, new frameworks have been recently developed in the form of discriminant connectivity analysis to address said issues, but also provide the connectivity patterns between nodes, which are often of greater scientific value

[65]. Although differential connectivity analysis might yield several advantages over standard differential network analysis, it is far more testing to implement as well as reliant on further assumptions [65]. Differential connectivity analysis remains as an area of possible future work, while the focus of this dissertation now switches. That is, specific analyses of differential networks are not discussed beyond this point within the dissertation, but the main objective of the dissertation is now examined. The intention is now to compare the various estimation techniques for $\mathbf{\Delta}$, taking into account various conditions, such as normality, dimension, sparsity and the accuracy of the estimated differential network obtained.

3 Optimisation and Estimation

As discussed within Section 1.3, there are two main specifications that must be made in order to estimate a differential network. Namely, an optimization algorithm must be selected and an objective function determined on which the optimization will be performed. Before specifics of the algorithm and objective functions used to estimate a differential network are provided, optimization as a whole is first introduced. Any optimization problem has the following mathematical representation:

$$\text{minimize} \quad f_0(x) \tag{8}$$

$$\text{subject to} \quad f_i(x) \leq b_i, \quad i = 1, ..., m. \tag{9}$$

The above representation, then consists of several components, namely equation 8 represents the objective function and equation 9 represents the constraint functions that must be satisfied [8]. Optimization has many sub-classes, for example linear optimization refers any optimization problem in which both the objective functions, and constraint functions are linear in nature. Non-linear optimization is another popular sub-class of optimization which describes problems in which one of the objectives functions, or constraint functions are non-linear. The category of optimization focused upon within this dissertation is convex optimization. A convex optimization problem can then be represented mathematically as follows:

$$\text{minimize} \quad f_0(x) \tag{10}$$

$$\text{subject to} \quad f_i(x) \leq b_i, \quad i = 1, ..., n, \tag{11}$$

where the functions $f_0, ..., f_n$ are all convex. That is:

$$f_i(\alpha x + \beta y) \leq \alpha f_i(x) + \beta f_i(y) \tag{12}$$

for all $\alpha, \beta \in R$ where $\alpha + \beta = 1$ with $\alpha, \beta \geq 0$ and $x, y \in R^n$ [8]. There exist many well known

functions that are convex in nature such as those shown in Figure 17.

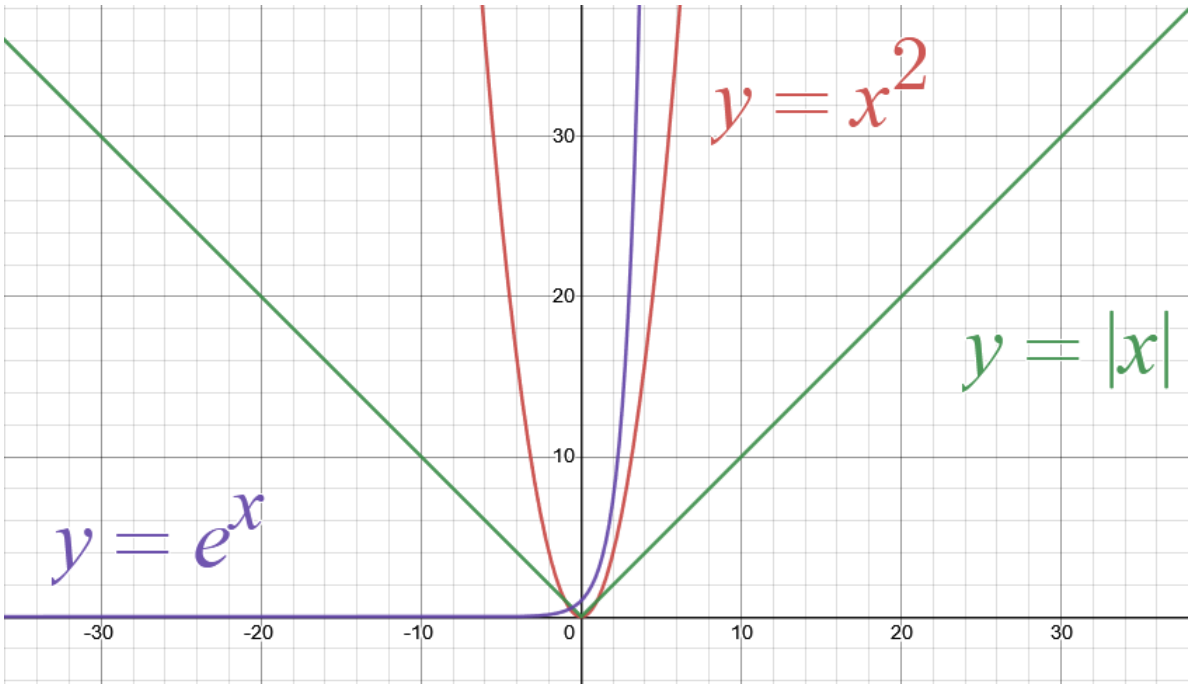

Figure 17: Common convex functions

Each of the functions shown in Figure 17 are convex on the domain $x \in (-\infty, \infty)$, however there exist functions such as $y = \frac{1}{x}$ which are convex on a subset of their domain and are concave elsewhere. For the purposes of this research, only functions which are convex in nature across the entirety of their domain will be considered. The motivation behind focusing upon convex optimization within this dissertation stems from the fact that convex optimization problems very rarely if ever have analytical solutions, convex optimization problems do however, have very efficient computational means of arriving at an approximate solution [9]. Koenker and Mizera [41] put it extremely well that, "without convexity we risk wandering around in the wilderness always looking for a higher mountain, or a deeper valley. With convexity we can proceed with confidence toward a solution". As such, convexity removes the risk of the optimization algorithm confusing any local minima or maxima for the global saddle point of interest. It is these computational methods that will be used to obtain differential network estimates, as in high-dimensional settings the inverse covariance matrices required to otherwise obtain an analytical estimate do not exist. Thus, having introduced the general form of optimization problems and the particular form focused upon within this work, it is now important to consider the framework that will be used to solve such problems.

3.1 Alternating Direction Method of Multipliers and it's Preliminaries

There currently exists a vast number of optimization algorithms capable of solving convex optimization problems, with each algorithm having a particular advantage and use case in which it should be preferred

over other alternatives. However there is currently only one optimization algorithm widely implemented in the estimation of differential networks. The algorithm is that of alternating direction method of multipliers, ADMM. ADMM is favoured for such applications due to the algorithm's excellent performance in high-dimensional settings while remaining extremely simple mathematically and computationally as will later be shown [9]. ADMM has also shown the ability to solve distributed problems, in which the data in question is spread across multiple storage locations, with efficiency equivalent to that of other state-of-the-algorithms [9]. ADMM is by no means the most powerful algorithm available, and for any particular problem there exists an algorithm capable of outperforming ADMM but it is the flexibility of ADMM to solve a wide range of problems all extremely well that make it useful for differential network estimation [9]. That is, ADMM is able to approximate a differential network in low-dimensional, high-dimensional, and even ultra high-dimensional settings with differing degrees of sparsity present and little assumptions regarding the loss function at hand [9].

Having introduced why ADMM is the optimization of choice throughout this dissertation, it is now important to define the optimization scheme. ADMM is said to be the conjunction between two popular procedures, namely the method of multipliers and dual ascent as advantages of each approach are present within ADMM [9]. Thus, before introducing ADMM these precursors will be discussed.

3.1.1 Augmented Lagrangian with Dual Ascent

Consider, a general convex optimization problem as introduced earlier, that is:

$$\text{minimize} \quad f(x) \tag{13}$$

$$\text{subject to} \quad Ax = b \tag{14}$$

where f is a convex function. The above problem can then be rewritten into two common components seen throughout optimization. That is, the problem can be broken down into a *primal* element and a *dual* element. Each iteration of the optimization then involves the solving of the dual problem first, and then the use of the solution of the dual to aid in the solving of the primal [9]. The motivation behind this restructuring of the problem arises from the fact that it simplifies the problem at hand by breaking the single problem into two slightly simpler sub-problems. More specifically, the solution of the dual problem is a lower bound, which eases the computation cost of the subsequent solving of the primal. In terms of complexity, the dual is often magnitudes easier to solve than the primal, hence providing any user a less demanding point of entry into the problem at hand [9]. There is a great deal of literature on how primal and dual elements can and should be selected, however as this research explores existing optimization schemes with well-defined primal and dual functions, the inspiration behind the specific selections will not be explored.

The problem described by equations 13 and 14 can then be represented as a dual Lagrangian problem whose primal function is:

$$L(x, y) = f(x) + y^T(Ax - b) \tag{15}$$

and whose dual is:

$$g(y) = \inf_x L(x, y) = -f^*(-A^T y) - b^T y, \tag{16}$$

where f^* is a convex conjugate of f and y is traditionally referred to as the Lagrange multiplier but is more commonly referred to as the dual variable. In order to solve the above dual Lagrangian problem, one must first resolve the dual problem. This is done as follows:

$$\text{maximize} \quad g(y). \tag{17}$$

The optimal primal point x^* can be obtained from the optimal dual point y^* by solving the following

$$x^* = \underset{x}{\text{argmin}}\ L(x, y^*) \tag{18}$$

Thus far, within this section, only how the original convex problem can be reformulated has been provided.

The next point of discussion is then how one can go about solving such a problem. The dual ascent method is one technique capable of just that [9]. The basic principle of dual ascent is to utilize the gradient of a function to obtain convergence. That is, the gradient of the function is used to ensure the algorithm proceeds in each iteration in the correct direction towards the global maximum. This process, referred to as gradient ascent requires g to be differentiable such that it's gradient, $\nabla g(y)$ exists [9]. This gradient, is trivially determined by inspecting equation 16 such that $\nabla g(y) = Ax^+ - b$, from which it is clear that in order to solve for the gradient, $x^+ = \underset{x}{\text{argmin}} L(x, y)$ must first be solved [9]. Dual ascent can be summarized as the following algorithm:

Algorithm 1: Dual Ascent

1 Initialise $x^+ = \underset{x}{\text{argmin}}\ L(x, y)$;

2 **repeat**

3 $\quad x^{k+1} := \underset{x}{\text{argmin}}\ L(x, y^k)$;

4 $\quad y^{k+1} := y^k + \alpha^k(Ax^{k+1} - b)$;

5 **until** *convergence*;

where $\alpha_k \geq 0$ is said to be the step-size of iteration k and represents the quantity by which the optimization can tend to the global minimum in any one iteration [9]. By further examining the above algorithm, the origins of the name dual ascent become clear. Although, the method is one of minimization, the dual variable is increased in each iteration shown in step 4 above. However, recalling that the dual

problem is actually a lower bound for the primal problem, as the scheme iterates the lower bound increases i.e *ascends* to a maximum at which the global solution is obtained. In summary dual ascent performs minimization, by indirectly obtaining a maximum for the dual problem [9]. Dual ascent is then an extremely powerful optimization scheme, with a wide-variety of uses but it is not without it's shortfalls. Dual ascent is extremely sensitive to several factors, amongst others the differentiability of g, the convexity of f and the choice of α_k [9]. Similarly, because the original problem has been broken down into two sub-problems, convergence is often very slow. A more efficient, robust alternative is thus required.

3.1.2 Augmented Lagrangian with Method of Multipliers

An interesting question to consider having defined dual ascent is that, why split the original problem as described by equations 13 and 14 into a primal and dual component instead of using the gradient of the objective function to obtain a solution. Especially given dual ascent's stringent assumptions and slow convergence. The answer is simply, by making use of the decomposition of a problem as described by dual ascent the user gains the ability to perform distributed computing [9]. Distributed computing, also referred to as decentralized optimization, is extremely valuable in today's age as a consequence of the ever increasing challenges surrounding modern data. Often times, it may be too costly to compile large datasets on a single storage device, or regulatory restrictions may not allow for data to leave it's territory of origin. In such scenarios, the only manner in which a global solution can be obtained is through the use of distributed computing in which solutions are obtained for each subset of the data, and then aggregated to obtain a final solution [8]. Dual ascent, is then an algorithm which addresses exactly that but due to the method's weaknesses as well as scenarios in which distributed computing may not be necessary, a more robust alternative is required. One such alternative is the method of multipliers.

As in the case of the dual ascent, the method of multiplier algorithm applies Lagrangian techniques to restructure the original problem [9]. The main difference between dual ascent and method of multipliers, is that while dual ascent incorporates a dual Lagrangian into the problem framework, method of multipliers utilizes the augmented Lagrangian ideology [9]. To see this difference, once again consider the general convex optimization as illustrated below:

$$\text{minimize} \quad f(x) \tag{19}$$

$$\text{subject to} \quad Ax = b \tag{20}$$

To circumvent the assumption of strict convexity required in dual ascent, method of multipliers considers

the following augmented Lagrangian:

$$L_p(x, y) = f(x) + y^T(Ax - b) + \left(\frac{\rho}{2}\right)||Ax - b||_2^2, \tag{21}$$

where $||X||_2 = \left(\sum_{i=1}^{n}\sum_{j=1}^{m} x_{ij}^2\right)^{0.5}$ and ρ is a penalty parameter [8]. Using the augmented Lagrangian, equations 19 and 20 can then be rewritten as follows:

$$\text{minimize} \quad f(x) + \left(\frac{\rho}{2}\right)||Ax - b||_2^2 \tag{22}$$

$$\text{subject to} \quad Ax = b \tag{23}$$

By comparing the problem described by equations 19 and 20 with the problem characterized by equations 22 and 23 it is clear the two problems are equivalents as when $Ax = b$, the second term in equation 22 is simply zero. Having restructured the problem through the use of augmented Lagrangian the problem can then be solved. The manner in which this is done is nearly identical to that of dual ascent, with the new algorithm being described below:

Algorithm 2: Method of Multipliers

1 Initialise $x^+ = \underset{x}{\text{argmin}}\ L_p(x, y)$;

2 **repeat**

3 $\quad x^{k+1} := \underset{x}{\text{argmin}}\ L_p(x, y^k)$;

4 $\quad y^{k+1} := y^k + \rho(Ax^{k+1} - b)$;

5 **until** *convergence*;

The algorithm has the same overall structure and steps as dual ascent, with the only significant difference occurring in step 3 in which L_p is now used rather than L. This change, although subtle has a number of significant advantages over dual ascent. That is, method of multipliers is applicable in circumstances where the function f is convex, unlike dual ascent which requires f to be strictly convex [9]. Similarly, dual ascent relies heavily on the differentiability of the dual function $g(y)$, but $g_p(y) = \inf_x L_p(x, y)$ the dual function under the method of multipliers is differentiable under a greater set of conditions making the approach more robust.

It appears then that the method of multipliers has addressed several of the issues present within dual ascent, however this has come at a cost. The cost being that method of multipliers can not be used to perform distributed computing as unlike the dual Lagrangian, the augmented Lagrangian function is not separable [9]. Ideally, there would be an algorithm with robustness equal to that of method of multipliers as well as the decentralized characteristics of dual ascent. Alternating direction method of multipliers, ADMM, is one such optimization procedure that attempts to incorporate the advantages of these two models into a single scheme [9].

3.1.3 Alternating Direction Method of Multipliers

ADMM, briefly can be thought of as the union of the dual ascent and method of multipliers algorithms above and as such inherits their best qualities. ADMM is a powerful, flexible and robust optimization scheme that is immune to violations to a variety of the assumptions required for dual ascent [9]. Once again, the algorithm is not without it's shortfalls. Due it's flexibility, ADMM is very general and as such the computational complexity of the approach is extremely high. In turn, this can make ADMM convergence rates appear exceedingly poor when compared to alternatives [9]. What ADMM does provide, that strongly motivates it's use for the estimation of differential networks despite the slower rates of convergence is the algorithm's flexibility [69]. ADMM has been shown to converge in low, high and even ultra high-dimensional settings, while remaining unaffected by differing sparsity and covariance structures for a variety of loss functions [9]. This makes ADMM the perfect algorithm for differential network estimation, as there exist a wide variety of sparsity levels and dimensionality structures of interest across the various sciences in which differential networks are of use. ADMM thus solves a constrained convex optimization problem, shown in equations 19 and 20 by solving, a different yet equivalent problem [9]:

$$\text{minimize} \quad f(x) + g(z) \tag{24}$$

$$\text{subject to} \quad Ax + Bz = c \tag{25}$$

The idea is then similar to dual ascent, in which the original problem is broken down into two, slightly simpler sub-problems except that rather than introducing an additional function directly ADMM introduces an auxiliary variable z. It remains assumed that f is convex, but that so too is g [9]. The solution to the above problem is then found through the use of the augmented Lagrangian which is now:

$$L_p(x, z, y) = f(x) + g(z) + y^T(Ax + Bz - c) + \left(\frac{\rho}{2}\right)||Ax + Bz - c||_2^2, \tag{26}$$

which is then solved through the following steps:

Algorithm 3: Alternating Direction Method of Multipliers

Initialise $x^+ = \underset{x}{\text{argmin}}\ L_p(x, z, y)$;

repeat

  $x^{k+1} := \underset{x}{\text{argmin}}\ L_p(x, z^k, y^k)$;

  $z^{k+1} := \underset{z}{\text{argmin}}\ L_p(x^{k+1}, z, y^k)$;

  $y^{k+1} := y^k + \rho(Ax^{k+1} + Bz^{k+1} - c)$;

until *convergence*;

By examining the above algorithm, it is evident the same formula present within dual ascent and method of multipliers still holds. That is, there is a primal component, in this case two variables x and z respectively, while y is the dual variable as before. The name alternating direction method of multipliers, then stems from the fact that the primal variables are not updated simultaneously, but rather in a sequential manner, first x then z, then x again alternating until convergence [9]. For further details regarding the theoretical properties of ADMM such as the method's convergence rates, the reader is referred to the work of Boyd et al. [9]. The focus of this dissertation, now becomes to adjust the standard convex optimization problem above into one which performs differential network estimation with the aid of ADMM.

3.2 Loss Functions

For the purposes of this dissertation, there are now four loss functions which will be considered. The d-trace loss function of Yuan et al. [90], the graphical LASSO loss function developed by Friedman et al. [28], as well as a SCAD and MCP modification thereof. These loss functions, although some of which are investigated in Tang et al. [69], differ as none of the functions utilized for the purpose of this research allow for an asymmetric estimate of the differential network, Δ. The motivation for this, is that a differential network as defined earlier, is the difference between two inverse covariances matrices, and as symmetry is preserved by inversion, Δ should consequently also be symmetric [65, 69]. Lastly, for conciseness none of the derivations of these loss functions are included with this dissertation as such results are readily available [69, 90, 60, 52]. The objective is thus to describe the loss functions used for differential network estimation, and the respective ADMM scheme to minimize each function.

3.2.1 D-trace

The first loss function considered is the d-trace loss function of Yuan et al. [90]. The loss function is thus given by the following equation:

$$L_D(\Delta) = \frac{1}{4}[\text{tr}\{S_1\Delta(\Delta S_2)^T\} + \text{tr}\{S_2\Delta(\Delta S_1)^T\}] - \text{tr}\{\Delta(S_1 - S_2)\} + \lambda||\Delta||_1 \tag{27}$$

It is thus extremely evident that the above loss functions draws inspiration from the standard LASSO of Tibshirani [70] and has been shown to have excellent performance in a variety of numerical scenarios [90]. The significant difference between the d-trace loss above, and the loss functions which follow is that the d-trace relies on what is referred to as an irrepresentability condition [90]. This condition represents several assumptions regarding Δ. Namely, Δ is assumed to be a positive-definite sparse matrix, whose elements have a finite upper bound [90]. Given the stringency of these assumptions, it is clear that they may not always hold in practice and as such the performance of the d-trace loss function may falter. It is

also not possible to check the validity of such assumptions before estimating Δ and as such one may be unknowingly violating the irrepresentability condition. However, should these assumptions be met the solving of the d-trace loss function through the use of ADMM is expected to be immensely efficient [90]. In order to solve equation 27 the augmented Lagrangian is first required, which is given as:

$$L(\Delta, A, B) = L_D(\Delta) + \left(\frac{\rho}{2}\right)||\Delta - A + B||_2^2 + \lambda||A||_1 \tag{28}$$

Using the theory discussed in Section 3.1.3 the ADMM scheme for the D-trace loss function is then:

Algorithm 4: ADMM for the D-trace Loss

Initialise $\Delta^+ = \text{argmin } L(\Delta, A, B)$;

repeat

  $\Delta^{k+1} := \text{argmin } L(\Delta, A^k, B^k) = \text{argmin } L_D(\Delta) + (\frac{\rho}{2})||\Delta - A + B||_2^2$;

  $A^{k+1} := \text{argmin } L(\Delta^{k+1}, A^k, B^k)$;

  $B^{k+1} := \Delta^{k+1} - A^{k+1} + B^{k+1}$;

until *convergence*;

Although the above algorithm is rather simple and elegantly fits with the ADMM framework the issue of possible violations to the irrepresentability condition remain, and as such alternative loss functions must be explored.

3.2.2 Graphical LASSO with SCAD and MCP

As in the case of the d-trace loss function above, the graphical LASSO loss function is built upon the work of Tibshirani [70]. Friedman et al. noticed that when making use of the standard LASSO loss to perform covariance and precision matrix estimation, there were several unexpected consequences. This included poor accuracies when estimating precision matrices, as the LASSO did not directly estimate the precision matrix but rather the covariance matrix which was then inverted [48, 28]. In scenarios which are high-dimensional, this approach is of little value as the covariance matrix estimate is non-invertible regardless of whether the matrix is estimated or not. Friedman et al. purposed a slight modification to the general LASSO that was able to address the shortcoming, and has been readily extended to differential network estimation [28, 69]. The graphical LASSO loss function is then as follows:

$$L(\Delta) = \frac{1}{2}\text{tr}\{\Delta^T S_1 \Delta S_2\} - \text{tr}\{\Delta(S_1 - S_2)\} + p(\Delta), \tag{29}$$

where $p(\Delta)$ is referred to as the penalization term which is $p(\Delta) = \lambda||\Delta||_1$ for standard graphical LASSO [28]. Within this dissertation standard graphical LASSO will be used going forward, but so too will two variations. The various penalization terms considered in conjugation with graphical LASSO are summarized as follows:

- Standard penalty term, referred to as the traditional graphical LASSO penalty [28, 69]:

$$p(\beta) = \lambda||\beta||_1$$

- SCAD penalty term [26, 60]:

$$p(\beta) = \begin{cases} \lambda|\beta| & \text{if } |\beta| \leq \lambda \\ \frac{2a\lambda|\beta| - \beta^2 - \lambda^2}{2(a-1} & \text{if} \lambda < |\beta| \leq a\lambda \\ \frac{\lambda^2(a+1)}{2} & \text{otherwise.} \end{cases}$$

- MCP penalty term [26, 60]:

$$p(\beta) = \begin{cases} \lambda|\beta| - \frac{\beta^2}{2a} & \text{if} |\beta| \leq a\lambda \\ \frac{1}{2}a\lambda^2 & \text{if} |\beta| > a\lambda \end{cases}$$

where a is referred to as a thresholding parameter, and β is the parameter undergoing estimation. Each of the above penalties behave in a slightly different manner, but have the same overall objective to reduce the effect of large values of β so as to improve the bias of any possible parameter estimate [26]. The manner in which each of these penalties does the above, can be visualized as follows:

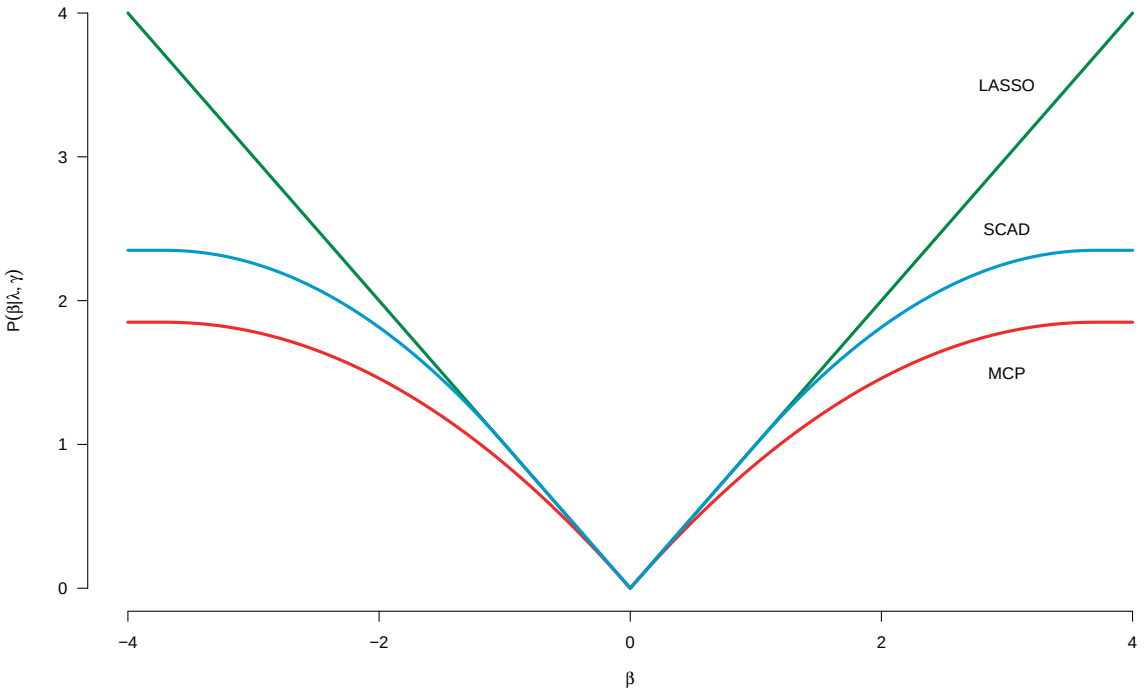

Figure 18: Behaviour of various penalty functions

From Figure 18 the varying degrees of penalization enforced by each of the different penalty functions is clearly evident. Having defined the different penalty terms which may be used within a graphical LASSO loss function, it is now important to consider the ADMM steps to solve such a function. The augmented Lagrangian for the graphical LASSO loss is then as follows [69]:

$$L(\Delta, A, B) = L(\Delta) + \left(\frac{\rho}{2}\right)||\Delta - A + B||_2^2, \tag{30}$$

where $L(\Delta)$ is as in equation 29. Hence, in order to solve the above Lagrangian the general framework of ADMM is utilized, with the algorithm given as [69]:

Algorithm 5: ADMM for the graphical LASSO loss

Initialise $\Delta^+ = \text{argmin } L(\Delta, A, B)$;

repeat

  $\Delta^{k+1} := \text{argmin } L(\Delta, A^k, B^k) = \text{argmin } L(\Delta) + \left(\frac{\rho}{2}\right)||\Delta - A + B||_2^2$;

  $A^{k+1} := \text{argmin } L(\Delta^{k+1}, A^k, B^k)$;

  $B^{k+1} := \Delta^{k+1} - A^{k+1} + B^{k+1}$;

until *convergence*;

Above is then an algorithm which can efficiently solve the graphical LASSO loss function to arrive at an estimate for a differential network making use of either one of the three different penalty functions considered.

3.3 Nonparanormal Transformations

There remains one final consideration before proceeding with the application section of this dissertation. That is, as the estimation of differential networks relies on Gaussian graphical models it is assumed that the underlying data considered is normal [89]. In practice, when working with non-simulated data this is an assumption which will almost always not hold. There thus exists a significant need in order to overcome violations to this assumption, and the ability to relax the assumption to a certain degree. One way to do this, is through what is referred to as a nonparanormal, semi-parametric Gaussian copula, transformation. The idea is that by replacing the variables $X_1, X_2, ..., X_p$ with a transformed version $f(X_1), f(X_2), ..., f(X_p)$, given the correct transformation f, $f(X_1), f(X_2), ..., f(X_p)$ are said to follow an extension of the normal distribution - the nonparanormal distribution [42]. The result is a family of distributions that are not assumed to be normally distributed, while the association changes between variables is entirely captured by the precision matrix as required for GGMs [42]. To visualize this extension of the normal distribution, a Venn diagram of the normal family of distribution and it's related families is considered. From Figure 19 it is clear the nonparanormal family of distributions includes the normal distribution but consists of a far greater number of distributions thus making the framework described

within this research for differential network estimation applicable under a larger variety of circumstances [42, 43, 89].

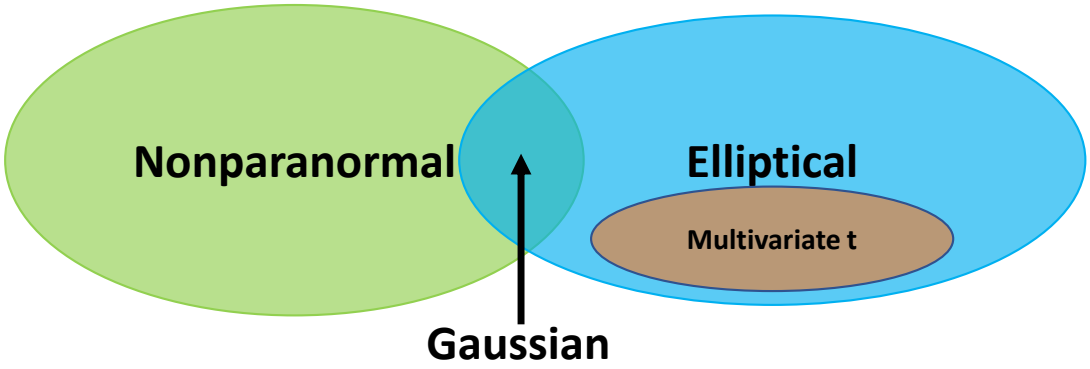

Figure 19: Venn diagram of relationships between distributions

The definition of a variable which is said to follow a nonparanormal distribution is now provided. The p-dimensional random vector X is said to follow a nonparanormal distribution if there exists differentiable and monotone functions $\{f_i\}_{j=1}^{p}$ so that $f(X) \sim N(\mu, \Sigma)$ [42]. If this criteria is met, then X is said to nonparanormally distributed represented as follows:

$$X \sim NPN(\mu, \Sigma, f). \tag{31}$$

From the above, it is clear that the choice of f is by no means trivial and there currently exist three well studied methods to transform X. The first method is the simplest and is referred to as the shrinkage method. That is, each observation is ranked according to it's size relative to all over observations in the dataset [43]. The rank of each observation is then divided by $n + 1$, where n is the number of observations, this value is then used to obtain a corresponding normal quantile value [43]. The second approach, referred to as the truncation method is near identical to the method above, except only a select few, determined through the use of a specified threshold, of the largest and smallest observations undergo the shrinkage transformation [43]. The third and final method of transformation is the skeptic transformation, in which observations are transformed through the use of the following equation:

$$f(X) = 2 * \sin\left(\frac{\pi}{6\rho}\right), \tag{32}$$

where ρ is the Spearman correlation coefficient of X. Thus, the application of any of the above transformations is non-iterative, and as such should carry an extremely minimal computation cost to the estimation of differential networks, while allowing for the stringent assumption of normality to be relaxed

to a large degree [42]. The focus of this dissertation now becomes to examine and quantify the impact that the above transformation amongst other factors has on differential network estimation.

4 Application

Upon beginning the application section of this dissertation, it became apparent that there does not exist a robust and readily available manner in which differential networks can be estimated from a programming perspective. Previous implementations such as those done by Zhang and Zou [92] and Friedman et al. [28] both rely significantly on coding experience and knowledge of the components of differential network estimation such as ADMM as both of these applications are coded from first principles. There does, however exist many packages centered around differential networks, but only upon their analysis such as those by Gill et al. [33] and Class et al. [19]. As a result, any researcher wishing to explore differential networks faces an extremely high barrier to entry as there is no accessible means to estimate a differential network. With the above in mind, as well as the growing popularity of differential networks discussed previously, the idea within this research then became to develop a R package capable of performing differential network estimation. This objective, amongst others was achieved.

The package dineR, **di**fferential **n**etwork **e**stimation in **R**, was developed throughout the course of this research. And although the development of a package is an accomplishment in it's own right, it was decided to go one step further. That is, the package was published and since approved by CRAN. CRAN, the Comprehensive R Archive Network is by no means a traditional scientific journal with an impact factor, however CRAN is seen by many as the senior authority on R packages [35]. As such, getting approval from CRAN is an extremely difficult task as many coding standards and practices must be met, all of which dineR met.

dineR is built entirely in R and thus completely open-source, freely available and capable of running on Windows 10, Windows 11, MAC and Linux. In terms of accessibility, the package is available directly from CRAN - here but also natively in RStudio. The package relies on only 3 other R packages, none of which are directly involved in the estimation procedures themselves. As such, any and all estimation done through the use of the package is extremely efficient. However, arguably the most significant contribution of dineR is the ease at which a differential network can be estimated. That is, a user need only specify the two different data samples representative of the states under investigation. Several of the package's capabilities are now discussed in further detail.

4.1 dineR Details

As mentioned previously, dineR was built entirely in R which is an interpreted programming language. This means that in order to execute any R code, or code from any interpreted programming language

for that matter, the code must first be translated to machine code before it is executed. As such, the package is not as fast as it would have been if it were to have been built in a complied programming language like C++ which is complied directly to machine code. However, what the package loses in speed is extremely marginal compared to what it gains in terms of ease of use, readability of the code and code development time. This will be shown in various of the functions available with the package.

4.1.1 Data Generation and Transformation

Within dineR, a function has been provided to generate data. The motivation behind providing such a function stems from the fact that simulations are arguably the most robust and reproducible manner to benchmark the estimation methods that follow. Secondly, should any user wish to explore the capabilities of the package and differential networks as a whole, providing a quick and easy way to generate appropriate data allows for exactly that. The function is then as follows:

```
data_generator(n, p, Delta = NULL, case = "sparse", seed = NULL),
```

while the arguments of the function are then described as:

- n - The number of observations.
- p - The number of features.
- *Delta* - An optional parameter that allows the user to provide a differential network, that will be used to obtain the sample covariance matrices.
- *case* - An optional parameter that allows the user to specify under which case they wish the covariance matrices to be determined. Options are "sparse" or "asymsparse".
- *seed* - An optional parameter that allows the user to make any data generation reproducible.

The function thus generates two multivariate normal samples of size $n \times p$. For these samples, the covariance structures are defined through the specification of an a-priori differential network. If no network is provided, the default is then as follows:

$$\Delta^* = \begin{pmatrix} 0 & -1 & 0 & \cdots & 0 \\ -1 & 2 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{pmatrix}$$

In practice, there would never be a need to estimate the differential network, if the network is known a-priori. However, the idea within this simulation is to have the differential network known so that the

accuracies of the estimation procedure can be investigated and benchmarked. Using either the default or user provided differential network, the sparsity of the sample covariance matrices can then be adjusted through the *case* parameter. That is, if the case selected is the sparse instance, then $\Sigma_1^{-1} = \Omega_1$ has the following properties:

- $\Omega_{1_{1,1}} = \Omega_{1_{p,p}} = \frac{4}{3}$,
- $\Omega_{1_{i,i}} = \frac{5}{3}$ when $i \neq 1$ and $i \neq p$,
- $\Omega_{1_{i,j}} = \frac{2}{3}$ when $|i - j| = 1$,
- $\Omega_{1_{i,j}} = 0$ otherwise.

If the case selected is the asymsparse option, which represents asymptotically sparse, then Ω_1 has the following property:

- $\Omega_{1_{i,j}} = 0.5^{|i-j|}$ for all values of i and j.

The data generating function although useful, is merely an auxiliary function as any meaningful estimation of differential networks will require real-world data. Real-world data, however will almost always not follow a normal distribution as is required by one of the early assumptions in order to make use of the Gaussian graphical model framework. The next function aids in the relaxation of this particular assumption should the data in question not follow a multivariate normal distribution.

4.1.1.1 Nonparanormal Transformations

Making use of the theory as discussed by Liu et al. [42], Xue and Zou [89], Liu et al. [43] which is mentioned in detail in Section 3.3 the following function provides a means to transform non-normal data into that of nonparanormal data through the use of three different methods. The function is then as follows:

```
npn(x, npn_func = "shrinkage", npn_thresh = NULL, verbose = TRUE),
```

while the arguments of the function are then described as:

- x - The multivariate non-normal data to be transformed.
- npn_func - An optional parameter that allows the user to specify the method of transformation. Can either be "shrinkage", "truncation" or "skeptic" as described in Section 3.3.
- npn_thresh - An optional parameter that allows the user to specify the truncation threshold that is used when making use of the truncation method of transformation.

- *verbose* - An optional Boolean parameter that controls whether additional output is provided when making use of the function.

The above function is particularly necessary, as a violation to such a key assumption such as normality would have otherwise unpredictable effects on the estimation, and make any subsequent analysis of the estimate obtained of little to no value. The next step, now that data has either been appropriately generated or transformed is to then perform the estimation in question.

4.1.2 Estimation

The function discussed within this subsection, is by far the most significant contribution made by dineR. The actions performed by the functions discussed thus far, can each be completed with rather simple code, requiring very little code development time. However, the procedures automated by the *estimation* function are by no means trivial. That is, the function specifies the loss function, performs ADMM optimization for a variety of values for the regularization/penalization term, λ, and allows for the tuning of λ through the use of either the Akaike information criterion, Bayesian information criterion or even the extended Bayesian information criterion. The function is then as follows:

```
estimation(X, Y, lambdas = NULL, lambda_min_ratio = 0.3, nlambda = 10, a = NULL,
                 loss = "lasso", tuning = "none", perturb = FALSE, stop_tol = 1e-5,
                 max_iter = 500, correlation = FALSE, Delta_init = NULL, rho=NULL,
                 gamma=NULL),
```

while the arguments of the function are then described as:

- X - The first multivariate normal sample, i.e for group 1 or state 1 of interest.
- Y - The second multivariate normal sample, i.e for group 2 or state 2 of interest.
- *lambda* - An optional parameter that specifies the regularization values of λ to be considered within the loss functions.
- *lambda_min_ratio* - An optional parameter that defines the smallest regularization value considered, as a portion of the largest regularization value.
- *nlambda* - An optional parameter that determines the number of regularization values to be considered.
- *a* - An optional parameter that allows the user to control the thresholding parameter used within the SCAD and MCP loss functions respectively.

- *loss* - An optional parameter that the user can use to alternate between the 4 different loss functions available.

- *tuning* - An optional parameter the user can specify to select a tuning scheme.

There exist 7 additional parameters, all of which are optional, available to the user. These additional parameters simply allow for greater control over the optimization process, such as the maximum number of iterations, stop tolerance or the ability to print a summary of the estimation results. Thus, having touched on the pertinent parameters available to any user of dineR, it is important to note that further details on each of the functions, their parameters, and outputs are all available within the package's documentation visible on CRAN, or GitHub.

4.1.3 Selection and Tuning

As mentioned above, dineR is capable of performing tuning to aid the user in selecting the best estimate from the various estimates obtained for the different values of λ. There exist three different built in methods to tune the model. That is through the well-known methods of AIC and BIC, however throughout the course of this research it was encountered that these two criterion did not always agree and as such a more sophisticated alternative was required. The extended Bayesian information criteria, EBIC, of Chen and Chen [14] was such an alternative. The AIC and BIC, although commonly used throughout a variety of settings in statistics have been shown to perform poorly in high-dimensional settings [14]. The EBIC has however been found to have excellent performance in ultra high-dimensional scenarios, with the consistency of the measure even been proven as the number of features tends to infinity [14]. In addition, the simplicity of the AIC and BIC often mean they are implemented despite violations to their required conditions for validity [14]. As such, the EBIC is an extremely attractive alternative even before taking into account it's superb ability in highly collinear scenarios as well as in sparse settings [15]. Having discussed the advantages of the EBIC it is important to note that the method is not without fault. The major difficulty being that within the EBIC there exists a tuning parameter, γ, that must be selected [14, 15]. This is extremely counter-intuitive as the motivation behind deploying these criterion is to tune the original estimation model to find the *best* λ. There does however exist an industry standard that has shown favourable behaviour under a large number of numerical settings, that is set $\gamma = 0.5$ [15]. This is the standard used within dineR, however should the user wish to experiment with alternatives the option is present within the package.

As the reasoning behind the inclusion of EBIC has now been provided, specifics regarding the measure are now introduced under the assumption that reader is familiar with the specifics of the AIC of Akaike [1] and BIC developed by Schwarz [64]. Assume y_i and x_i represent independent observations for $i = 1, ..., n$, then the conditional density of y_i on x_i is $f(y_i|x_i, \theta)$ with θ a p-dimensional vector of model parameters

and p some positive integer. By definition, the likelihood function of θ follows as:

$$L(\theta) = f(x;\theta) = \prod_{i=1}^{n} f(y_i|x_i,\theta). \tag{33}$$

The BIC of Schwarz is then derived as follows:

$$\text{BIC} = -2\log L(\hat{\theta}(s)) + v(s)\log(n), \tag{34}$$

where $s \subset \{1,...,p\}$, $v(s)$ is the cardinality of s, that is the number of components in s, and $\theta(s)$ is the set of parameters from θ who are outside the set s. However, to determine the first component in the above equation, a Laplace approximation is required [14], and as the name suggests BIC relies on Bayes theorem. That is, BIC relies on a prior density to determine the posterior probabilities of s. The precision of the Laplace approximation has been shown to be extremely sensitive to the prior density, and in high-dimensional settings this results in BIC selecting far too many parameters [14]. This is due to the fact that models with more parameters, always have higher posterior probabilities [14]. For the purposes of this dissertation, it is thus not of great value to have a method that says the model with more parameters, the differential network with the least zero entries, is simply best. The manner in which the EBIC overcomes this particular shortfall is by introducing a penalty which increases as the number of parameters grows [15]. EBIC is then as follows:

$$\text{EBIC} = -2\log L(\hat{\theta}(s)) + v(s)\log(n) + 2v(s)\gamma\log p, \tag{35}$$

where $\gamma \geq 0$ [15]. Comparing equation 34 to equation 35, it is clear that the first two terms in both methods are identical. The difference between methods is thus attributed to the third term within EBIC - the penalization term. Previously, when using the BIC method, the model with the most parameters which would subsequently have the highest posterior probability would be selected [14]. However, in the EBIC this is no longer guaranteed to be the case - allowing for more parsimonious model selection. Having briefly introduced EBIC, the reader is referred to the works of Chen and Chen [14] and Chen and Chen [15] for further details on EBIC in which the theoretical validity and deviations of consistency, which are excluded from this research for conciseness, are provided.

In conclusion, dineR's capabilities as described above along with the growth of differential networks in many scientific fields combined with the rising popularity of R as a programming language as documented by the TIOBE index shown below in Figure 20 place dineR at the cutting-edge of what could be many future statistical analyses. Some such analyses are shown within the next section.

Jul 2020	Jul 2019	Change	Programming Language	Ratings	Change
1	2	↑	C	16.45%	+2.24%
2	1	↓	Java	15.10%	+0.04%
3	3		Python	9.09%	-0.17%
4	4		C++	6.21%	-0.49%
5	5		C#	5.25%	+0.88%
6	6		Visual Basic	5.23%	+1.03%
7	7		JavaScript	2.48%	+0.18%
8	20	⇈	R	2.41%	+1.57%
9	8	↓	PHP	1.90%	-0.27%
10	13	↑	Swift	1.43%	+0.31%

Figure 20: TIOBE index for programming language popularity

4.2 Implementation

4.2.1 Simulation Study

The objective is to now benchmark and analysis the performance of dineR under a variety of different environmental conditions. There are 3 main scenarios in which dineR will be implemented. The first scenario of interest is a simulation study, using the package's built-in function to generate a variety of data. Similar to the work of Tang et al. [69], the number of observations will be fixed and several different dimensionality structures will be considered. Namely, $n = 100$ and $p = 10, 20, 50, 100, 200, 500$. As mentioned previously, the data generated will follow a multivariate normal distribution, and as such there are no concerns regarding violations to the assumption of normality. However, to explore the effect on the estimation process of the nonparanormal transformation, nonparanormal data will also be generated. The impact of varying correlation structures will also be investigated by generating data of varying covariance and subsequent correlation structures. Lastly, in high-dimensional and ultra high-dimensional settings the run-times of the differential network estimation procedures can become very large, such that it may not be possible to apply each loss function in dineR. As a result, this research will aim to determine under which experimental conditions each loss function should be favoured if at all. In order to compare the respective loss functions, and the impact of the differing environment conditions the computation time of the estimation process along with the accuracy of the differential network estimate obtained will all be investigated. To investigate the accuracy of the estimate the true positive rate, TPR, will be considered. The TPR in this context is defined as [62]:

$$\text{TPR} = \frac{\#\{(i,j) : \hat{\omega}_{ij}) = 0 \quad \text{and} \quad \omega_{ij} = 0\}}{\#\{(i,j) : \omega_{ij} = 0\}}, \tag{36}$$

where ω_{ij} is an element of the true differential network, Δ, and $\hat{\omega}_{ij}$ is the corresponding element in the estimated differential network Δ^*. The differential networks under the above conditions are now estimated for 20 different values of λ, for a maximum of 500 iterations with EBIC the choice of tuning mechanism. The results for this, when comparing the competing loss functions yields the following results:

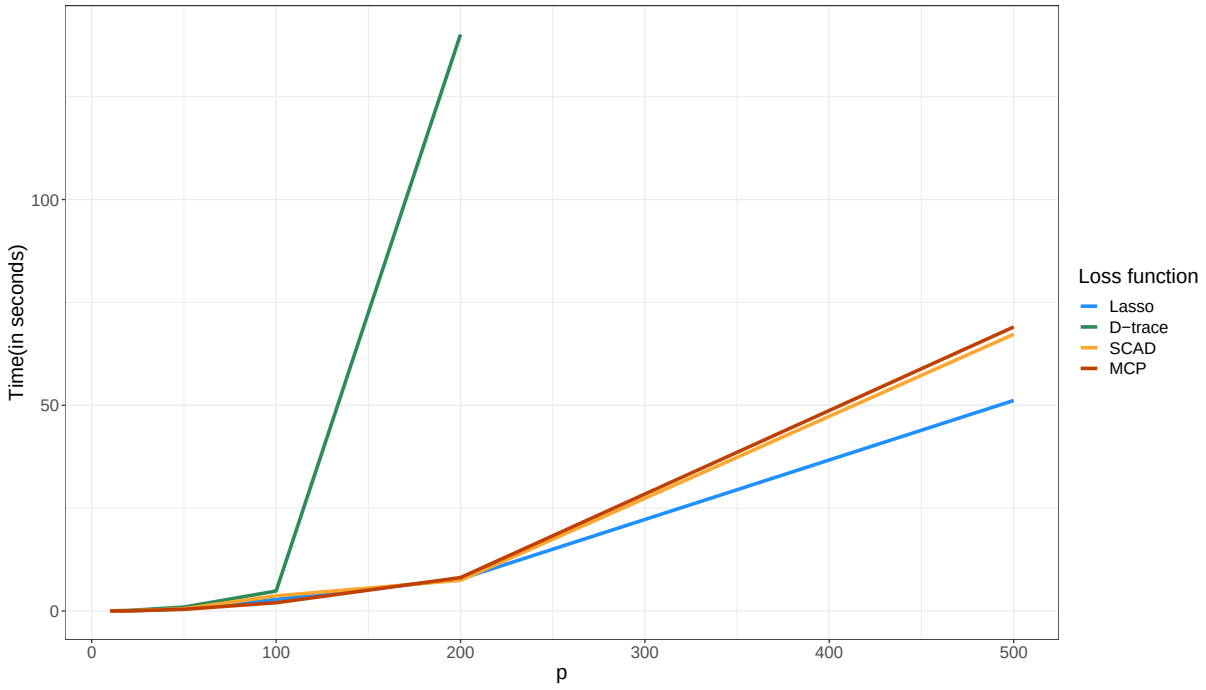

Figure 21: Computation time against dimensionality - Sparse case

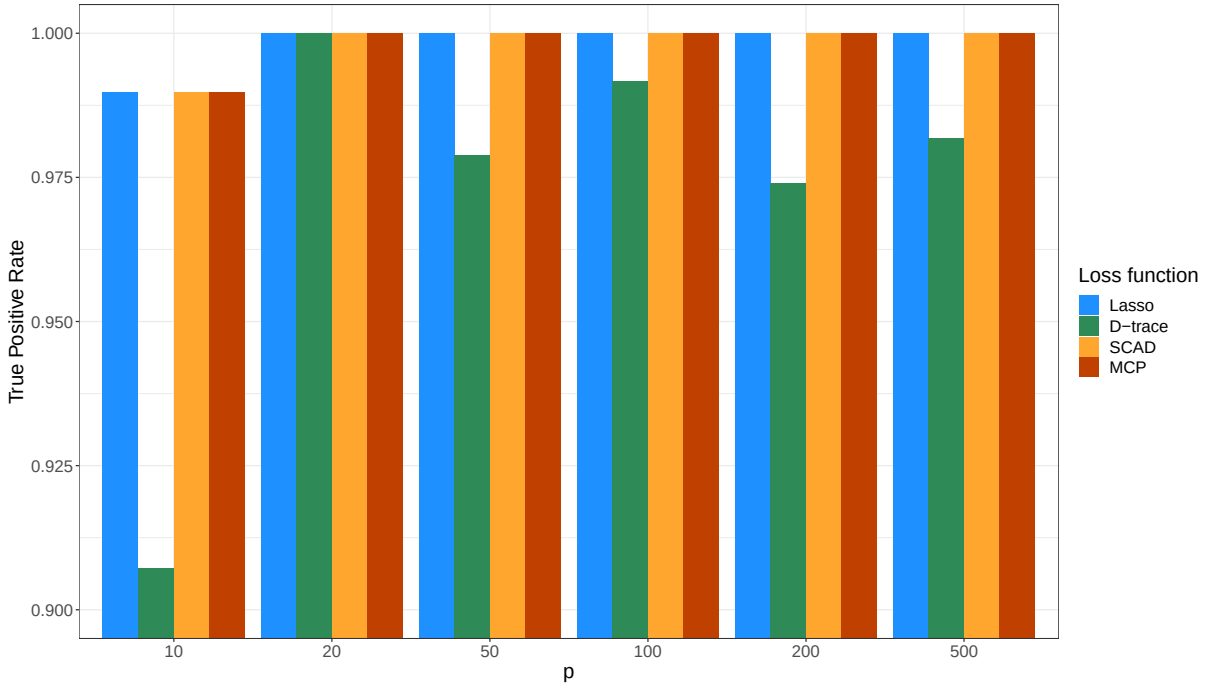

Figure 22: True positive rate against dimensionality - Sparse case

Table 1: Computation time(in seconds) - Sparse case

Dimensions	LASSO	D-trace	SCAD	MCP
10	0.05	0.06	0.03	0.03
20	0.04	0.14	0.06	0.04
50	0.51	0.96	0.53	0.42
100	2.82	4.86	3.70	2.02
200	7.79	140.13	7.47	8.13
500	51.17	10388.93	67.25	69.03

Table 2: True positive rate - Sparse case

Dimensions	LASSO	D-trace	SCAD	MCP
10	0.9896907	0.9072165	0.9896907	0.9896907
20	1.0000000	1.0000000	1.0000000	1.0000000
50	1.0000000	0.9787745	1.0000000	1.0000000
100	1.0000000	0.9915975	1.0000000	1.0000000
200	1.0000000	0.9740481	1.0000000	1.0000000
500	1.0000000	0.9817758	1.0000000	1.0000000

From Figure 21 and Table 1 it is clear that the LASSO, SCAD and MCP loss functions all appear to have equivalent computational performance with the d-trace loss function being significantly slower particularly for $p = 500$ in which the computation time was so large that for clarity it was removed from the plot. It was also of great interest as to whether the additional computational time taken by the d-trace implementation would yield additional accuracy as a result. But it is clear from Figure 22 and Table 2 that this was not the case with the d-trace loss function producing accuracies worse in all but one dimension in which the accuracy was only equal to that of the other methods. Thus, the above early signs indicate that despite the d-trace loss function having several favourable theoretical properties, it's numerical performance is well below that of it's competitors. The above results, summarize the performance for the four loss functions in which the covariance matrices are sparse, naturally the next comparison explores their performance under an asymptotically sparse covariance structures. As before, the computation times, and accuracy through the use of the TPR are investigated, and were obtained as shown below.

Table 3: Computation time(in seconds) - Asymptotically sparse case

Dimensions	LASSO	D-trace	SCAD	MCP
10	0.02	0.06	0.03	0.02
20	0.05	0.11	0.04	0.04
50	0.23	1.07	0.25	0.19
100	1.62	3.23	1.61	1.82
200	6.11	51.24	6.42	8.09
500	37.36	5451.88	37.15	44.85

From Figure 23 and Table 4 it is evident that the results under the asymptotically sparse case follow closely from the results of the sparse case. That is, the d-trace loss function illustrated far worse performance than the other loss functions, with the computation time of the d-trace loss function for $p = 200$

significantly exceeding the computation time of the LASSO, SCAD and MCP loss functions even when the number of dimensions was more the double, $p = 500$.

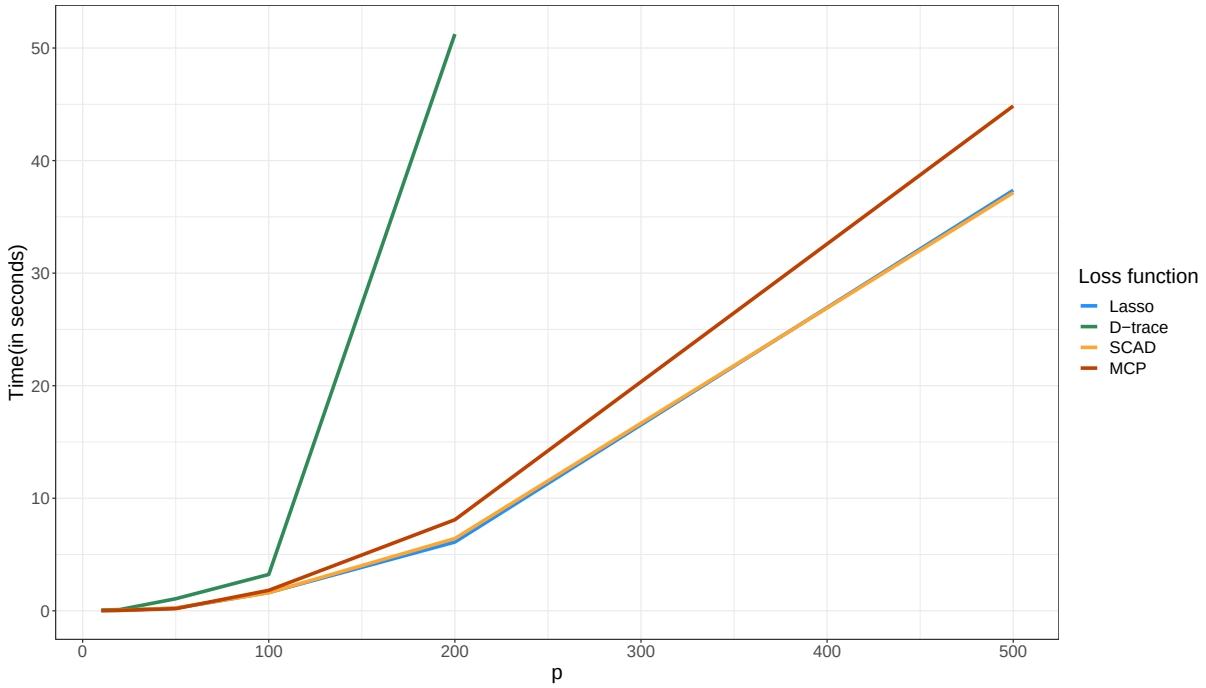

Figure 23: Computation time against dimensionality - Asymptotically sparse case

The main difference between the results for the asymptotically sparse case and those shown previously for the sparse case is that the LASSO function is not longer far and ahead the fastest loss function as the SCAD function produced near identical times for the different dimensions considered. The TPR's of the four loss functions were then obtained as:

Table 4: True positive rate - Asymptotically sparse case

Dimensions	LASSO	D-trace	SCAD	MCP
10	0.9896907	0.8453608	0.9896907	0.9896907
20	1.0000000	1.0000000	1.0000000	1.0000000
50	1.0000000	0.9991990	1.0000000	1.0000000
100	1.0000000	0.9900970	1.0000000	1.0000000
200	1.0000000	0.9887242	1.0000000	1.0000000
500	1.0000000	0.9981320	1.0000000	1.0000000

Investigating Table 4 and Figure 24 it can be seen that as the with computations time for the asymptotically sparse case, the d-trace loss function produces the worst accuracy of any of the functions considered across the variety of dimensionality structures considered. Thus far there appears to be little to no motivation for the use and implementation of the d-trace loss function given the results observed. However, as mentioned previously dineR allows several nonparanormal transformations to be applied to data that is otherwise non-normal. As with the above results, the effect of applying such transformations is now examined to determine what if any affect the transformation has on both the computation time and

accuracy of the differential network estimation.

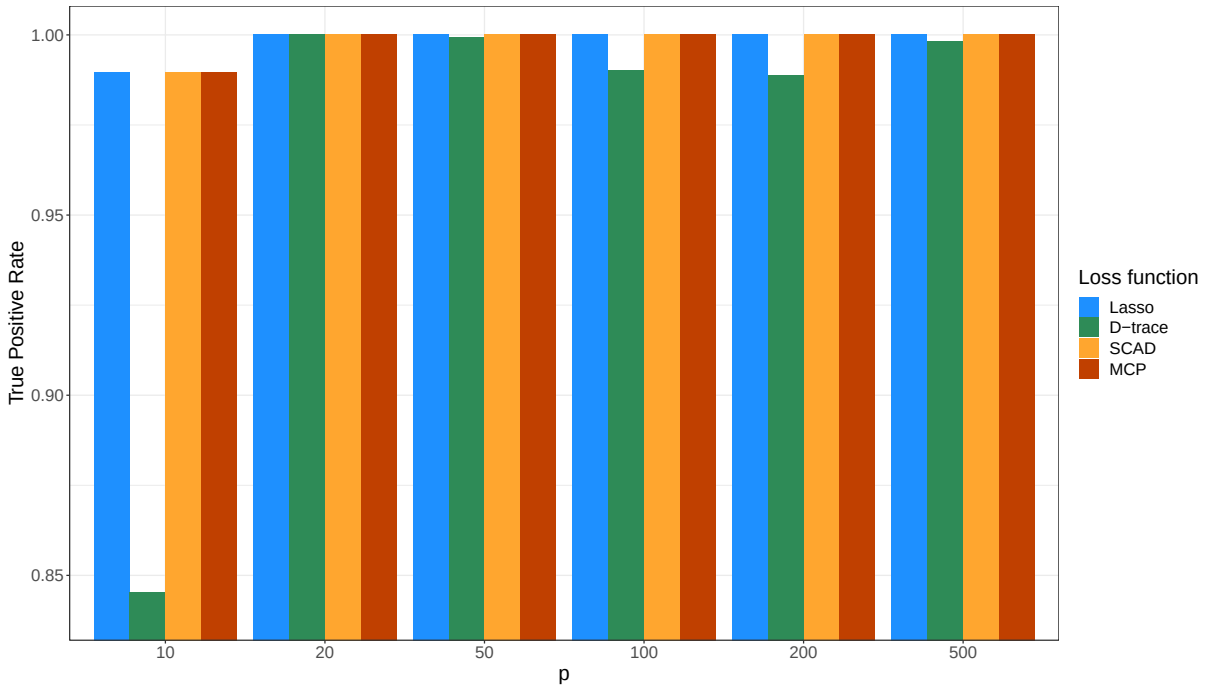

Figure 24: True positive rate against dimensionality - Asymptotically sparse case

Thus, having completed the analyses for standard normal data, the analyses are repeated using data that has undergone the truncation nonparanormal transformation. The results obtained were as follows:

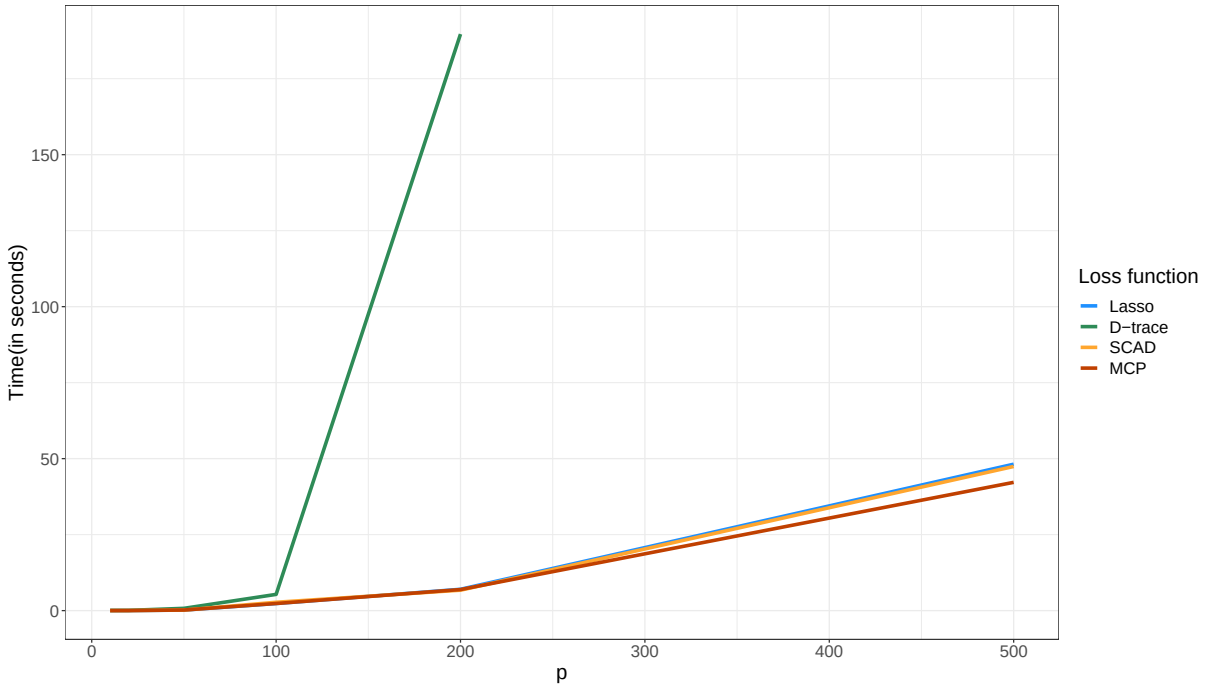

Figure 25: Computation time against dimensionality - Sparse case with nonparanormal transformation

Table 5: Computation time(in seconds) - Sparse case with nonparanormal transformation

Dimensions	LASSO	D-trace	SCAD	MCP
10	0.01	0.07	0.01	0.02
20	0.02	0.08	0.03	0.03
50	0.18	0.75	0.18	0.20
100	2.34	5.34	2.75	2.34
200	7.02	189.67	6.72	6.93
500	48.17	8182.14	47.42	42.20

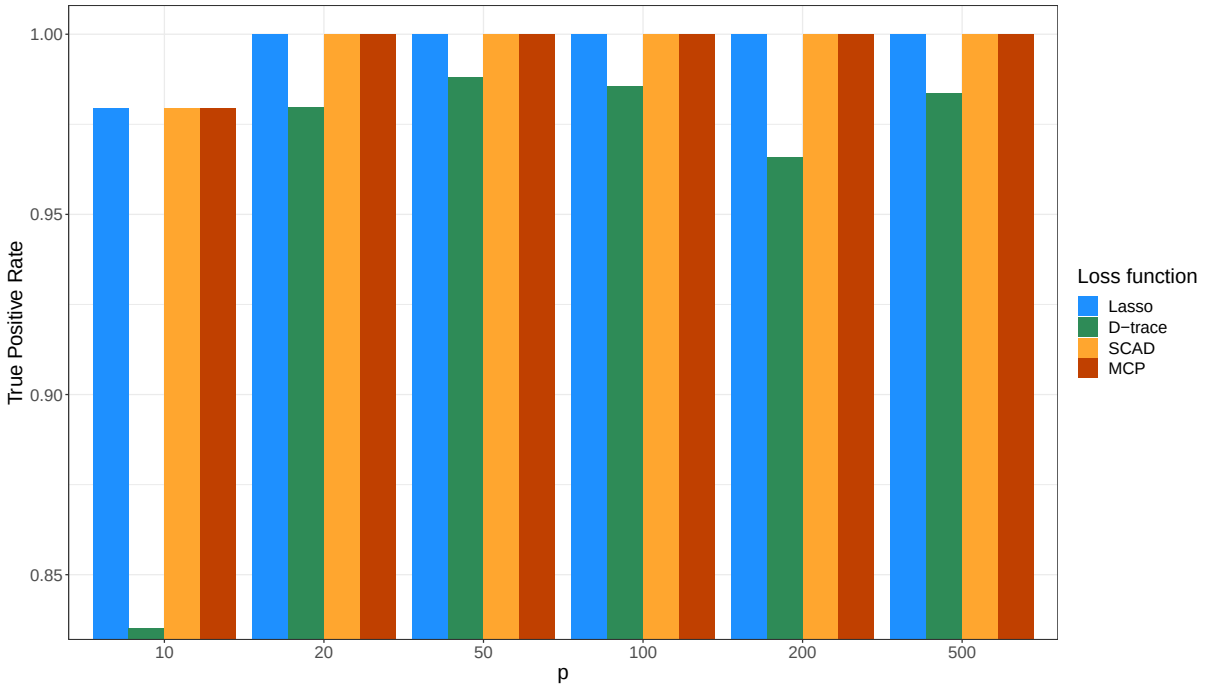

Figure 26: True positive rate against dimensionality - Sparse case with nonparanormal transformation

Table 6: True positive rate - Sparse case with nonparanormal transformation

Dimensions	LASSO	D-trace	SCAD	MCP
10	0.9793814	0.8350515	0.9793814	0.9793814
20	1.0000000	0.9798489	1.0000000	1.0000000
50	1.0000000	0.9879856	1.0000000	1.0000000
100	1.0000000	0.9855957	1.0000000	1.0000000
200	1.0000000	0.9658974	1.0000000	1.0000000
500	1.0000000	0.9837518	1.0000000	1.0000000

From Figures 25 and 26, along with the results shown in Tables 5 and 6 the effect of the nonparanormal transformation is evident. That is, applying such a transformation had an extremely marginal impact with the overall trends remaining consistent with what was observed without the transformation applied. The d-trace loss functions provides, by far the worst computational performance and once again the worst accuracy of estimates. One key difference observable within these results, is that unlike previously where LASSO and SCAD provided superior performance, the MCP loss function is now the most efficient. It is of interest to examine whether this remains the case when considering asymptotically sparse covariance matrices along with the nonparanormal transformation. These results were obtained as follows:

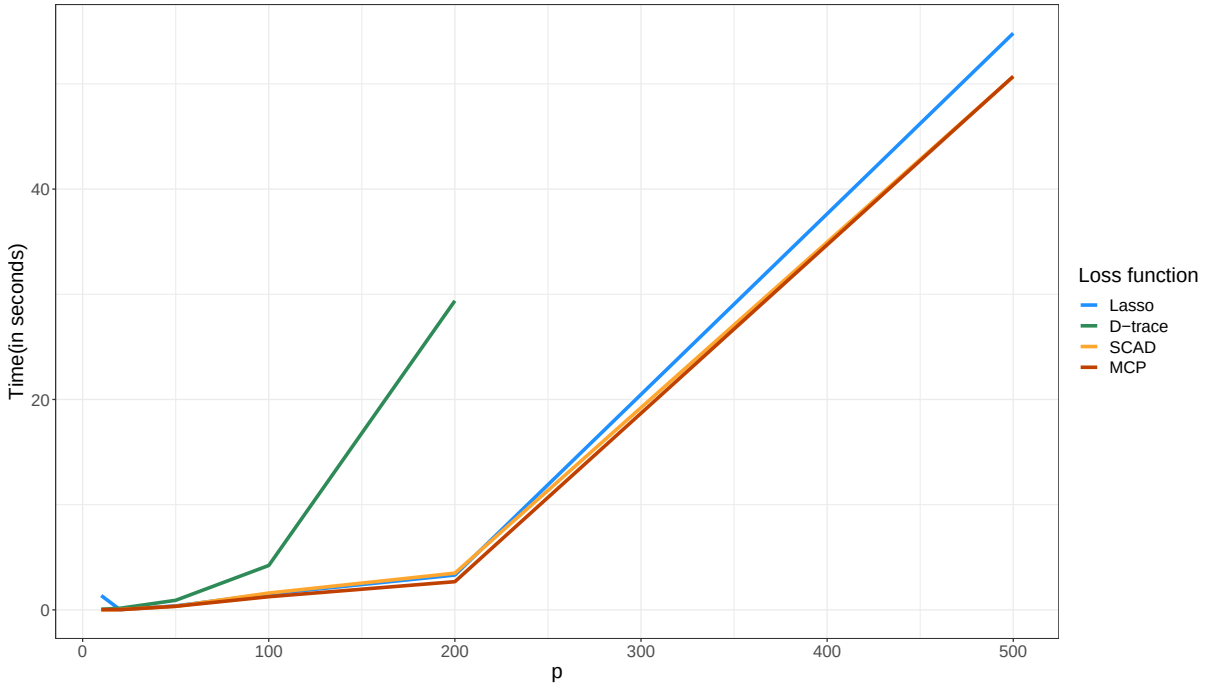

Figure 27: Computation time against dimensionality - Asymptotically sparse case with nonparanormal transformation

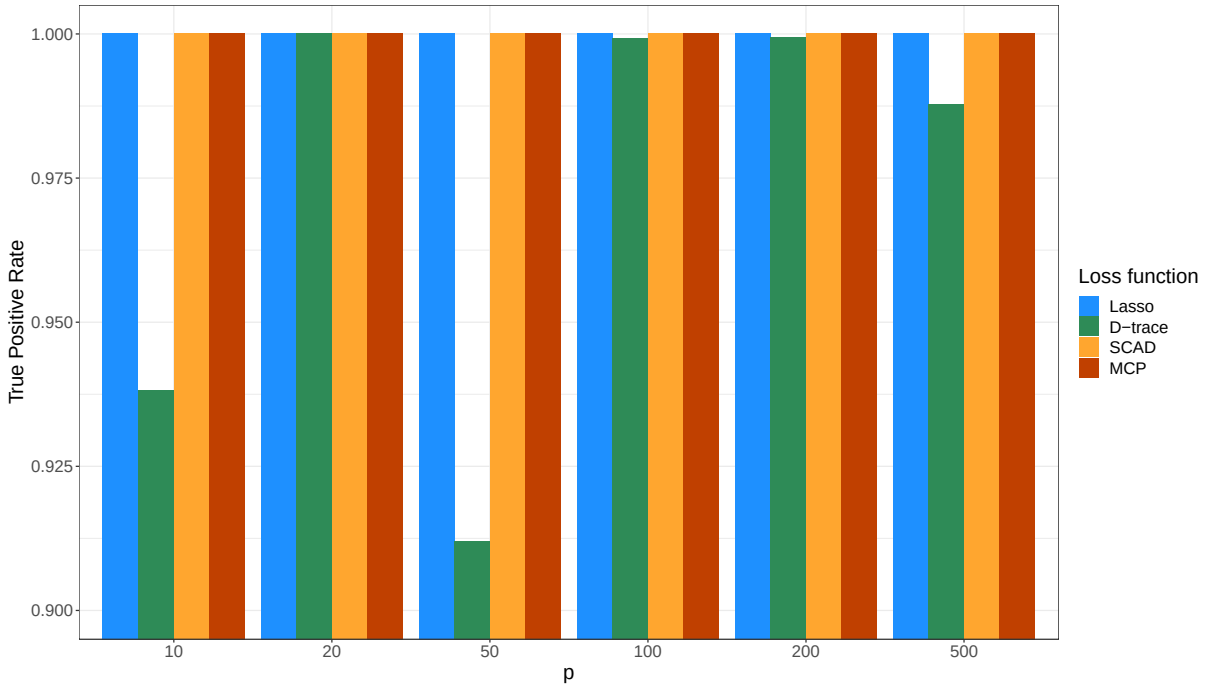

Figure 28: True positive rate against dimensionality - Asymptotically sparse case with nonparanormal transformation

The final analysis shown above indicates a similar story to that of the previous analyses. The d-trace loss function is by far the slowest of the loss functions considered, with the worst overall accuracy. For the remainder of the loss functions, their performance remains relatively equivalent with the SCAD and

MCP providing slightly better performance as has been the case for both experiments making use of the nonparanormal data.

Table 7: Computation time(in seconds) - Asymptotically sparse case with nonparanormal transformation

Dimensions	LASSO	D-trace	SCAD	MCP
10	1.36	0.06	0.01	0.02
20	0.03	0.15	0.03	0.02
50	0.38	0.91	0.34	0.34
100	1.51	4.22	1.60	1.25
200	3.31	29.38	3.49	2.68
500	54.81	9252.86	50.69	50.71

Table 8: True positive rate - Asymptotically sparse case with nonparanormal transformation

Dimensions	LASSO	D-trace	SCAD	MCP
10	1	0.9381443	1	1
20	1	1.0000000	1	1
50	1	0.9118943	1	1
100	1	0.9991998	1	1
200	1	0.9994500	1	1
500	1	0.9877839	1	1

From the above analyses, it can be seen that the d-trace function should almost always be avoided, and a possible explanation for it's particular poor showing across the four different settings considered is that in none of the investigations was the the irrepresentability condition verified. From the theory within Section 3.2 violations to this condition are known to have a significant impact on the optimization. It is important to note that despite possible violations to this assumption, the accuracy of the estimate obtained from the d-trace loss function, along with the accuracies of the other loss functions were all extremely excellent and as such one can be confident of obtaining an accurate estimate in similar circumstances to the ones described above using dineR. Lastly, it is also of interest to take note that throughout the four different cases presented above, no one loss function dominated the alternatives. This is in line with expectations, as each loss function is expected to thrive to differing degrees under different experimental conditions [52].

Having completed the above analyses in which the different loss functions were directly compared to one another under a variety of dimensionality structures and covariance patterns, it then became of interest to investigate the impact that these differences had on any one loss functions as a whole. That is, the four different scenarios, the sparse case with normal data, the asymptotically sparse with normal data, the sparse case with the nonparanormal data and the asymptotically sparse case with nonparanormal data were all reconsidered. This time however, each individual loss function was only compared with itself to provide insight into the effect of the nonparanormal transformation and differing covariance structures had on the performance of the ADMM optimization. The results obtained, were as follows:

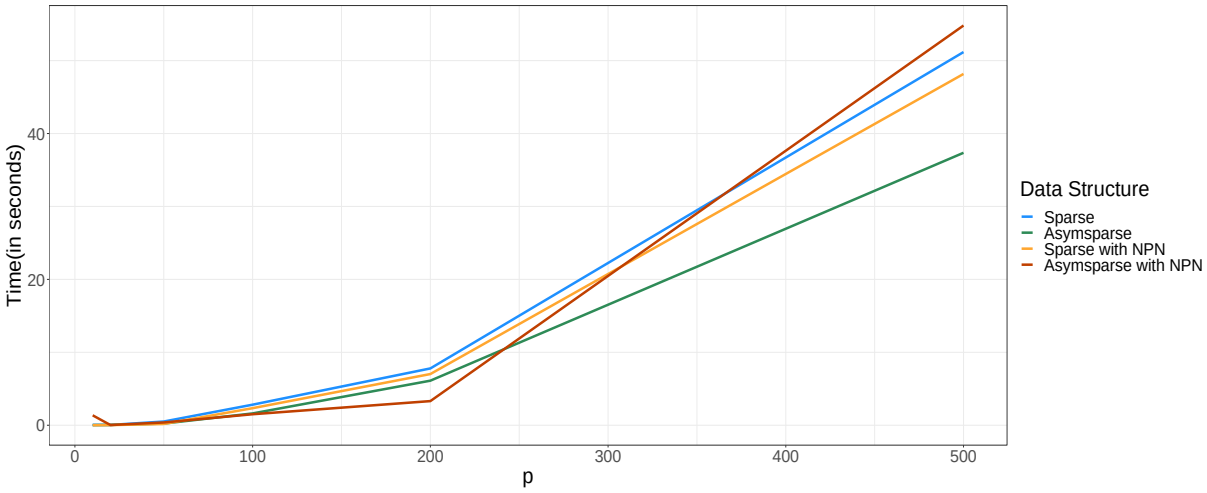

Figure 29: Computation time against dimensionality - LASSO loss function

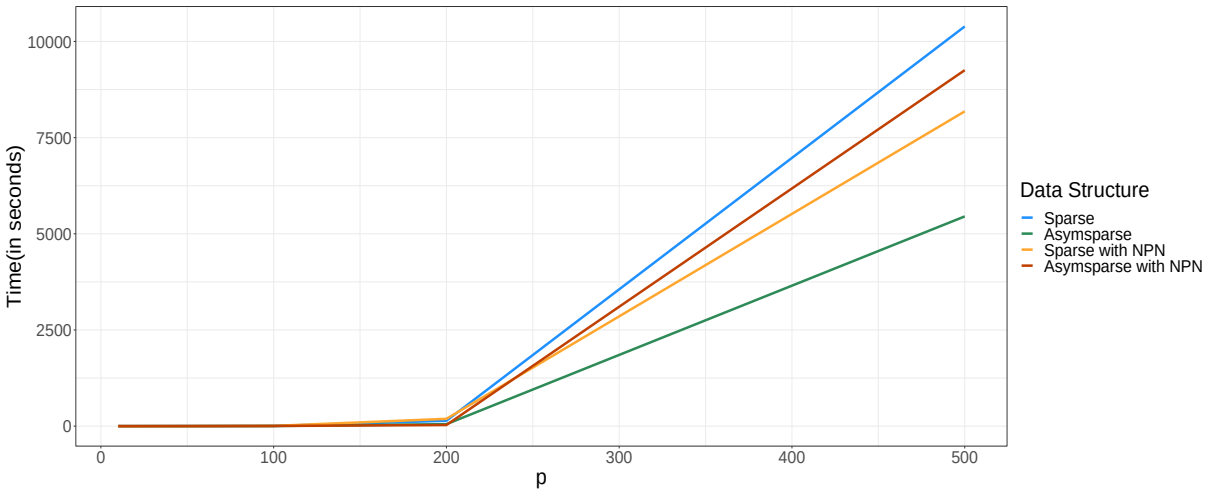

Figure 30: Computation time against dimensionality - D-trace loss function

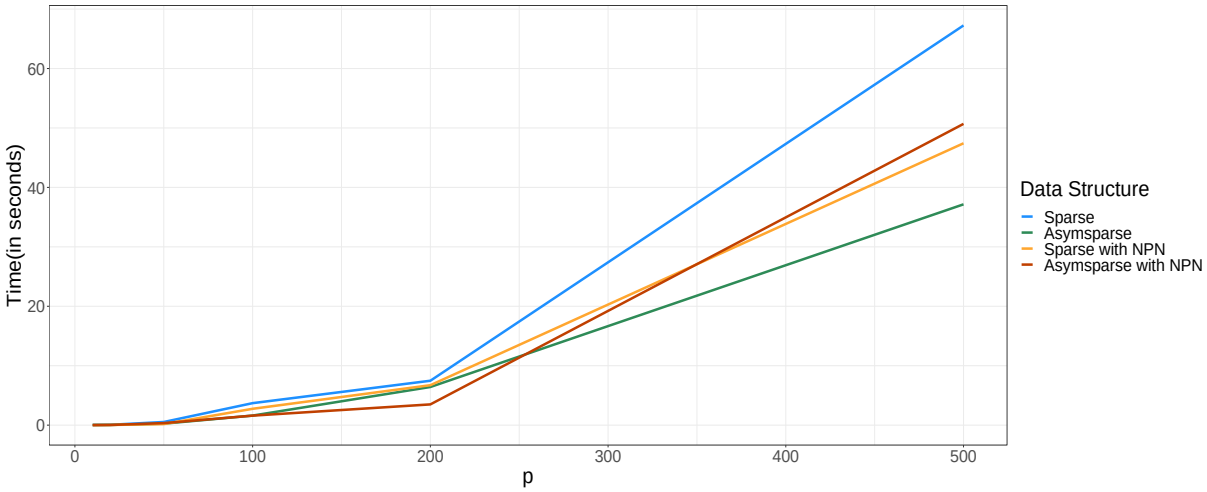

Figure 31: Computation time against dimensionality - SCAD loss function

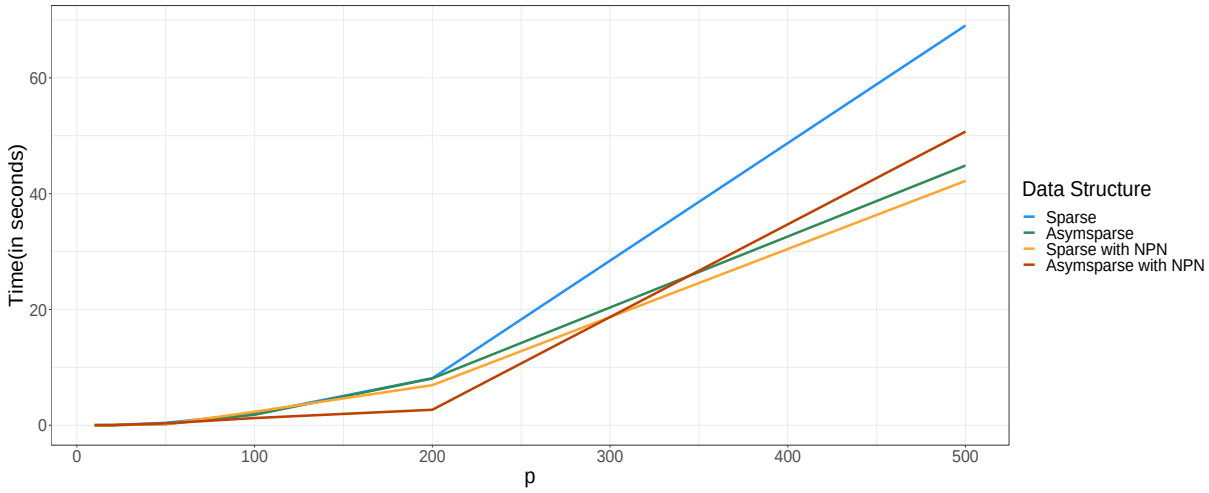

Figure 32: Computation time against dimensionality - MCP loss function

By examining the above figures, it is evident that the optimization was most efficient in the asymptotically sparse case for all of the loss functions, except the MCP loss in which the sparse case with nonparanormal data slightly edged it. The most interesting finding from the above figures is the fact that when the precision matrices are sparse, it is more efficient to apply the nonparanormal transformation even if the data in questions is normally distributed. The simulation studies thus far have provided available insight into the numerical behaviour of the different loss functions, and shown under which circumstances each loss function should be slightly favoured over it's alternatives. Using this information, two real-world datasets are visited for which differential networks will be estimated with the use of dineR.

4.2.2 SARS-CoV-2 Analysis

Given the recent SARS-CoV-2, COVID-19, global pandemic there has been significant interest on studies surrounding the long-term effects of the illness and it's variants, as well as the safety and efficacy of the vaccines in place. However, there exists a shortage of openly available patient-level data required for differential network estimation to be of use. As such, the dissertation aims to examine the impact of SARS-CoV-2 to various non-patient level factors, such as unemployment, death rates, and income. To perform such an analysis, the data of Xu et al. [88] was considered. The data in question, consists of a variety of information for each state in the United States of America. This includes, the median temperature of the state, whether the state is held by the Republican party or the Democratic party, the unemployment rate, median age, population density, median income and several others. The aim is to then find several unique and valuable insights present within the data that are not easily discoverable without the use of differential networks. There were then three different analyses explored. The first investigation performed, was to determine whether Republican states were adversely effected more so or less so during the pandemic when compared to Democratic states. The differential network obtained

when comparing these two different political regimes is then the following:

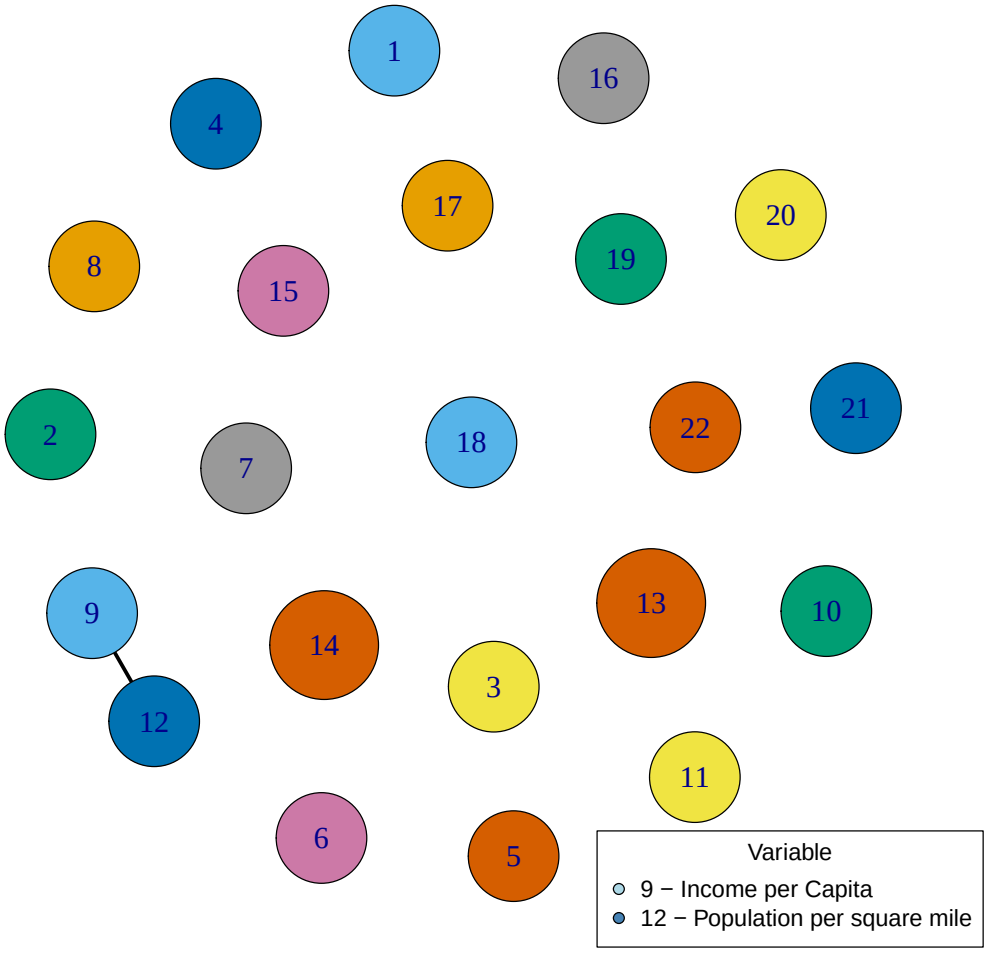

Figure 33: Differential network investigating the impact of the different governing parties

From Figure 33 it thus appears that the differences between political regimes across the various states through the United States of America did not effect any of the variables related to the SARS-CoV-2 pandemic such as the death or vaccination rates or even the unemployment figures. The one relationship that did however differ between these political parties' states is the income per capita and the population density of their respective states. One possible explanation for this could be that two of the most densely populated states, California and New York are both ran by the Democratic party, while states such as Texas, Alabama and Missouri which all have far lower population densities are Republican states. Having investigated the impact of SARS-CoV-2 across the different states, taking into account the ruling political

party, it then became of interest to investigate as to what, if any effect temperature had on the variables in question. The differential network comparing the 25 hottest states, to the 25 coldest states was then as follows:

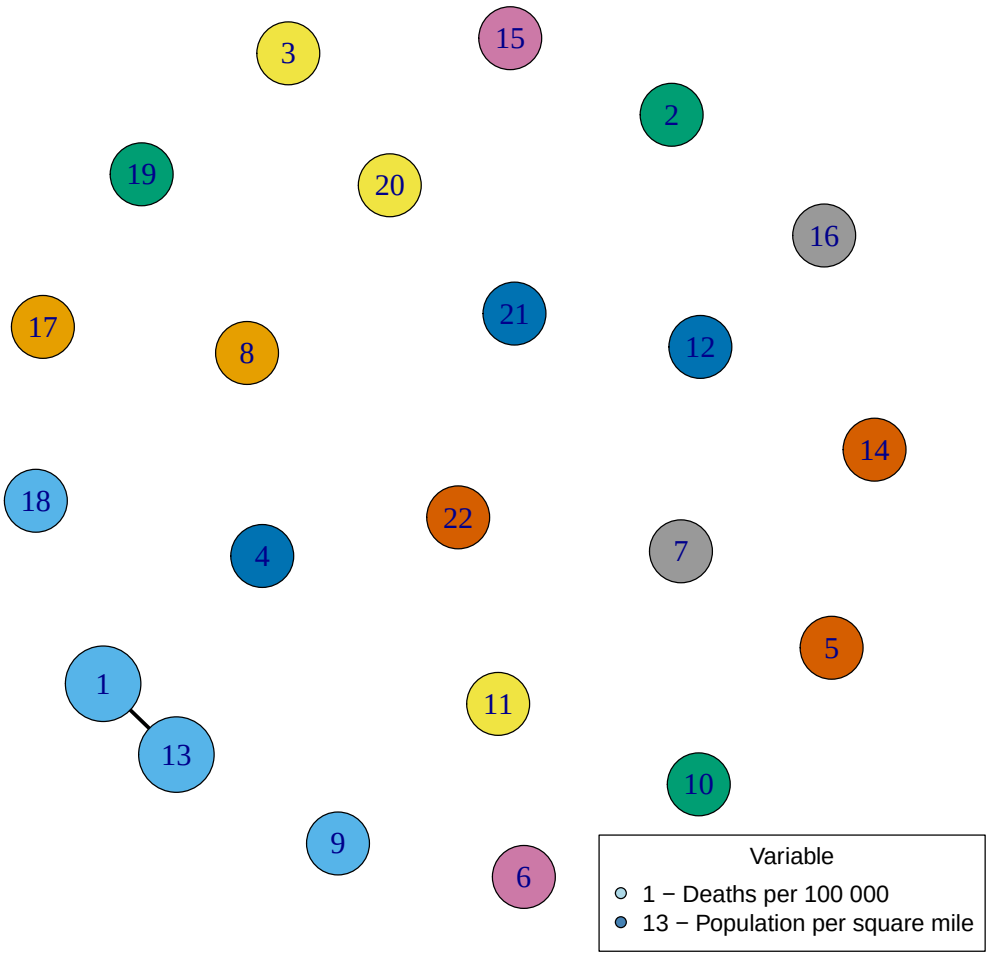

Figure 34: Differential network investigating the impact of median temperature

From Figure 34 there is again, one singular difference between the changes between variables observed between the hotter states when compared to the colder states. That is, there was a change between the relationship of the number of deaths per 100 000 individuals and the population density. This is by no means a ground breaking insight, as due to the air-born nature of SARS-CoV-2 as well as the contagiousness of the illness, it is expected that as the population density increases, so too does the deaths. However, what is of great interest here is that as the population density changed, the number

of deaths changed disproportionately between hotter and colder states, indicating that temperature may indeed have had an effect on the spread of SARS-CoV-2. Whether this difference is as a result of the virus behaving differently in different climates is unsure, as the difference could possibly be related to differences in population behaviour, such as poorer mask usage in warmer weather, or people isolating at home more in colder circumstances.

The final experiment considered was then whether the median impact played a role on the effect of the pandemic. The differential network obtained, comparing the 25 wealthiest states, to the 25 poorest states is then as follows:

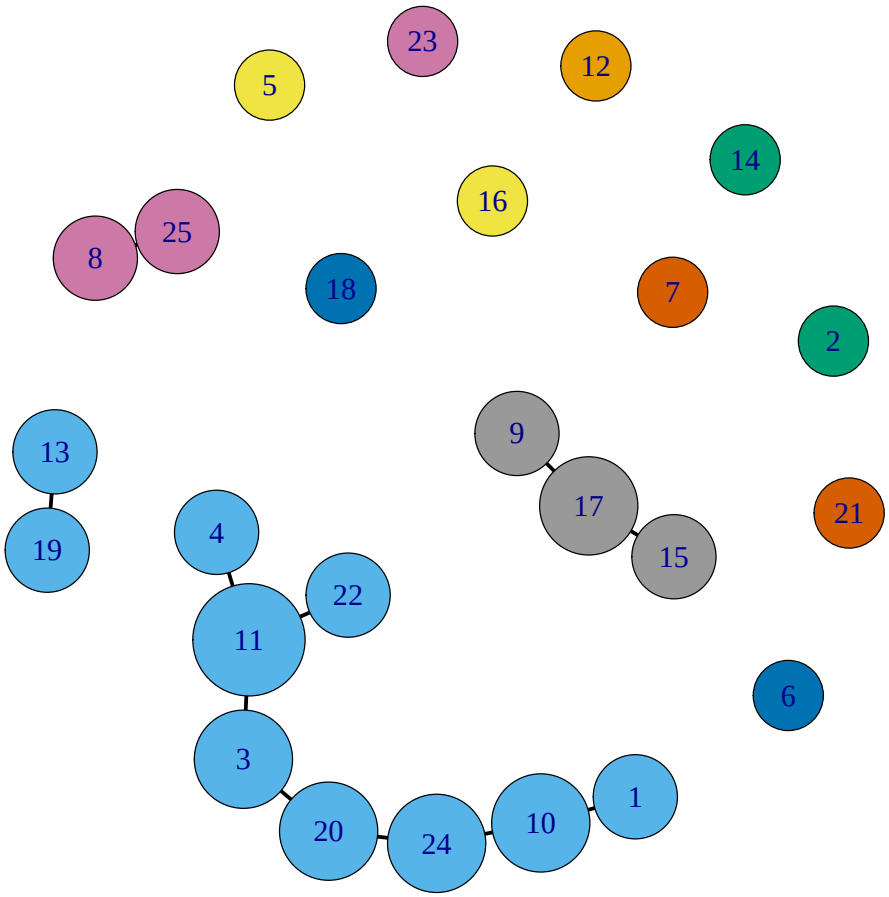

Figure 35: Differential network investigating the impact of median income

From Figure 35 it is clear that of the three different scenarios considered, income per capita had by

far the most differences across the relationships observed. This indicates that there was a stark contrast between the impact of SARS-CoV-2 on poor states when compared to the richer states. This type of result, is expected as richer states will have access to better health care systems. Individuals in high income positions, can also afford to quarantine in the event of suspected exposure to the virus, something less wealthy individuals may not be afforded. Similarly, low income states will have large numbers of their population staying in smaller households, in which there may be a large number of occupants to reduce costs. Lastly, wealthier individuals are also not as reliant on public transport to the same degree as lower income citizens where the risk of exposure is high. Thus, while many of the above factors are well-known and extensively documented, differential networks such as the above can be used to pinpoint the area in most urgent need of being addressed to lessen and possibly prevent disproportionate impacts of global crises such as the current ongoing pandemic. For example in the above graphic, variable 3 - death rate per 100 000 inhabitants and variable 20 - the obesity rate undergo a change in association between the richer and poor states. As such, the United States government may be able to lower the death rate in poorer states, by addressing the obesity rates in these states. This is an extremely powerful insight, as uncertainty remains high, combined with vaccine hesitancy and the threat of SARS-CoV-2 becoming endemic alternative measures are desperately needed to curb this virus, with no end in sight in the near future.

4.2.3 Ultra High-Dimensional Analysis

Having illustrated the power of differential networks in Section 4.2.2, their ability in a high-dimensional setting is now considered. To showcase this, the data in the *spls* R package [18] complied and collected by Singh et al. [67] is explored. The data consists of only 102 observations but more than 6000 features and as such is considered truly ultra high-dimensional according to Fan and Lv's definition which states data is considered ultra high-dimensional when the number of features is one or more magnitudes larger than the number of observations. The number of features under consideration here, is then 60 magnitudes larger than the number of observations. The data in question consists of prostate tumor gene expression data, with two different cohorts present. That is, there are 52 individuals' measurements present with the dataset, while there are also 50 gene expression measurements for individuals confirmed to have a prostate tumor.

A differential network for the data will then be estimated, considering 10 different values for λ, with the optimal value of λ being selected through the use of the EBIC. The data has also been normalized and as such, the nonparanormal transformation will not be applied. The loss function of choice, will be the LASSO loss function, as it is assumed that many of the interactions between the two states will be similar making it appropriate to assume $\hat{\Delta}$ will be sparse and the LASSO loss was most efficient for such

a setting in the simulation studies. The differential network was thus obtained, with the a set of 6033 vertices as visualized from the below output:

```
+ 6033/6033 vertices, from e3bf322:
   [1]    1    2    3    4    5    6    7    8    9
  [10]   10   11   12   13   14   15   16   17   18
  [19]   19   20   21   22   23   24   25   26   27
  [28]   28   29   30   31   32   33   34   35   36
  [37]   37   38   39   40   41   42   43   44   45
  [46]   46   47   48   49   50   51   52   53   54
  [55]   55   56   57   58   59   60   61   62   63
  [64]   64   65   66   67   68   69   70   71   72
  [73]   73   74   75   76   77   78   79   80   81
  [82]   82   83   84   85   86   87   88   89   90
+ ... omitted several vertices
```

Figure 36: Vertices for prostate tumor gene expression data

If the number of edges are then considered, the results are as follows:

```
+ 549/549 edges from 4a8260d:
 [1]  37--3183  37--4177  54--3183  54--4336
 [5]  54--4404  59-- 127  59-- 527  59--1245
 [9]  59--3183  59--5214 105--3183 105--3287
[13] 105--3346 105--4177 105--4404 105--4494
[17] 118--3183 118--4177 126--3183 126--3287
[21] 126--3346 126--4177 126--4494 127-- 251
[25] 127-- 309 127-- 428 127-- 527 127-- 596
[29] 127--2852 127--3183 127--3287 127--3346
[33] 127--3621 127--4177 127--4404 127--4494
[37] 127--4872 127--5032 127--5445 141-- 251
+ ... omitted several edges
```

Figure 37: Non-zero edges for prostate tumor gene expression data

There are thus, only 549 non-zero edges in the entire network of $6033 \times 6033 = 36397089$ possible interactions. Since, only the variables which have some non-zero interaction between them are of interest, all of the vertices without an edge can be dropped from the analysis. This produces the following output:

```
+ 125/125 vertices, from e4d5612:
  [1]   1   2   3   4   5   6   7   8   9  10  11
 [12]  12  13  14  15  16  17  18  19  20  21  22
 [23]  23  24  25  26  27  28  29  30  31  32  33
 [34]  34  35  36  37  38  39  40  41  42  43  44
 [45]  45  46  47  48  49  50  51  52  53  54  55
 [56]  56  57  58  59  60  61  62  63  64  65  66
 [67]  67  68  69  70  71  72  73  74  75  76  77
 [78]  78  79  80  81  82  83  84  85  86  87  88
 [89]  89  90  91  92  93  94  95  96  97  98  99
[100] 100 101 102 103 104 105 106 107 108 109 110
+ ... omitted several vertices
```

Figure 38: Vertices with non-zero edges for prostate tumor gene expression data

With the corresponding differential network obtained as:

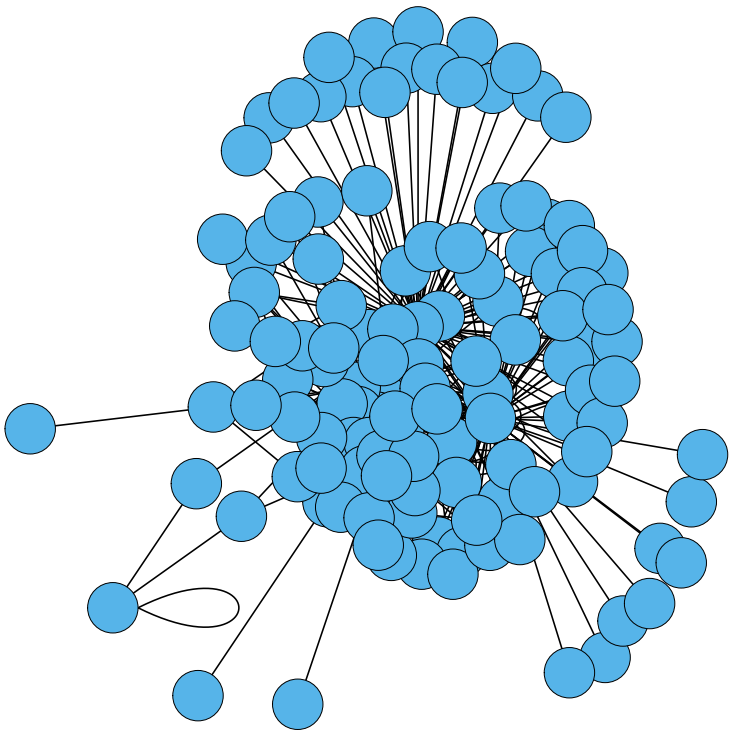

Figure 39: Differential network with only non-zero edges for prostate tumor gene expression data

Hence, although differential networks are not typically considered as a means to perform dimensionality reduction, for this particular problem the network was able to reduce the number of features that must be studied from 6033 to a far more manageable 125 features. This is a reduction of approximately 98%, and as such makes any and all statistical analysis far simpler as now differential network analysis can be applied to these selected features to quantify the extent to which the relationships differ and if there are any underlying clusters present to assist researchers in developing efficient cancer treatments, but also investigating cancer progression.

5 Conclusion

Upon beginning this research, there did not exist a readily available manner to estimate differential networks, however with the development of dineR the foundation has been laid to place differential networks at the forefront of big data statistical analysis going forward. This research then acts as a manual on all things differential networks. From the technicalities of their estimation in R, to their definition, understanding and the motivation for their use. Details regarding the history and development of differential network estimation were also discussed, while the optimization schemes and loss functions most commonly used were also provided. Numerical analyses were also considered to evaluate dineR and it's respective efficiency as well as the efficiency and accuracy of the proposed methods of estimation. Violations to the crucial assumptions of the theoretical underpinning graphical theory was also addressed. Lastly, real-world data examples were considered to showcase the true power of differential networks, in both low-dimensional setting and the far more complicated and traditionally troublesome ultra high-dimensional scenario. Despite each of the above, and having addressed each of the objectives of this research in full, there exist several interesting avenues for future work which may prove groundbreaking.

5.1 Future Works

Such avenues include:

- Investigation and formalization of graphical models in which the underling distribution is no longer assumed to be Gaussian. Several scenarios in which these models may be of great use, include real-world applications in which the data is heavy-tailed, exponential, or in which the data is discrete such as in a binary or Poisson setting [65].

- More and more techniques are being developed to aid in the estimation of differential networks, while very few specialized techniques exist to perform informative and efficient statistical analysis on differential networks [37]. With most existing methods being far too general, or extremely specific

to a particular problem and as such requiring significant background knowledge. It thus may be of significant interest to develop and fine-tune approaches to assist with this task.

- Most existing estimation schemes for differential networks make use of the same optimization procedure. That is, they all make sure of alternating direction method of multipliers, ADMM. It should thus be examined as to whether appropriate alternative optimization procedures exist and can be applied to benefit estimation.

- dineR currently does not allow for parallelization of code to speed up the performance, and as such given the computation cost of solving large scale differential networks should be included within the package to further improve efficiency.

- Lastly, and arguably the most exciting yet difficult avenue for future work is the consideration of directed graphs. The inclusion of directions within the graphical structure considered carries both a complex theoretical component, but also a far greater computational cost than in the undirected case. However, the resulting networks may be of even greater interest, due to additional information regarding the changes in relationships encoded within the network.

References

[1] Hirotogu Akaike. Information Theory and an Extension of the Maximum Likelihood Principle. In *Selected papers of hirotugu akaike*, pages 199–213. Springer, 1998.

[2] Douglas Bates and Martin Maechler. *Matrix: Sparse and Dense Matrix Classes and Methods*, 2019. URL `https://CRAN.R-project.org/package=Matrix`.

[3] Amir Beck and Marc Teboulle. A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[4] Tatiana Benaglia, Didier Chauveau, David R. Hunter, and Derek Young. mixtools: An R Package for Analyzing Finite Mixture Models. *Journal of Statistical Software*, 32(6):1–29, 2009. URL `http://www.jstatsoft.org/v32/i06/`.

[5] Peter J. Bickel and Elizaveta Levina. Regularized Estimation of Large Covariance Matrices. *The Annals of Statistics*, 36(1):199–227, 2008.

[6] Roger Bivand and David W. S. Wong. Comparing implementations of global and local indicators of spatial association. *TEST*, 27(3):716–748, 2018. URL `https://doi.org/10.1007/s11749-018-0599-x`.

[7] Stephen P. Borgatti and Daniel S. Halgin. On Network Theory. *Organization science*, 22(5):1168–1181, 2011.

[8] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge university press, 2004.

[9] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Eckstein Jonathan. Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. *Foundations and Trends in Machine Learning*, 3, 2011.

[10] Jennifer C. Britton, Stephan F. Taylor, Keith D Sudheimer, and Israel Liberzon. Facial Expressions and Complex IAPS Pictures: Common and Differential Networks. *Neuroimage*, 31(2):906–919, 2006.

[11] Lawrence Cabusora, Electra Sutton, Andy Fulmer, and Christian Forst. Differential Network Expression During Drug and Stress Response. *Bioinformatics*, 21(12):2898–2905, 2005.

[12] Tony Cai and Weidong Liu. Adaptive Thresholding for Sparse Covariance Matrix Estimation. *Journal of the American Statistical Association*, 106(494):672–684, 2011.

[13] Tony Cai, Weidong Liu, and Xi Luo. A Constrained ℓ_1 Minimization Approach to Sparse Precision Matrix Estimation. *Journal of the American Statistical Association*, 106(494):594–607, 2011.

[14] Jiahua Chen and Zehua Chen. Extended Bayesian Information Criteria for Model Selection with Large Model Spaces. *Biometrika*, 95(3):759–771, 2008.

[15] Jiahua Chen and Zehua Chen. Extended BIC for small-n-large-p Sparse GLM. *Statistica Sinica*, pages 555–574, 2012.

[16] Pei Chen, Yongjun Li, Xiaoping Liu, Rui Liu, and Luonan Chen. Detecting the Tipping Points in a Three-State Model of Complex Diseases by Temporal Differential Networks. *Journal of Translational Medicine*, 15(1):217, 2017.

[17] Julien Chiquet, Yves Grandvalet, and Christophe Ambroise. Inferring Multiple Graphical Structures. *Statistics and Computing*, 21(4):537–553, 2011.

[18] Dongjun Chung, Hyonho Chun, and Sunduz Keles. *spls: Sparse Partial Least Squares (SPLS) Regression and Classification*, 2019. URL `https://CRAN.R-project.org/package=spls`. R package version 2.2-3.

[19] Caleb A Class, Min Jin Ha, Veerabhadran Baladandayuthapani, and Kim-Anh Do. iDINGO-Integrative Differential Network Analysis in Genomics with Shiny Application. *Bioinformatics*, 34 (7):1243–1245, 2018.

[20] Gábor Csárdi and Rich FitzJohn. *progress: Terminal Progress Bars*, 2019. URL `https://CRAN.R-project.org/package=progress`.

[21] Gábor Csárdi and Tamas Nepusz Nepusz. The igraph Software Package for Complex Network Research. *InterJournal*, Complex Systems:1695, 2006. URL `http://igraph.org`.

[22] Gábor Csárdi and Maëlle Salmon. *rhub: Connect to 'R-hub'*, 2019. URL `https://CRAN.R-project.org/package=rhub`. R package version 1.1.1.

[23] Patrick Danaher, Pei Wang, and Daniela M Witten. The Joint Graphical LASSO for Inverse Covariance Estimation across Multiple Classes. *Journal of the Royal Statistical Society. Series B, Statistical Methodology*, 76(2):373, 2014.

[24] Steven Durlauf and Lawrence Blume. *The new Palgrave dictionary of economics*. Springer, 2016.

[25] Jianqing Fan and Jinchi Lv. Sure Independence Screening for Ultrahigh Dimensional Feature Space. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(5):849–911, 2008.

[26] Jianqing Fan, Yang Feng, and Yichao Wu. Network Exploration via the Adaptive LASSO and SCAD Penalties. *The Annals of Applied Atatistics*, 3(2):521, 2009.

[27] Jianqing Fan, Yuan Liao, and Han Liu. An Overview on the Estimation of Large Covariance and Precision Matrices. *The Econometrics Journal*, 19:C1–C32, 2016.

[28] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Sparse Inverse Covariance Estimation with the Graphical Lasso. *Biostatistics*, 9(3):432–441, 2008.

[29] Nick Galov. 77+ Big Data Stats for the Big Future Ahead | Updated 2021. `https://hostingtribunal.com/blog/big-data-stats/`, 2021. [Online; accessed 19-January-2021].

[30] Simon Garnier, Noam Ross, Bob Rudis, Macro Sciaini, Antônio P. Camargo, and Cédric Scherer. *viridis - Colorblind-Friendly Color Maps for R*, 2021. URL `https://sjmgarnier.github.io/viridis/`.

[31] Ryan Gill, Somnath Datta, and Susmita Datta. A Statistical Framework for Differential Network Analysis from Microarray Data. *BMC Bioinformatics*, 11(1):95, 2010.

[32] Ryan Gill, Somnath Datta, and Susmita Datta. Differential Network Analysis in Human Cancer Research. *Current pharmaceutical design*, 20(1):4–10, 2014.

[33] Ryan Gill, Somnath Datta, and Susmita Datta. dna: An R Package for Differential Network Analysis. *Bioinformation*, 10(4):233, 2014.

[34] Jian Guo, Elizaveta Levina, George Michailidis, and Ji Zhu. Joint Estimation of Multiple Graphical Models. *Biometrika*, 98(1):1–15, 2011.

[35] Kurt Hornik. The Comprehensive R Archive Network. *Wiley interdisciplinary reviews: Computational statistics*, 4(4):394–398, 2012.

[36] Wolfgang Huber, Vincent J. Carey, Robert Gentleman, Simon Anders, Marc Carlson, Benilton S. Carvalho, Hector Corrada Bravo, Sean Davis, Laurent Gatto, Thomas Girke, et al. Orchestrating High-Throughput Genomic Analysis with Bioconductor. *Nature methods*, 12(2):115, 2015.

[37] Trey Ideker and Nevan J. Krogan. Differential Network Biology. *Molecular Systems Biology*, 8(1): 565, 2012.

[38] Binyan Jiang, Xiangyu Wang, and Chenlei Leng. A Direct Approach for Sparse Quadratic Discriminant Analysis. *The Journal of Machine Learning Research*, 19(1):1098–1134, 2018.

[39] Michael Jordan. Graphical Models. *Statistical Science*, 19(1):140–155, 2004.

[40] Michael I Jordan and Chris Bishop. An Introduction to Graphical Models, 2004.

[41] Roger Koenker and Ivan Mizera. Convex Optimization in R. *Journal of Statistical Software*, 60(5): 1–23, 2014.

[42] Han Liu, John Lafferty, and Larry Wasserman. The Nonparanormal: Semiparametric Estimation of High Dimensional Undirected Graphs. *Journal of Machine Learning Research*, 10(10), 2009.

[43] Han Liu, Fang Han, Ming Yuan, John Lafferty, and Larry Wasserman. High-Dimensional Semiparametric Gaussian Copula Graphical Models. *The Annals of Statistics*, 40(4):2293–2326, 2012.

[44] Xiaoke Ma, Long Gao, Georgios Karamanlidis, Peng Gao, Chi Fung Lee, Lorena Garcia-Menendez, Rong. Tian, and Kai Tan. Revealing Pathway Dynamics in Heart Diseases by Analyzing Multiple Differential Networks. *PLoS computational biology*, 11(6), 2015.

[45] Ricardo Marques. *dineR: Differential Network Estimation in R*, 2021. URL `https://CRAN.R-project.org/package=dineR`. R package version 1.0.0.

[46] Philippe Massicotte and Dirk Eddelbuettel. *gtrendsR: Perform and Display Google Trends Queries*, 2021. URL `https://CRAN.R-project.org/package=gtrendsR`.

[47] Rahul Mazumder and Trevor Hastie. The Graphical Lasso: New Insights and Alternatives. *Electronic Journal of Statistics*, 6:2125, 2012.

[48] Nicolai Meinshausen and Peter Bühlmann. High-Dimensional Graphs and Variable Selection with the Lasso. *The Annals of Statistics*, 34(3):1436–1462, 2006.

[49] Mahesh More. How Much Data does Google Handle? `https://www.heshmore.com/how-much-data-does-google-handle/`, 2017. [Data retrieved on the 7-June-2020].

[50] Martin Morgan. *BiocManager: Access the Bioconductor Project Package Repository*, 2019. URL `https://CRAN.R-project.org/package=BiocManager`. R package version 1.30.10.

[51] Kevin Murphy. An Introduction to Graphical Models. *Rap. Tech*, 96:1–19, 2001.

[52] Joseph O. Ogutu and Hans-Peter Piepho. Regularized Group Regression Methods for Genomic Prediction: Bridge, MCP, SCAD, Group Bridge,Group LASSO, Sparse Group LASSO, Group MCP and Group SCAD. In *BMC proceedings*, volume 8, pages 1–9. Springer, 2014.

[53] Le Ou-Yang, Hong Yan, and Xiao-Fei Zhang. Identifying Differential Networks Based on Multi-Platform Gene Expression Data. *Molecular BioSystems*, 13(1):183–192, 2017.

[54] Le Ou-Yang, Xiao-Fei Zhang, Min Wu, and Xiao-Li Li. Node-Based Learning of Differential Networks from Multi-Platform Gene Expression Data. *Methods*, 129:41–49, 2017.

[55] Szilard Pafka. Big RAM is eat big data - Size of datasets used for analytics. `https://www.kdnuggets.com/2015/11/big-ram-big-data-size-datasets.html`, 2015. [Online; accessed 19-January-2021].

[56] Edzer Pebesma. Simple Features for R: Standardized Support for Spatial Vector Data. *The R Journal*, 10(1):439–446, 2018. doi: 10.32614/RJ-2018-009. URL `https://doi.org/10.32614/RJ-2018-009`.

[57] Thomas Lin Pedersen. *tidygraph: A Tidy API for Graph Manipulation*, 2020. URL `https://CRAN.R-project.org/package=tidygraph`.

[58] Thomas Lin Pedersen. *ggraph: An Implementation of Grammar of Graphics for Graphs and Networks*, 2021. URL `https://CRAN.R-project.org/package=ggraph`.

[59] Pablo Pedregal. *Introduction to Optimization*, volume 46. Springer Science & Business Media, 2006.

[60] Benedikt M. Pötscher and Hannes Leeb. On the Distribution of Penalized Maximum Likelihood Estimators: The LASSO, SCAD, and Thresholding. *Journal of Multivariate Analysis*, 100(9):2065–2082, 2009.

[61] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2013. URL `http://www.R-project.org/`.

[62] Adam J. Rothman, Elizaveta Levina, and Ji Zhu. Generalized Thresholding of Large Covariance Matrices. *Journal of the American Statistical Association*, 104(485):177–186, 2009.

[63] Klaus Schwab. *The Fourth Industrial Revolution*. Currency, 2017.

[64] Gideon Schwarz. Estimating the Dimension of a Model. *The annals of statistics*, pages 461–464, 1978.

[65] Ali Shojaie. Differential Network Analysis: A Statistical Perspective. *Wiley Interdisciplinary Reviews: Computational Statistics*, page e1508, 2020.

[66] Ali Shojaie and Nafiseh Sedaghat. How Different are Estimated Genetic Networks of Cancer Subtypes? In *Big and complex data analysis*, pages 159–192. Springer, 2017.

[67] Dinesh Singh, Phillip G Febbo, Kenneth Ross, Donald G Jackson, Judith Manola, Christine Ladd, Pablo Tamayo, Andrew A Renshaw, Anthony V D'Amico, Jerome P Richie, et al. Gene Expression Correlates of Clinical Prostate Cancer Behavior. *Cancer cell*, 1(2):203–209, 2002.

[68] Todd Stephenson. An Introduction to Bayesian Network Theory and Usage. Technical report, IDIAP, 2000.

[69] Zhou Tang, Zhangsheng Yu, and Cheng Wang. A Fast Iterative Algorithm for High-dimensional Differential Network. *Computational Statistics*, 35(1):95–109, 2020.

[70] Robert Tibshirani. Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[71] Tiejun Tong, Cheng Wang, and Yuedong Wang. Estimation of Variances and Covariances for High-Dimensional Data: A Selective Review. *Wiley Interdisciplinary Reviews: Computational Statistics*, 6(4):255–264, 2014.

[72] Christopher Tozzi. Big Data Industries: 5 Industries Being Reshaped by Data Analytics. `https://www.precisely.com/blog/big-data/big-data-industries-data-analytics`, 2021. [Data retrieved on the 1-June-2021].

[73] Google Trends. Big Data. `https://trends.google.com/trends/explore?date=2004-01-01%202021-01-01&q=big%20data`, 2021. [Data retrieved on the 20-October-2020].

[74] Beatriz Valcárcel, Peter Würtz, Nafisa-Katrin Seich al Basatena, Taru Tukiainen, Antti J Kangas, Pasi Soininen, Marjo-Riitta Järvelin, Mika Ala-Korpela, Timothy M Ebbels, and Maria de Iorio. A Differential Network Approach to Exploring Differences Between Biological States: an Application to Prediabetes. *PLoS One*, 6(9):e24702, 2011.

[75] Ann Van de Winckel, Stefan Sunaert, Nicole Wenderoth, Ron Peeters, Paul Van Hecke, Hilde Feys, Els Horemans, Guy Marchal, Stephan P. Swinnen, Carlo Perfetti, and De Weerdt Willy. Passive Somatosensory Discrimination Tasks in Healthy Volunteers: Differential Networks Involved in Familiar Versus Unfamiliar Shape and Length Discrimination. *Neuroimage*, 26(2):441–453, 2005.

[76] William Venables and Brian Ripley. *Modern Applied Statistics with S*. Springer, New York, fourth edition, 2002. URL `https://www.stats.ox.ac.uk/pub/MASS4/`. ISBN 0-387-95457-0.

[77] Wolfram Weckwerth, Marcelo Loureiro, Kathrin Wenzel, and Oliver Fiehn. Differential Metabolic Networks Unravel the Effects of Silent Plant Phenotypes. *Proceedings of the National Academy of Sciences*, 101(20):7809–7814, 2004.

[78] WHATWG. *HTML: HyperText Markup Language*, 1993. URL `https://html.spec.whatwg.org/`.

[79] Hadley Wickham. Reshaping Data with the Reshape Package. *Journal of Statistical Software*, 21 (12), 2007. URL `http://www.jstatsoft.org/v21/i12/paper`.

[80] Hadley Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. ISBN 978-3-319-24277-4. URL `https://ggplot2.tidyverse.org`.

[81] Hadley Wickham. *tidyr: Tidy Messy Data*, 2021. URL `https://CRAN.R-project.org/package=tidyr`.

[82] Hadley Wickham and Jay Hesselberth. *pkgdown: Make Static HTML Documentation for a Package*, 2020. URL `https://CRAN.R-project.org/package=pkgdown`. R package version 1.6.1.

[83] Hadley Wickham and Jim Hester. *readr: Read Rectangular Text Data*, 2021. URL `https://CRAN.R-project.org/package=readr`.

[84] Hadley Wickham, Jennifer Bryan, and Malcolm Barrett. *usethis: Automate Package and Project Setup*, 2021. URL `https://CRAN.R-project.org/package=usethis`. R package version 2.1.3.

[85] Hadley Wickham, Romain FranÃ§ois, Lionel Henry, and Kirill MÃŒller. *dplyr: A Grammar of Data Manipulation*, 2021. URL `https://CRAN.R-project.org/package=dplyr`.

[86] Hadley Wickham, Jim Hester, and Winston Chang. *devtools: Tools to Make Developing R Packages Easier*, 2021. URL `https://CRAN.R-project.org/package=devtools`. R package version 2.4.2.

[87] Yin Xia, Tony Cai, and Tianxi Cai. Testing Differential Networks with Applications to the Detection of Gene-Gene Interactions. *Biometrika*, 102(2):247–266, 2015.

[88] Bo Xu, Bernardo Gutierrez, Sumiko Mekaru, Kara Sewalk, Lauren Goodwin, Alyssa Loskill, Emily Cohn, Yulin Hswen, Sarah C. Hill, Maria M Cobo, Alexander Zarebski, Sabrina Li, Chieh-Hsi Wu, Erin Hulland, Julia Morgan, Lin Wang, Katelynn O'Brien, Samuel V. Scarpino, John S. Brownstein, Oliver G. Pybus, David M. Pigott, and Moritz U. G. Kraemer. Epidemiological Data from the COVID-19 Outbreak, Real-Time Case Information. *Scientific Data*, 7(106), 2020. doi: doi.org/10.1038/s41597-020-0448-0.

[89] Lingzhou Xue and Hui Zou. Regularized Rank-Based Estimation of High-Dimensional Nonparanormal Graphical Models. *The Annals of Statistics*, 40(5):2541–2571, 2012.

[90] Huili Yuan, Ruibin Xi, Chong Chen, and Minghua Deng. Differential Network Analysis via Lasso Penalized D-Trace Loss. *Biometrika*, 104(4):755–770, 2017.

[91] Cun-Hui Zhang. Nearly Unbiased Variable Selection Under Minimax Concave Penalty. *The Annals of Statistics*, 38(2):894–942, 2010.

[92] Teng Zhang and Hui Zou. Sparse Precision Matrix Estimation via Lasso Penalized D-Trace Loss. *Biometrika*, 101(1):103–120, 2014.

[93] Xiao-Fei Zhang, Le Ou-Yang, Xing-Ming Zhao, and Hong Yan. Differential Network Analysis from Cross-Platform Gene Expression Data. *Scientific reports*, 6(1):1–12, 2016.

[94] Sihai Zhao, Tony Cai, and Hongzhe Li. Direct Estimation of Differential Networks. *Biometrika*, 101 (2):253–268, 2014.

[95] Shuheng Zhou, Sara van de Geer, and Peter Bühlmann. Adaptive Lasso for High Dimensional Regression and Gaussian Graphical Modeling. *arXiv preprint arXiv:0903.2515*, 2009.

[96] Yunzhang Zhu and Lexin Li. Multiple Matrix Gaussian Graphs Estimation. *Journal of the Royal Statistical Society. Series B, Statistical methodology*, 80(5):927, 2018.

[97] Hui Zou. The Adaptive LASSO and its Oracle Properties. *Journal of the American Statistical Association*, 101(476):1418–1429, 2006.

Appendix

The programming language of choice for this dissertation is that of the programming language R, version 3.6.3, developed by the R Core Team [61].

R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria, available at: `https://cran.r-project.org/bin/windows/base/`

The respective versions of the packages utilised within this research are:

- BiocManager - version 1.30.10 [50]
- Bioconductor - version 2.46.0 [36]
- devtools - version 2.4.2 [86]
- dineR - version 1.0.0 [45]
- dplyr - version 1.0.7 [85]
- ggplot2 - version 3.3.0 [80]
- ggraph - version 2.0.5 [58]
- gtrendsR - version 1.4.8 [46]
- igraph - version 1.2.6 [21]
- MASS - version 7.3.54 [76]
- Matrix - version 1.2.18 [2]
- mixtools - version 1.2.0 [4]
- pkgdown - version 1.6.1 [82]
- progress - version 1.2.2 [20]
- readr - version 2.0.1 [83]
- reshape - version 1.4.4 [79]
- rhub - version 1.1.1 [22]
- sf - version 1.0.2 [56]
- spdep - version 1.1.8 [6]
- spls - version 2.2.3 [18]

- tidygraph - version 1.2.0 [57]
- tidyr - version 1.1.3 [81]
- usethis - version 2.1.3 [84]
- viridis - version 0.6.1 [30]

There was also a single figure that was generated through the use of HTML [78].

All code used throughout this research dissertation is available online at: `https://github.com/RicSalgado/DifferentialNetworks` and was executed on a personal computer containing the following specifications:

- Operating System: Windows 10 Home Single Language 64-bit.
- Processor: AMD Ryzen 5 1600 (AF) Hexa Core @ 3.2GHz.
- Memory: 16 GB DDR4 RAM @ 3200Mhz.

Supplementary Tables and Output

Table 9: Introductory Correlation Matrix

	Var 1	Var 2	Var 3	Var 4	Var 5	Var 6	Var 7	Var 8	Var 9	Var 10
Var 1	1.00	-0.26	0.21	0	0.00	0.00	0.00	0.25	0.00	-0.32
Var 2	-0.33	1.00	-0.20	0	0.00	0.43	0.00	0.00	0.30	0.00
Var 3	0.27	-0.20	1.00	0	0.00	0.00	0.00	0.00	0.00	0.00
Var 4	0.00	0.00	0.00	1	0.00	0.00	0.00	0.00	0.00	0.00
Var 5	0.00	0.00	0.00	0	1.00	0.00	0.32	0.27	-0.25	-0.29
Var 6	0.00	0.34	0.00	0	0.00	1.00	0.00	0.00	0.00	-0.26
Var 7	0.00	0.00	0.00	0	0.44	0.00	1.00	0.28	0.00	0.00
Var 8	0.30	0.00	0.00	0	0.42	0.00	0.32	1.00	0.00	0.00
Var 9	0.00	0.21	0.00	0	-0.30	0.00	0.00	0.00	1.00	0.00
Var 10	-0.36	0.00	0.00	0	-0.42	-0.29	0.00	0.00	0.00	1.00