



Norwegian University of
Science and Technology

Implementation and Application of Method for Differential Correlation Network Analysis.

Magnus Olav Helland

Master of Science in Physics and Mathematics

Submission date: August 2017

Supervisor: Rita de Sousa Dias, IFY

Co-supervisor: Eivind Almaas, IBT

Norwegian University of Science and Technology
Department of Physics

Abstract

A novel framework for differential co-expression networks have recently been developed at the Department of Biotechnology and Food Science at the Norwegian University of Science and Technology (NTNU). The method, referred to as the *CSD-framework* [1] aims to conserve more information about the co-expression patterns in the analyzed gene-expression data than other existing methods, by separating between three different forms of differential co-expression; *conserved* (C), *specific* (S), and *diverging* (D).

As a contribution to this project, a new program, *CSD-CS*, for fast and easy generation of differential co-expression networks using the CSD-framework, was developed in this thesis. The entire software has been implemented, tested, and analyzed as part of this thesis, and are described in chapters 3 and 4.

An application of the software, and a corresponding network analysis, were performed to illustrate the full scope of the differential co-expression network methodology. The analysis were performed on real-life gene-expression data from rheumatoid arthritis (RA) patients, and a healthy control group. The analysis identified biology with known association to the disease-state, such as immune system and inflammation related biology. It also identified the genes CD5, CCL21 and CCR6 in the network, which has been associated with RA. Additionally, enrichment of genes related to PRDM1 and TYK2, two other RA-associated genes were identified. Finally, the analysis identified several genes with possible RA-related functions, including CD72, FOXD1, AP8B2 and HAPLN4, which could be interesting to analyze further.

Some of the findings from the analysis did not have any known relation to the disease-state. The CSD-method is designed to only capture co-expression patterns related to genes that are tightly co-regulated across all patients, in at least one of the datasets. It is therefore reasonable to assume that the constructed network captures only essential and/or disease-related expression-patterns. This means that the network provides a good basis for generating hypotheses about RA, and that the findings with unknown disease-relevance may point to biology with currently undiscovered or indirect association to RA.

Sammendrag

Et nytt rammeverk for differensielle koekspressionsnettverk har nylig blitt utviklet ved Institutt for Bioteknologi og Matvitenskap ved Norges Tekniske og Naturvitenskaplige Universitet (NTNU). Metoden, kjent som *CSD-metoden* [1], bevarer mer av koekspressionsmønstrene i de analyserte genuttrykksdataene, enn andre eksisterende metoder, ved å skille mellom tre forskjellige former for differensiell koekspressjon; *konservert* (C), *spesifikk* (S), og *divergerende* (D).

Som et bidrag til denne metoden ble et nytt program, *CSD-CS*, for rask og enkel konstruksjon av differensielle koekspressionsnettverk ved hjelp av *CSD*-metoden, utviklet i denne masteroppgaven. Hele programmet har blitt implementert, testet og analysert som en del av denne oppgaven, og er beskrevet i kapittel 3 og 4.

En anvendelse av programmet, og tilhørende netverksanalyse, ble utført med den hensikt å illustrere hele metodologien bak differensielle koekspressionsnettverk. Analysen ble utført på virkelige genuttrykksdata fra pasienter med revmatoid artritt (RA) og en frisk kontrollgruppe. Analysen identifiserte biologi med assosiasjoner til sykdomstilstanden, slik som immunsystem- og inflammsjonsrelatert biologi. Den identifiserte også genene *CD5*, *CCL21* og *CCR6* i nettverket, som har blitt assosiert med RA. I tillegg ble det detektert en anrikelse av gener relatert til *PRDM1* og *TYK2*, som er to andre RA-assosierte gener. Analysen fant også flere gener med mulig RA-relaterte funksjoner, inkludert *CD72*, *FOXD1*, *AP8B2* and *HAPLN4*, som er kan være interessante prospekter for fremtidige analyser.

En del av funnene i analysen har ingen kjent relasjon til RA. *CSD*-metoden er imidlertid designet for kun fange opp koekspressionsmønstre relatert til gener som er tett samregulerte over alle pasientene i minst ett av datasettene. Det er derfor rimelig å anta at det konstruerte nettverket kun gjengir essensielle og/eller sykdomsrelaterte koekspressionsmønstre. Dette gjør at nettverket er et godt hypotesegenererende verktøy for RA, og at funnene fra analysen med ukjent sykdomstilknytning potensielt peker på biologi med foreløpig uoppdaget eller indirekte tilknytning til RA.

Preface

This thesis marks the end of my 5 years as a student at the Norwegian University of Science and Technology (NTNU) in Trondheim. The project concludes my M.Sc degree in *Applied Physics and Mathematics* where I have specialized in *Biophysics*.

I would like to start by expressing my sincere gratitude towards my supervisor, professor Eivind Almaas, at the Department of Biotechnology and Food Science, for his assistance with my thesis. His support and guidance have been indispensable for the outcome of this thesis. His enthusiasm and ability to communicate his insights, have made this project an inspiring conclusion to my studies at NTNU. I would also like to thank his Ph.D student, André Voigt, for his input and assistance, and for his help in clarifying methods and topics in this thesis. From the Department of Physics I would like to thank my internal supervisor, Rita de Sousa Dias, for her thorough revision of the thesis and constructive comments and feedback.

Last but not least, I would like to thank all of my friends and family. I would never have managed this without your support. Thank you for all the good times, and for making my time here both entertaining and meaningful. You are the ones that have made my stay in Trondheim truly unforgettable.

Magnus Olav Helland
August 2017

Table of Contents

Abstract	i
Abstract (norwegian)	i
Preface	ii
Table of Contents	v
List of Tables	ix
List of Figures	xiii
Abbreviations	xiv
1 Introduction	1
2 Background	3
2.1 Network theory	3
2.1.1 Adjacency matrix and node degree	4
2.1.2 Degree distribution and network topology	4
2.1.3 Hubs, communities and modularity	6
2.1.4 Louvain community detection algorithm	9
2.1.5 The significance of hubs and communities	9
2.1.6 Node parameters	10
2.1.7 Network parameters	11
2.1.8 Multilayer networks	12
2.2 Gene co-expression networks	13
2.2.1 Construction of gene co-expression networks	13
2.2.2 CSD-framework for differential co-expression networks	15
2.3 Microarray data	17
2.3.1 Transcriptome profiling	17
2.4 Rheumatoid arthritis	18

2.5	Computer science	18
2.5.1	Asymptotic time complexity	18
2.5.2	Parallel programming	19
2.6	Correlation and similarity measures	20
2.6.1	Pearson correlation	20
2.6.2	Spearman correlation	20
2.6.3	Kendall's tau	20
2.6.4	Biweight midcorrelation	21
2.6.5	Mutual information	22
2.7	Statistics	22
2.7.1	Fisher exact test	22
2.7.2	Bonferroni correction	23
2.7.3	Benjamini-Hockberg method	24
3	Methods: Software	25
3.1	Program overview	26
3.2	Setting up and using the program	26
3.3	Program parameters	28
3.3.1	Input file	28
3.3.2	Subsample size	30
3.3.3	Importance level	30
3.3.4	Input data and file format	30
3.3.5	Output data	30
3.3.6	Correlation method	31
3.3.7	Subsampling parameters	31
3.3.8	Threads	31
3.3.9	The <i>outfilePrefix</i> parameter	32
3.3.10	Default values and unnecessary parameters	32
3.4	Data formats	32
3.5	Algorithms	33
3.5.1	Preprocessing gene-IDs	33
3.5.2	Correlation calculations	33
3.5.3	Subsampling algorithms for variance estimation	34
3.5.4	Variance estimation	36
3.5.5	Calculating CSD-scores	36
3.5.6	Estimation of importance thresholds	37
3.6	Parallelization	37
4	Results: Software	39
4.1	Subsampling algorithms	39
4.2	Time consumption and parallel efficiency	41
4.2.1	Time complexity analysis	41
4.2.2	Parallelization of correlation and variance calculations	43
4.2.3	Further parallelization	44
4.2.4	Memory usage	45
4.3	Program testing	45

5	Results: Application to empirical gene expression data	47
5.1	Data collection	47
5.1.1	Rheumatoid arthritis expression data	47
5.1.2	Control group expression data	48
5.2	Network construction	49
5.3	Network analysis: tools and procedure	49
5.4	Network properties	51
5.4.1	Component analysis	53
5.4.2	Module analysis	57
5.5	Biological analysis	58
5.5.1	Enrichment analysis	58
5.5.2	Biological functions of central nodes	60
5.5.3	Localizing disease-associated genes	63
6	Discussion and further work	65
6.1	CSD-CS	65
6.1.1	Code parallelization	65
6.1.2	Memory reduction	66
6.1.3	Increased functionality	67
6.2	Network analysis	68
7	Conclusion	71
	Bibliography	73
	Appendix A	83
	Appendix B	84
	Appendix C	89
	Appendix D	92
	Appendix E	95
	Appendix F	99
	Appendix G	100

List of Tables

2.1	Example of contingency table for two nominal variables.	23
3.1	Overview of the indicators, and arguments used by <i>CSD-CS</i> . Upon calling the program in the terminal, it should be followed by some combination of these arguments to specify the execution of the program. Each argument should be preceded by the appropriate indicator to specify which parameter it sets. The arguments given in italic are keywords specific for the indicator. Attempting to set one of these parameters using anything other than a keyword will return an error.	29
3.2	The table shows the default values of the input parameters of the <i>CSD-CS</i> -program.	32
4.1	The time-complexity of different processes in the <i>CSD-CS</i> software. . . .	41
5.1	The input parameters used by <i>CSD-CS</i> in creating the network analyzed in this thesis.	49
5.2	The number of nodes and edges in the networks and its major components, along with the distribution of specific interaction types.	51
5.3	The network parameters for the full network and its two main components.	52
5.4	The table shows the average of the network parameters calculated for the degree-preserving random networks based on the S-component. Difference in parameter values between the S-component and the random networks are highlighted in red. The radius is not comparable as the random network give rise to pairs of nodes separated from the main component yielding a radius of 1.	55
5.5	The table shows the average of the network parameters calculated for the degree-preserving random networks based on the CD-component. Difference in parameter values between the CD-component and the random networks are highlighted in red. The radius is not comparable as the random network give rise to pairs of nodes separated from the main component yielding a radius of 1.	55

5.6	The table shows a comparison between the Spearman correlation between different node parameters within in the CD-dominated network component (top half) and degree-preserving random networks based on the CD-component (bottom half). Blue indicates strong correlations, while red indicates notable differences in correlation scores.	56
5.7	The table shows a comparison between the Spearman correlation between different node parameters within in the S-dominated network component (top half) and degree-preserving random networks based on the S-component (bottom half). Blue indicates strong correlations, while red indicates notable differences in correlation scores.	57
5.8	The 10 biggest modules identified by the Louvain algorithm sorted by the number of nodes.	57
5.9	The table shows the enrichment scores for the C-, S- and D-networks most relevant for RA. The listed scores are not the maximum observed scores, but are the ones most relevant with respect to RA.	60
5.10	The table shows the most interesting findings from the modular enrichment analysis. The listed scores are not the maximum observed scores, but the ones most relevant with respect to RA.	61
5.11	The highest degree nodes from the network.	62
5.12	The nodes with the highest clustering coefficient and degrees higher than 13.	62
5.13	The nodes with the highest betweenness centrality from within one of the two largest components.	62
5.14	The nodes with the highest closeness centrality from within one of the two largest components.	62
5.15	The nodes with the highest neighborhood connectivity of the nodes with a degree higher than 15.	62
5.16	The nodes with the highest topological coefficient from nodes with degree higher than 15.	62
6.1	A comparison of different similarity scores between 4 randomly chosen genes (<i>A</i> , <i>B</i> , <i>C</i> and <i>D</i>) in the rheumatoid arthritis data set. The binning of the values for calculation of MI-scores was chosen to generate scores in the same size range as the rest of the similarity measures (for the purpose of comparison).	67
1	This is a list of 52 genes with suspected association to rheumatoid arthritis.	83
2	Network parameters for the degree-preserving random networks generated from the S-component.	89
3	Network parameters for the degree-preserving random networks generated from the CD-component.	89
4	Spearman correlation between different node parameters within in the CD-dominated network component.	93
5	Spearman correlation between different node parameters within in the S-dominated network component.	94

6	The table shows the Bonferroni corrected p-values related to the Spearman correlations between different node parameters for nodes within the network as a whole (aside from minor components and single nodes). . .	95
7	The table shows the Bonferroni corrected p-values related to the Spearman correlation between different node parameters for nodes within the CD-dominated network component.	95
8	The table shows the Bonferroni corrected p-values related to the Spearman correlation between different node parameters for nodes within the S-dominated network component.	95
9	The table shows the Bonferroni corrected p-values related to the Spearman correlation between different node parameters for nodes within module 0.	96
10	The table shows the Bonferroni corrected p-values related to the Spearman correlation between different node parameters for nodes within module 1.	96
11	The table shows the Bonferroni corrected p-values related to the Spearman correlation between different node parameters for nodes within module 2.	96
12	The table shows the Bonferroni corrected p-values related to the Spearman correlation between different node parameters for nodes within module 3.	96
13	The table shows the Bonferroni corrected p-values related to the Spearman correlation between different node parameters for nodes within module 5.	97
14	The table shows the Bonferroni corrected p-values related to the Spearman correlation between different node parameters for nodes within module 6.	97
15	The table shows the Bonferroni corrected p-values related to the Spearman correlation between different node parameters for nodes within module 7.	97
16	The table shows the Bonferroni corrected p-values related to the Spearman correlation between different node parameters for nodes within module 10.	97
17	The table shows the Bonferroni corrected p-values related to the Spearman correlation between different node parameters for nodes within module 15.	98
18	This is a list of 60 genes found in the network related to transcription of the PRDM1 gene.	99
19	This is a list of 19 genes found in the network related to knockdown of the TYK2 gene.	100

List of Figures

1.1	The number of citations for the most cited papers in different areas related to complexity, over time. It shows a great interest in network science starting approximately two decades ago. The figure is from [2].	2
2.1	Example of an unweighted, undirected network and its corresponding adjacency matrix.	4
2.2	The figure shows the different network regimes of a <i>Erdős-Rényi</i> [3] random network as a function of N_G/N (a). (b) shows the subcritical regime, (d) the supercritical regime, and (e) shows the connected regime. The figure is from [2].	5
2.3	The figure, from [4], shows an example of a scale-free network. The network represents protein interactions in yeast where red nodes represent essential proteins, orange proteins of some importance, while yellow and green indicate proteins of unknown and lesser significance respectively.	7
2.4	Different definitions of a community; (a) maximum-clique, (b) strong-community, (c) weak-community. Figure is from [2].	8
2.5	Illustration of a multilayer network. Figure from [5].	13
2.6	Graphical representation of the C-, S-, and D-type co-expression regions of interest.	16
2.7	Graphical representation of the definitions of the asymptotic time-complexity parameters $\Theta(f(n))$, $O(f(n))$, and $\Omega(f(n))$	19
3.1	The figure illustrates the program modularity. Each module represents one of the one of the main processes of the program, and can be performed independently of each other.	27
3.2	The figure illustrates how the program can be executed using the linux terminal. Notice how all parameters are preceded by an appropriate identifier.	28

3.3	Illustration of the idea behind the preprocessing algorithm, for the case of four input files. Each row is a sorted gene-list from one of the input-files. The gene lists are iterated over incrementally, and all genes not present in all data-sets are ignored. By sorting the gene-lists first, the algorithm only needs to iterate over all elements once, allowing the comparison to run in linear time. The color is added to highlight the successful comparisons and make them easier to separate from one another.	34
4.1	The number of subsamples generated by the different subsampling algorithms. The random subsampling algorithm generally outperforms the other two methods.	40
4.2	The figure shows the CPU-time used by CSD-CS for input data containing 100 measurements for a various number of genes $n \in [1000, 25000]$. The CPU-time is initially dominated by the contribution from the threshold estimation algorithm, which runs in approximately constant time for a given combination of p and s_{imp} . The figure shows the strong influence of the number of genes in the data set on the time consumption of the program. For large n the time consumption is purely exponential and approximately equal the time consumption of the correlation calculations.	42
4.3	The figure shows how the CPU-time of the program depends on the number of genes in the analyzed data-sets. A second degree polynomial fit (the red line) verifies the estimated time-complexity of the program $O(n^2)$. . .	42
4.4	Result of parallelizing the correlation and variance calculations. The top figure shows how the wall-time of the program section follows the the number of cores used. The middle plot show the speed-up-factor S , and the bottom figure shows the efficiency E of the parallelization.	43
4.5	The effect on the overall program performance from all the parallelized code.	44
4.6	The maximum memory usage of the program as a function of the number of genes in the data sets n	46
5.1	The figure shows the full CSD-network produced by CSD-CS as visualized by Cytoscape. Conserved interactions are indicated by blue edges, while specific and diverging are shown in green and red respectively. . . .	52
5.2	The figure shows the degree distribution of the full network. Subplot 2 and 3 shows log-log plots of the degree-distribution. As indicated by the fitted line in subplot 3, the degree-distribution follows a heavy-tailed power-law (with $\gamma = 1.886$).	53
5.3	The neighborhood-connectivity distributions of the full network and the two main components, along with a fitted power law (red line) . Interestingly the CD- and S-components show opposite assortativity.	54
5.4	The 10 largest modules detected in the CSD-network by using the Louvain community detection algorithm.	58
5.5	The figure shows the two small components where the RA-associated genes CCL21 and CCR6 are located, as well as module 10 where CD5 is located. The relevant genes are highlighted by a different color.	64

1	Example illustrating the <i>ged</i> file-format.	84
2	Example illustrating the <i>corr</i> file-format.	85
3	Example illustrating the <i>csd</i> file-format	86
4	Example illustrating the <i>all</i> file format.	87
5	Example illustrating the <i>network</i> file-format.	88
6	Example of a degree-preserving random network generated from the S-dominated component in the arthritis-network.	90
7	Example of a degree-preserving random network generated from the CD-dominated component in the arthritis-network.	91

Abbreviations

C	=	Conserved
S	=	Specific
D	=	Diverging
GB	=	Giga bytes
HLA	=	Human Leukocyte Antigen
NK-cell	=	Natural killer cell
RA	=	Rheumatoid Arthritis

Chapter 1

Introduction

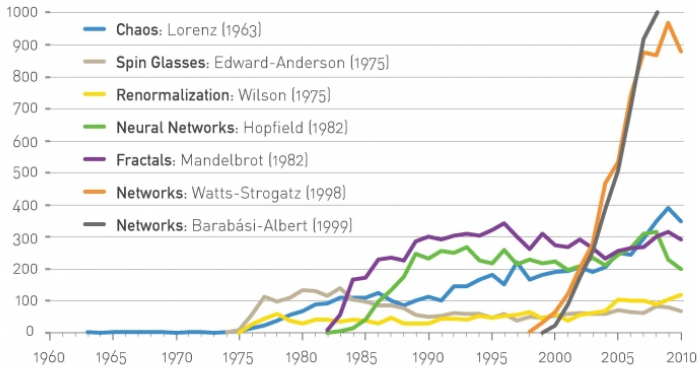
"Reductionism, as a paradigm, is expired, and complexity, as a field is tired. Data-based mathematical models of complex systems are offering a fresh perspective, rapidly developing into a new dicipline: network science."

- Albert-Lázló Barabási (2011) [6]

Network science is a rapidly developing field of science, and has seen great interest over the last two decades. Its goal is to analyze and characterize the structure, interactions and behaviour of complex systems. A comparison between the number of citations of the most prominent papers from network science and other fields related to complex systems (figure 1.1) illustrates the impact of this novel scientific area. Although the field itself is young, its theoretical foundations is by no means new. The history of graph theory, the fundamental formalism of network theory, can be traced all the way back to Leonhard Euler's paper "The seven bridges of Königsberg" from 1736. Network science distinguish itself from graph theory however, in its more empirical nature. The focus is on function and utility, and methods are judged by the insights offered about a system's properties and behaviour [2]. In addition to graph theory, network theory employs methods from statistics, statistical physics [7], control and information theory [8], and computer science [9, 10].

Network science has found applications across many different branches of science, from social networks [11] to the metabolic networks in cells [12, 13]. This thesis will explore an application of network science within the area of *systems biology*. The background for this thesis is the development of a new framework for generation of *differential gene co-expression networks* at the *Institute for Biotechnology and Food Science* at NTNU. A differential gene co-expression network captures similarities and differences in gene-expression patterns across suitably chosen tissues, conditions or specimens, providing insights into the genetic regulatory machinery causing different phenotypes. The novel framework, referred to as the *CSD-framework* [1], separates itself from existing methods

Figure 1.1 The number of citations for the most cited papers in different areas related to complexity, over time. It shows a great interest in network science starting approximately two decades ago. The figure is from [2].



by distinguishing between three different forms of co-expression. It evaluates *conserved* (C), *specific* (S) and *diverging* (D) co-expression, thus preserving more information about co-expression patterns than comparable methods such as *DCGL/DCe* [14].

The main goal of this thesis was the development of a software for construction of differential gene co-expression networks, using the CSD-framework. The software was developed from scratch with focus on easy and efficient network construction. Additionally, an application of the software and a corresponding network analysis was performed to illustrate the full scope of a co-expression network analysis. As gene expression and regulation can be highly tissue-specific, differential co-expression analysis is a suitable tool for detecting genes related to disease states [15, 16], tissue types [17] and developmental stages [18], because these genes are more likely to be regulators that underlie phenotypic differences [19]. For the analysis performed in this thesis, gene expression data from patients with the disease *Rheumatoid Arthritis* (RA), and a healthy control group was evaluated. The goal of the analysis was to identify relevant biology in the network and investigate potential roles of genes in the disease state.

The thesis consists of 7 chapters. Chapter 2 presents relevant theory and background for the project. Chapter 3 and 4 are related to the software development, where chapter 3 describes the operation and functionality of the developed software, and chapter 4 features an analysis of program performance. Chapter 5 then presents the network generated from application of the software, and the corresponding network analysis. Finally chapter 6 and 7 contains a discussion of the results, and a conclusion.

The reader of this thesis is expected to have basic knowledge of biochemistry, cell biology, programming, and mathematics. Detailed knowledge should however not be necessary, as relevant concepts and theory is presented in chapter 2.

Background

This chapter aims to give readers unfamiliar with the main topics of this thesis and its methods an introduction to relevant concepts and background theory. Most topics discussed will be of direct relevance to later chapters, while some of the content is included for overview or completeness. The section on network theory is mainly drawn from different parts of Barabasi's book "Network Science" [2]. The reader is therefore referred to his book for a more in-depth discussion of the subject, if so required. The general framework for construction of gene co-expression networks presented in section 2.2.1 follows the one proposed by Zhang and Horvath in their article "A General Framework for Weighted Gene Co-Expression Network Analysis" [20]. It does not describe their proposed method, but simply lists the general steps of gene co-expression network construction.

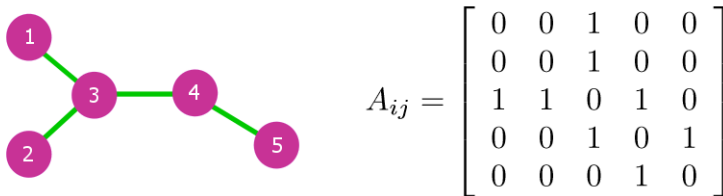
2.1 Network theory

"A *network* consists of a set of items, which we will call *vertices* or sometimes nodes, with connections between them called *edges*." This is the definition of a network as presented by Newman in his review article "The Structure and Function of Complex Networks" [21]. The concept is simple; a set of objects (from here on referred to as *nodes*) share a connection or property which is represented by an *edge*, also often called *links*. This simple representation, originating in the mathematical branch of graph theory, have proven effective in capturing and describing complex systems and behaviour. The network representation can be used to model a wide range of systems, across different scientific branches. Social networks [11] for instance are networks in which nodes represent persons, and edges some sort of relation between persons, be it friendliness, sexual relations or familiarity. Other examples of networks include the World Wide Web [22], the internet [23], infrastructure [24], metabolic networks in biology [12, 13], protein-protein interaction networks [25], gene co-expression networks [26] and the human disease network [27] to mention but a few. Network science has proven a powerful tool and can be used to describe and predict complex behaviors such as spreading phenomena [28], system robustness [29], cascading failures [30], structural phase transitions [31], synchronization [32] and cooperation [33].

2.1.1 Adjacency matrix and node degree

In order to completely describe a network, it is necessary to keep track of all involved objects and their connections. A simple way of doing this is through an *adjacency matrix*. In an adjacency matrix A_{ij} each matrix element a_{ij} quantifies the connection between node i and j , hence representing an edge in the network. For an *unweighted* network, each matrix element is assigned a value of either 1, or 0 indicating an edge between two nodes, or a lack thereof. For the case of *weighted* networks the elements of the adjacency matrix are assigned a value from the interval $[0, 1]$, indicating the strength of the connection. One can also distinguish between directed and undirected networks. For an *undirected* network $a_{ij} = a_{ji}$, resulting in an adjacency matrix symmetrical around the diagonal. In the case of a *directed* network the direction in which an edge points convey meaningful information. Thus, in the adjacency matrix of a directed network a_{ij} is not necessarily the same as a_{ji} . This distinction gives rise to asymmetrical adjacency matrices for directed networks. As the networks relevant for this thesis are all undirected and unweighted, all further discussions assumes an unweighted, undirected network, unless explicitly stating otherwise.

Figure 2.1 Example of an unweighted, undirected network and its corresponding adjacency matrix.



A fundamental property of networks are the *degree distribution*. The degree refers to the number of nodes directly connected to a node i through edges. The degree of a node thus describes the number of nearest neighbors of that node. For an undirected network the degree of node i is given from its adjacency matrix by:

$$k_i = \sum_j a_{ij} \quad (2.1)$$

The degree distribution of a network is the probability that a randomly selected node from the network has a given degree. The nature of the degree distribution have important implications for the properties of a network, as described in the section below.

2.1.2 Degree distribution and network topology

The section below is mainly based on the contents from Barabasi network science book [2]. Only main results are stated, and so the reader is referred to the book for a full derivation. Several figures have been borrowed from [2] under its *Creative Commons license*.

The degree distribution of a network convey important information about the structure of the network. To illustrate this, let's consider two famous network models; *random networks* as described by E.N. Gilbert [34], and *scale-free networks* [22, 35]. A random network can be constructed by iteratively adding new nodes to the network, where each new node is connected to the network through a predefined number of edges. The probability of an edge occurring between two nodes is equal for all nodes in the network. The resulting degree distribution of a Gilbert random network is given by the binomial distribution,

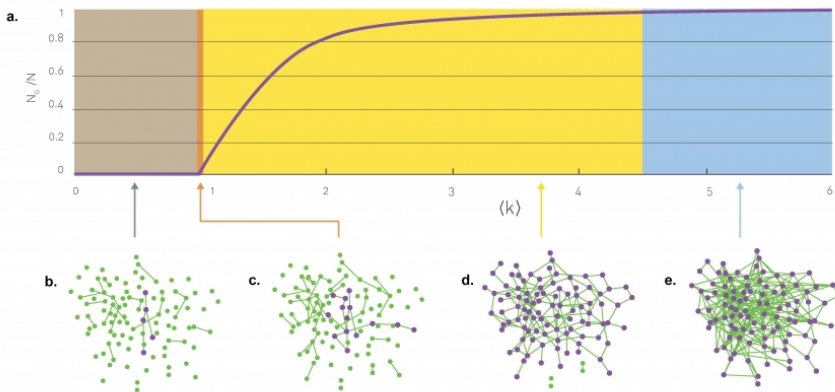
$$p_k = \binom{N-1}{k} p^k (1-p)^{N-1-k} \quad (2.2)$$

where N is the number of nodes, k is the degree, and p is the probability of a link occurring between two nodes. Most real networks are sparse, causing the average node degree to be much less than the number of nodes ($\langle k \rangle \ll N$). In this limit the degree distribution is well approximated by the Poisson distribution:

$$p_k = e^{-\langle k \rangle} \frac{\langle k \rangle^k}{k!} \quad (2.3)$$

The degree distribution thus has its peak at $k = \langle k \rangle$, and a standard deviation of $\sigma_k = \langle k \rangle^{(1/2)}$. In [2] three different topological regimes originating in this distribution are discussed; *subcritical*, *supercritical* and *connected* 2.2.

Figure 2.2 The figure shows the different network regimes of a *Erdős-Rényi* [3] random network as a function of N_G/N (a). (b) shows the subcritical regime, (d) the supercritical regime, and (e) shows the connected regime. The figure is from [2].



The subcritical regime appears when $0 < \langle k \rangle < 1$. In this regime only a small number of nodes are connected hence the network consists mainly of many tiny clusters. The number of genes in the largest cluster N_G scales proportionally to $\ln(N)$ and so the relative size of the largest cluster goes to zero: $\lim_{N \rightarrow \infty} \frac{N_G}{N} \rightarrow 0$.

In the supercritical regime ($\langle k \rangle > 1$) a large connected component emerges, developing a more connected network. In the vicinity of the *critical point* $\langle k \rangle = 1$, the relative size of the giant component scales as $\frac{N_G}{N} \sim \langle k \rangle - 1$. Finally in the connected regime ($\langle k \rangle > \ln(N)$), $N_G \approx N$ and the giant component absorbs more or less all nodes in the network.

Real networks are usually not random. It turns out that the degree distribution of many real networks follows a power law,

$$p_k \sim k^{-\gamma} \quad (2.4)$$

which is the defining characteristic of *scale-free* networks [35]. The important difference between random and scale-free networks lies in the tail of the degree distribution. While the great majority of nodes in a random network will have a degree of $\langle k \rangle \pm \sqrt{\langle k \rangle}$, resulting in a fairly homogeneous network, scale free networks lack a scale on the degree of a node. What this means is that a randomly drawn node from a scale-free network can have a wide range of different degrees, and distribution parameters such as mean and variance yields little practical information about average node properties. Although the scale-free property is used to describe any network following a power law distribution, the scale-free behaviour is especially prominent for networks with $\gamma \in \langle 2, 3 \rangle$. The reason can be seen from the n^{th} -moments of the degree distribution of networks with power law distributions¹:

$$\langle k^n \rangle = C \frac{k_{max}^{n-\gamma+1} - k_{min}^{n-\gamma+1}}{n - \gamma + 1} \quad (2.5)$$

As $k_{max} = k_{min} \cdot N^{\frac{1}{1-\gamma}}$ for power-law networks, all moments for which $n > \gamma - 1$ as $N \rightarrow \infty$ diverge. The result is that the variance of the degree distribution ($\sigma^2 = \langle k^2 \rangle - \langle k \rangle^2$) diverges for power-law networks with $2 < \gamma < 3$. This allows the existence of nodes of arbitrarily high degree. The existence of such nodes greatly affects the topology of the corresponding network, making it more heterogeneous by introducing hubs and communities. Figure 2.3 shows an example of a scale-free network. The scale-free network in figure 2.3 shows a much more heterogeneous topology than the random networks from figure 2.2.

2.1.3 Hubs, communities and modularity

A prominent feature of many real networks is the presence of *hubs* and *communities*. A hub refers to a highly connected node, in other words, a high degree node. A community however does not have a clear definition. Instead it is related to a set of hypotheses [2]:

1. **Fundamental hypothesis:**

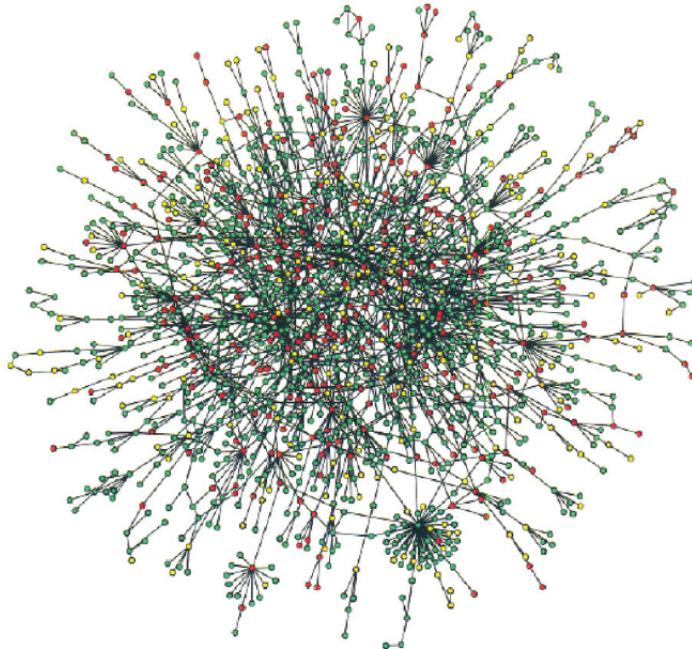
“A network’s community structure is uniquely encoded in its wiring diagram.”

2. **Connectedness and density hypothesis:**

“A community is a locally dense connected subgraph in a network.”

¹See Barabasi Network Science chapter 4 for the full derivation of equation (2.5)

Figure 2.3 The figure, from [4], shows an example of a scale-free network. The network represents protein interactions in yeast where red nodes represent essential proteins, orange proteins of some importance, while yellow and green indicate proteins of unknown and lesser significance respectively.



3. Random hypothesis:

“Randomly wired networks lack an inherent community structure.”

4. Maximal modularity hypothesis:

“For a given network the partition with maximum modularity corresponds to the optimal community structure.”

The first hypothesis simply states that there is a ground truth about a network’s community organization, which can be learned from the adjacency matrix of the network. The second hypothesis restricts the definition of a community by requiring that all nodes must be reachable from other members of the same community. The definition of a locally dense subgraph however is not well defined. In [2] three different definitions are presented; *Maximum clique*, *Strong communities*, and *Weak communities*.

A maximum clique is a *complete*² subgraph within a network. The drawback of such a definition is that complete subgraphs larger than triangles are rare, hence it may be too restrictive. An alternative definition is that of strong communities, in which it is required

²In graph theory a complete graph is one in which all nodes are connected to all others.

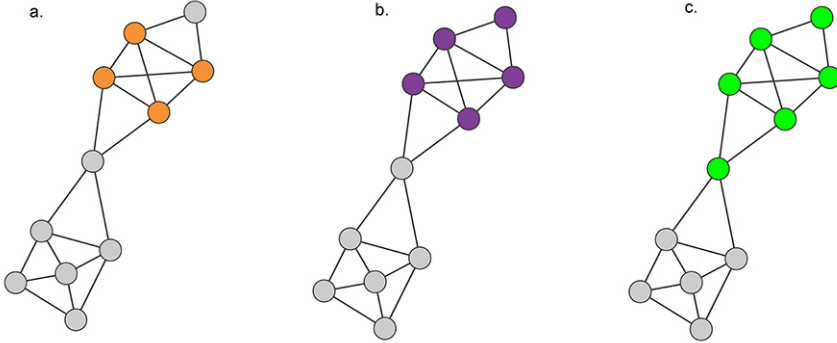
that each node within a community has a higher *internal degree* than *external degree*. The internal degree of a node k_i^{int} refers to the number of links node i makes with other nodes within the community. The external degree k_i^{ext} is the conjugate set of neighboring nodes, so that $k_i = k_i^{int} + k_i^{ext}$. Mathematically, the criterion for including a node i in the strong community C can be expressed as:

$$k_i^{int}(C) > k_i^{ext}(C) \quad (2.6)$$

Finally, the definition of a weak community requires the the total internal degree of a subgraph to be greater than its external degree.

$$\sum_{i \in C} k_i^{int} > \sum_{i \in C} k_i^{ext} \quad (2.7)$$

Figure 2.4 Different definitions of a community; (a) maximum-clique, (b) strong-community, (c) weak-community. Figure is from [2].



The above definitions of communities are given in decreasing order of rigidity, where each consecutive definition entails the communities of the preceding one. These examples are meant to illustrate that identification of communities is dependent on the definition used.

The third hypothesis is related to the concept of *modularity*. The implication of this hypothesis is that dense subgraphs can be verified by comparing its link density with a random network consisting of the same number of nodes. The modularity is thus defined as:

$$M_c = \frac{1}{2L} \sum_{(i,j) \in C_c} (A_{ij} - p_{ij}) \quad (2.8)$$

where L is the total number of links or edges in the network, and $p_{ij} = \frac{k_i k_j}{2L}$ is the probability of a random connection between two nodes of degree k_i and k_j . Equation 2.8 can be equivalently formulated as,

$$M_c = \frac{L_c}{L} - \left(\frac{k_c}{2L} \right)^2 \quad (2.9)$$

where L_c is the total number of links within community C_c and k_c is the total degree. This equation can be generalized to evaluate an overall modularity score for a given way of partitioning a network into n_c communities:

$$M = \sum_{c=1}^{n_c} \left[\frac{L_c}{L} - \left(\frac{k_c}{2L} \right)^2 \right] \quad (2.10)$$

2.1.4 Louvain community detection algorithm

The Louvain algorithm [36] is a graph partitioning algorithm used for identifying communities within a network. The exact formulation of the problem, involving identification of an optimal community partition, is known to be a computationally intractable optimization problem. As a result there are many different algorithms for finding reasonably good partitions of a network in a reasonable amount of time. The Louvain algorithm uses an iterative two step approach to identify communities. Initially all nodes are assigned their own community, resulting in an equal number of nodes and communities. The first step then consists of the following process: for each node i , the change in modularity caused by moving node i to the community of its neighboring node j is calculated for all possible j . Node i is then placed in the community that maximizes the modularity score, whether it means moving to a neighboring community or staying in its current one. This process is repeated until no further improvements in modularity score can be achieved. This might involve multiple evaluations of each node. Phase one thus terminates upon reaching a local modularity maximum.

The second phase of the algorithm consists of constructing a new network, where each community detected in phase one is replaced by a node. The weight of the link between two nodes is calculated as the sum of the links connecting nodes in the two communities. After the generation of the new network, step one may be reapplied. These two processes are repeated iteratively until no further increase in modularity score occurs.

The Louvain algorithm has been shown to produce good partitions with high modularity scores in a short amount of time [36]. The majority of time is spent in the first pass through step one, as the problem size is reduced quickly through the generation of the meta-networks in phase 2. The number of passes through both phase one and two is therefore generally a small number even for large networks.

2.1.5 The significance of hubs and communities

The discussion from the two previous sections raises the following question: Why are hubs and communities of such interest? Knowing that hubs and modules do not arise from random network construction, is it reasonable to assume that their presence convey some information about the nature of the system? Two famous network models provide some insight into these questions; the Barabasi-Albert model [35] and the Bianconi-Barabasi model [37].

In the Barabasi-Albert model, the construction of a network is based on two mechanisms, *growth* and *preferential attachment*. This means that a network continuously evolves and that the probability of attaching to a node increases with its degree. The

Bianconi-Barabasi model is somewhat similar to the Barabasi-Albert model, but introduces a node property called *fitness*. In this model the probability of connecting a new node to an existing one is dependent on both the existing node's degree and its fitness. The interesting about these models is the fact that the resulting networks are scale-free. Although the modeling mechanisms is fairly simple, scale-free networks emerge, verifying the distinction between random networks and scale-free ones. This result gives an indication that non-random organizing principles and system properties are what give rise to the scale-free topology. It is generally recognized that the topology and evolution of real networks are in fact governed by robust organizing principles [7].

Hubs and communities have been shown to have prominent roles in the structure and properties of networks [7, 25, 29]. In protein-protein interaction networks for example, it has been shown that deletion of hub genes is more likely to be lethal for the organism than deletion of non-hub genes [38]. Different measures of topological properties is of particular interest in *static analysis* [39], in which the topology of a network is correlated to function and behavior. This type of analysis is highly relevant for gene co-expression networks, which is the main focus of this thesis. In particular, analysis of network communities (also referred to as *modules* in relation to co-expression networks) are widely used and have provided insight into biological function and disease [40, 41, 42, 43].

2.1.6 Node parameters

Many different node parameters exists for identifying nodes of interest and evaluating their roles in a network. The node degree is an example of such a parameter. Several node parameters convey similar information, and so only the ones used in the network analysis presented later in this thesis will be described.

The clustering coefficient and centrality measures gives important information about the roles of different nodes in a network. A *cluster*, or a *component* in graph theory, is a connected subset of a network, for which addition of more nodes will break the connectedness. A properly placed link might connect two clusters, in which case the link will act as a *bridge*. The clustering coefficient of a node describes the degree to which its neighbors link to each other, acting as a measure of local link density around a node. It is defined as:

$$C_i = \frac{2L_i}{k_i(k_i - 1)} \quad (2.11)$$

where L_i is the number of links between the k_i neighbors of node i . For $C_i = 0$ none of node i 's neighbors are linked, while for $C_i = 1$ node i and its neighbors constitutes a complete graph.

The average clustering coefficient distribution gives the average of the clustering coefficients for all nodes n with k neighbors for $k = 2, 3, \dots$

Centrality is the concept of node or edge importance in a network. Thus different centrality measures assign a score to each node yielding information about its role in the network. The *betweenness centrality* measure C_B [44] is defined as the fraction of shortest paths between pair of nodes passing through a node i :

$$C_B(i) = \sum_{i \neq j \neq k} \frac{\sigma_{jk}(i)}{\sigma_{jk}} \quad (2.12)$$

Here σ_{jk} is the total number of shortest paths between nodes j and k while $\sigma_{jk}(i)$ is the number of shortest paths through i . Nodes involved in a *bridge* as described above, will typically have a high betweenness score, indicating their central role in information transmission between different parts of the network.

Another centrality measure is the *closeness centrality* [45]. The closeness centrality is defined as the reciprocal sum of shortest paths from a node i to all other nodes in the network. The shortest path between two nodes is also referred to as *geodesic path* between two nodes. Closeness is thus given by:

$$C_C(i) = \frac{1}{\sum_{j \neq i} d_g(i, j)} \quad (2.13)$$

where $d_g(i, j)$ is the geodesic distance between node i and j . The closeness centrality can be seen as a node's ability to spread information to all other nodes of the network.

A nodes eccentricity ϵ is defined as the maximum finite distance of a shortest path between i and any other node in the network. In the case of i being an isolated node, its eccentricity is set to zero.

The neighborhood connectivity of a node k_{ii} is the average connectivity (degree) of a node i 's k_i neighbors. From the neighborhoods connectivities in a network, the neighborhood connectivity distribution can be determined. It is given by the average of the neighborhood connectivities for all nodes i with degree k for $k = 1, 2, \dots$. The neighborhood connectivity distribution gives information about the assortative nature of a network. If a network is assortative, high degree nodes tend to connect to high degree nodes, and low degree nodes to other low degree nodes. The assortativity is related to degree correlations between the nodes in the network, which have implications for clustering of a network, its shortest paths, its diameter, and its robustness to perturbations [2].

Finally a topological coefficient for the network can be defined as follows [46]:

$$T_i = \frac{\text{mean}(J(i, j))}{k_i} \quad (2.14)$$

Here $J(i, j)$ is defined for all nodes j that share at least 1 neighbor with i . $J(i, j)$ is then the number of neighbors shared between node i and j , plus one if node i and j are connected directly. This topological coefficient is thus a measure of the extent to which nodes share neighbors with its neighbors.

2.1.7 Network parameters

There are different network parameters, that capture various properties of a network. The *diameter* of a network is defined as the largest geodesic path in a connected component. The *radius*, on the other hand, is the minimum of the non-zero eccentricities³ in a network. The two parameters can be formulated in terms of the eccentricity in the following way:

$$\text{Diameter} = \max(\epsilon_i) \quad (2.15)$$

$$\text{Radius} = \min(\epsilon_i) \quad (2.16)$$

³See subsection above for the definition of eccentricity.

A network's *characteristic pathlength* refers to the expected geodesic distance between two connected nodes. It is thus a measure of the average distance between nodes in the network. All the three parameters defined so far yields information about inter-node distances in a network.

Other interesting parameters are ones more directly related to network topology. The network density for example gives the fraction of edges present in the network compared to the maximum number of edges:

$$\text{Density} = \frac{1}{n(n-1)} \sum_i \sum_{j \neq i} a_{ij} \quad (2.17)$$

The density is thus a measure of the connectedness of the network. Another topologically motivated parameter is the network centralization [47]:

$$\text{Centralization} = \frac{n}{n-2} \left(\frac{\max(k)}{n-1} - \text{Density} \right) \approx \frac{\max(k)}{n} - \text{Density} \quad (2.18)$$

The network centralization is related to the connectivity distribution. For instance decentralized networks will have a centralization index close to 0, while networks with star-like topologies have an index close to 1.

Network heterogeneity [47] is measure of a network's tendency to contain hub nodes. It is defined as:

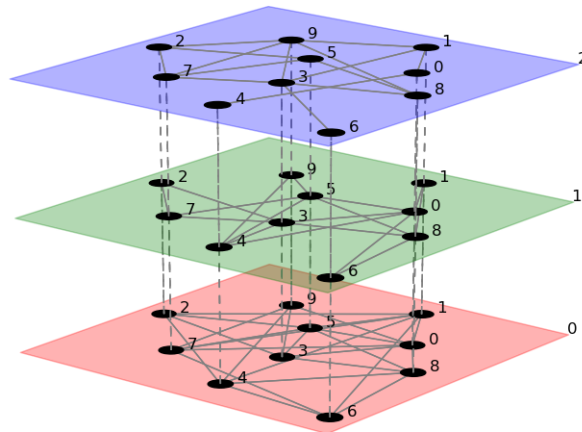
$$\text{Heterogeneity} = \frac{\text{variance}(k)}{\text{mean}(k)} = \sqrt{\frac{n \cdot S_2(k)}{S_1(k)^2} - 1} \quad (2.19)$$

The *network clustering coefficient* can be defined as the average clustering coefficient for all nodes in the network. The network clustering coefficient is thus a measure of the clustering tendency of the network.

2.1.8 Multilayer networks

In general, a network consists of a set of nodes (or vertices) V , and set of edges E . In a graph theoretical formulation the graph G is thus represented by $G = (V, E)$. In order to accommodate the existence of several types of node-relations, multiple dimensions (*layers*) can be added to the network [21]. The network graph is then represented by $G = (V, E, D)$, in which V is the set of nodes, D is the set of dimensions, and the edges E is represented by a tuple (u, v, d, w) . Here $u, v \in V$ are nodes, $d \in D$ is the dimension, and w is the weight of the edge ($w = 1$ for unweighted networks). It might be instructive to think of the multilayer network in terms of its adjacency matrix. The adjacency matrix of a multilayer network can be represented by a 3-dimensional matrix, in which dimension 1 and 2 constructs a regular adjacency matrix for the layer d encoded in the third matrix-dimension. An illustration of a multilayer network can be seen in figure 2.5.

Figure 2.5 Illustration of a multilayer network. Figure from [5].



2.2 Gene co-expression networks

Gene co-expression networks is an analytic tool for probing the genetic regulatory machinery, and its role in disease and cell differentiation. The fundamental assumption behind the study of gene co-expression networks (from here on simply referred to as co-expression networks), is that genes showing predictable co-expression patterns somehow relates to phenotypic and functional differences. Co-expression networks are generated by calculating correlation scores between all possible pair of genes from their measured expression levels. Co-expression networks are limited in that they only identify correlations, but do not provide information about causality or distinguish between regulated and regulatory genes.

Differential co-expression network analysis is a more specialized method, in which the analyst evaluates differences in expression patterns across appropriately chosen tissues, conditions or specimens. As gene expression and regulation can be highly tissue-specific, differential co-expression analysis is a suitable tool for detecting genes related to disease states [15, 16], tissue types [17] and developmental stages [18], because these genes are more likely to be regulators that underlie phenotypic differences [19].

2.2.1 Construction of gene co-expression networks

Many different methods for generating co-expression networks exist, utilizing a wide range of different methods. The general structure laid out in this section is by no means the only way of constructing co-expression networks. This section is rather meant as a guide for the unfamiliar reader to help him/her identify the general process of converting gene-

expression data into a network adjacency matrix. The section draws inspiration from the framework proposed by Zhang and Horvath in "A general framework for weighted gene co-expression network analysis" [20].

A gene co-expression network consists of a set of nodes, representing different genes, connected by edges, representing a similarity in expression patterns. Constructing such a network can be divided into two main steps:

1. Estimating co-expression through a similarity measure
2. Calculating the adjacency matrix from the similarity scores

For differential co-expression networks the following step is also needed:

3. Relating co-expression scores from the different data sets

The first step of co-expression network generation is to define a measure of similarity for comparing the gene expression profiles. Different correlation parameters are common examples of similarity measures, such as Pearson correlation [48], Spearman correlation [49] and biweight midcorrelation [50]. Other notable examples are mutual information [51] and regression-models [52]. For each pair of genes the similarity score s_{ij} between their expression patterns are calculated, resulting in the similarity matrix $S = S_{ij}$.

The next step is transforming the similarity matrix S_{ij} into the adjacency matrix using an *adjacency function*. A simple example of an adjacency function is using the Heaviside step function, to define a *hard* threshold:

$$a_{ij}(s_{ij}, \tau) = H(s_{ij} - \tau) = \begin{cases} 1 & s_{ij} > \tau \\ 0 & s_{ij} < \tau \end{cases} \quad (2.20)$$

Here a_{ij} is the argument for the adjacency matrix, s_{ij} is the corresponding similarity score, and τ is the threshold. The result of using a hard threshold is that all gene-pairs with a lower similarity score than τ are rejected. Zhang and Horvath [20] argues that a soft threshold might be more advantageous as "hard thresholding include loss of information and sensitivity to the choice of threshold." Also, "on a more fundamental level, the question is whether it is biologically meaningful to encode gene co-expression using binary information". Thus another approach is to use a smooth approximation of the Heaviside step function for obtaining a *soft* threshold [20]:

$$a_{ij}(s_{ij}, \tau_0, \alpha) = \frac{1}{1 + e^{-\alpha(s_{ij} - \tau_0)}} \quad (2.21)$$

In this case, both the soft threshold parameter τ_0 , and α which gives the sharpness of the transition, needs to be specified. Using a soft threshold yields a *weighted* gene co-expression network, while a hard threshold yields an *unweighted*.

2.2.2 CSD-framework for differential co-expression networks

The CSD-framework for generating gene co-expression networks has been developed by André Voigt and Eivind Almaas at the Department of Biotechnology and Food Science at NTNU⁴ and is described in the article "A composite network of conserved and tissue specific gene interactions reveals possible genetic interactions in glioma" [1]. The CSD-framework aims to conserve more of the information attainable from the co-expression measures than other existing methods by classifying differential co-expression into three categories; conserved (*C*), specific (*S*), and diverging (*D*). *Conserved* co-expression reflects a similar co-expression pattern in both the gene-expression data-sets. *Specific*, refers to cases in which one of the cases exhibits a "significant" level of co-expression between two genes, while the other does not. Finally *Diverging* co-expression corresponds to opposite co-expression (i.e. positive vs negative correlation) between two genes in the two data-sets. Figure 2.6 illustrates how the CSD-scores are related to the co-expression values.

Intra-cohort similarity

As the CSD-framework is a method for differential co-expression analysis, the method uses two or more sets of gene expression data originating from different tissues, species or conditions. The first step of the method is calculating the similarity scores for all pairs of genes within the each of the data-sets. The similarity scores are based on Spearman's rank correlation, where s_{ij} is determined by calculating the Spearman correlation over all measurement points at each gene. Thus, in a data-set consisting of expression data from 100 patients, the Spearman correlation is calculated between arrays with 100 data points for each gene.

Inter-cohort similarity

The next step in the CSD-method is quantifying the difference in co-expression patterns across the different cohorts. Three different "similarity"-scores are defined:

$$C_{ij} = \frac{|\rho_{ij,1} + \rho_{ij,2}|}{\sqrt{\sigma_{ij,1}^2 + \sigma_{ij,2}^2}} \quad (2.22a)$$

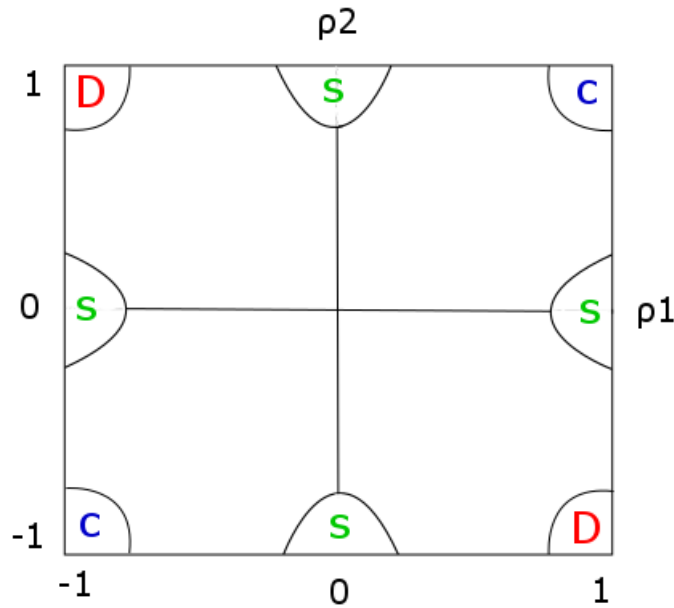
$$S_{ij} = \frac{||\rho_{ij,1}| - |\rho_{ij,2}||}{\sqrt{\sigma_{ij,1}^2 + \sigma_{ij,2}^2}} \quad (2.22b)$$

$$D_{ij} = \frac{|\rho_{ij,1}| + |\rho_{ij,2}| - |\rho_{ij,1} - \rho_{ij,2}|}{\sqrt{\sigma_{ij,1}^2 + \sigma_{ij,2}^2}} \quad (2.22c)$$

The equations are given for the case of two different data-sets, indicated by the subscripts 1 and 2. $\rho_{ij,1}$ represents s_{ij} for case 1. $\sigma_{ij,1}$ is the estimated variance in the

⁴Norwegian University of Science and Technology

Figure 2.6 Graphical representation of the C-, S-, and D-type co-expression regions of interest.



Spearman correlation parameters. The variance is estimated by drawing independent subsamples of size k from the m measurement points in a data-set. For each subsample the Spearman correlation is calculated, resulting in a distribution of correlation values for each subsample. From this distribution the corresponding variance is estimated. This form of variance estimation introduces some limitations on the data-sets that can be analyzed by the methods. In order to achieve sufficiently good variance estimates a minimal number datapoints is required in each subsample to get sufficient accuracy in the subsample correlations. Additionally a minimum number of values in each distribution is required to obtain a representative estimate of the variance. The current subsampling algorithm generates few subsamples when the number of data points is less than the square of the subsample size ($m \approx k^2$). A minimum of 49 data points ($m = 49$), and a subsample size of $k = 7$ is recommended in order to achieve good results using this method.

The argument for using a variance weighted comparison is that a distribution showing a high variance is related to a pair of genes less consistently co-expressed than a distribution with low variance. In other words; a gene-pair is less co-expressed the higher the variance is. Therefore, as a measure of differential co-expression, the CSD-scores need to take this variance into account.

The calculation of the CSD-scores results in three different interaction-scores for each gene-pair, effectively making the network a 3-dimensional multilayer network⁵.

⁵Multilayer networks are described in section 2.1.7.

Adjacency function

The final step of the method is calculating the adjacency matrix from the inter-cohort similarity matrix. The CSD-framework uses a hard threshold for the adjacency function (i.e a Heaviside step function). However, as the three interaction scores (network layers) have distributions with considerable differences in median, mean, variances, and shape in general [1], a single cut-off value τ across the network layers might lead to inconsistent significance requirements. Instead an empirical *importance threshold* X_p is determined independently from the underlying distributions of each of the network layers. From the M interaction scores calculated for the different gene pairs in each network layer, m samples s_i of size $L \ll M$ are drawn with replacement. The importance threshold is then defined as the average of the maximum values from the generated samples:

$$X_p = \frac{1}{m} \sum_{i=1}^m \max_{\{s_i\}} X \quad (2.23)$$

The importance level associated with the threshold values X_p^C , X_p^S , and X_p^D , is $p = 1/L$. In other words, the researcher picks the parameters p and m , from which the importance thresholds $X_p^{C,S,D}$ are determined. Using the formalism described in the previous subchapter, the Heaviside step function is then applied to each of the network layers, using $\tau = X_p$.

2.3 Microarray data

Microarrays is a set of high-throughput analytic devices used for probing biological information about an organism. A microarray generally consist of a solid substrate (glass slide or silicon thin-film cell), labeled with a 2D grid of probes that interact with various biological molecules. Typically the procedure of measuring the presence (and relative amount) of a certain molecule involves labeling the relevant molecules with fluorescent markers. The microarray is then scanned with a laser, and the observed fluorescent emission at a probe becomes a relative measure of the amount present of the corresponding biological molecule. The resulting data is usually normalized and corrected for background noise as a part of the procedure [53]. Different types of microarrays exist for probing different information. Examples are DNA microarrays, protein microarrays, peptide microarrays, and antibody arrays amongst others.

2.3.1 Transcriptome profiling

DNA microarrays are used for profiling of a cell's transcriptome⁶. The probes on a DNA microarray consist of oligonucleotides (short segments of DNA) which allow for hybridization with matching DNA-strands from the supplied specimen. In order to perform transcriptome profiling the mRNA from the specimen is isolated and copied into stable double stranded cDNA. The cDNA is then fragmented and labeled with a fluorescent dye. The cDNA fragments are added to the microarray where hybridization between

⁶The set of all RNA-molecules present in a cell.

probes and targets occur. Finally, the probes are illuminated, and the resulting fluorescent emission indicates the relative amount of the associated nucleotide sequence present in the transcriptome of the specimen.

2.4 Rheumatoid arthritis

Rheumatoid arthritis (RA) is a common autoimmune disease that primarily affects the joints. The effects are most prominent in the small joints of the hands and feet, where immune cells invade the synovial membrane around the joints. The result of RA is inflammation, hyperplasia, and progressive destruction of bone and cartilage. Other systemic features related to RA includes cardiovascular, pulmonary, psychological, and skeletal disorders, resulting in a higher mortality rate among RA-patients than for healthy individuals [54].

RA is a multifactorial disease arising from a combination of environmental and genetic factors. A twin study estimated that inherited genetic factors account for approximately 60% of the variation in disease liability [55, 56]. The the most highly associated gene, HLA-DRB1, (HLA = human leukocyte antigen) has been estimated to account for as much as 37% of the inherited liability [57] and as little as 18% for ACPA-positive (Anti-citrullinated protein antibodies) and 2.4% for ACPA-negative patients [56]. The fact that the most highly associated gene only show moderate association, illustrates the difficulty in identifying single genetic factors related to RA. Instead, a combination of several gene variants in combination with environmental factors causes the onset of the disease. A list of 52 genes associated with the disease state can be found in appendix A [58, 59, 60, 61, 62, 63, 64].

Non-genetic factors associated with RA include changes in sex hormones, occupational exposure to certain kinds of dusts and fibres, viral and bacterial infections, and long term smoking [65].

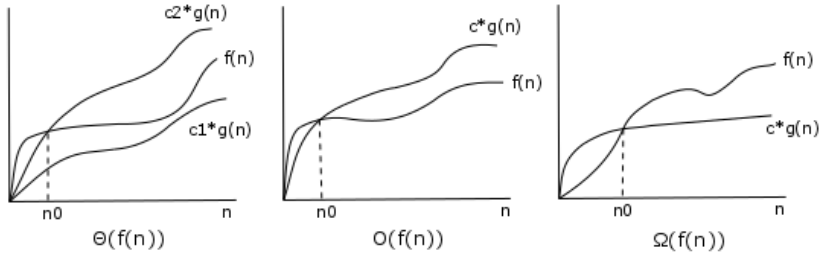
2.5 Computer science

2.5.1 Asymptotic time complexity

The *asymptotic time complexity* describes how the time consumption of a program or function depends on the size of the input data. The reason it is called the *asymptotic* time complexity is that it is expressed in terms of the dominating factor in the asymptotic limit of the input data size. For instance, a program might have a worst-case running time described by $an^3 + bn^2 + cn + d$, where n is the size of input data and the other parameters are constant factors. Such a function/program will have an asymptotic running time of $\Theta(n^3)$. For small input data the running time is not necessarily dominated by the third order contribution. If the constant b is very high for instance, the running time will be dominated by the second order contribution. However, for some value $n = n_0$, the third order contribution will start to dominate the running time, and will do so for all $n > n_0$. The time complexity notation thus describes the asymptotic time complexity of a function.

Three different time-complexity notations are used; $\Theta(f(n))$, $O(f(n))$, and $\Omega(f(n))$.

Figure 2.7 Graphical representation of the definitions of the asymptotic time-complexity parameters $\Theta(f(n))$, $O(f(n))$, and $\Omega(f(n))$.



Formally they are defined as follows [66]:

$\Theta(f(n))$ means that there exist positive constants c_1 , c_2 and n_0 such that:
 $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$ for all $n > n_0$.

$O(f(n))$ means that there exist positive constants c and n_0 such that:
 $0 < f(n) \leq c g(n)$ for all $n > n_0$.

$\Omega(f(n))$ means that there exist positive constants c and n_0 such that:
 $0 < c g(n) \leq f(n)$ for all $n > n_0$.

Figure 2.7 shows a graphical representation of the above definitions.

2.5.2 Parallel programming

In computer science a thread of execution is a sequence of instructions that can be performed independently by a scheduler. A regular serial computer program executes on one thread, where all computations involved are performed in sequence by the same thread. Parallel computing is the process of structuring a program to accommodate parallel execution of operations on multiple threads simultaneously. This opens up for significant reduction in the time spent by a computer executing a program. This is of course provided that the system contains hardware that supports parallel execution.

In the context of program time consumption it is important to distinguish between two different concepts of time; *CPU-time* and *wall-time*. CPU-time is the number of operations required to execute the program, divided by the number of operations performed per second by a CPU. Wall-time on the other hand, refers to the actual time it takes for the program to execute. In other words, the wall-time is the time from you start the program until the program terminates. The CPU-time is thus a measure of the amount of work required

to execute the code, and is independent of the number of cores executing the program⁷, while the wall-time is the time it takes to execute the code.

2.6 Correlation and similarity measures

The following sections provide the definitions of different similarity measures used in co-expression networks. Although Spearman correlation is used throughout this thesis, a brief discussion on the choice of similarity measure is included in the discussion chapter. The relevant methods are therefore described here as a reference.

2.6.1 Pearson correlation

Pearson's correlation coefficient is a measure of linear correlation between two variables. The formula for the Pearson correlation coefficient is as follows:

$$\rho_{X,Y} = \frac{\text{cov}(X, Y)}{\sigma_X \sigma_Y} \quad (2.24)$$

Here $\text{cov}(X, Y)$ is the co-variance between the variables X and Y , and σ gives their standard deviations. The Pearson correlation coefficient thus assumes a value between -1 and 1, where $\rho_{X,Y} = 1$ implies perfect positive linear correlation and $\rho_{X,Y} = -1$ perfect negative correlation between X and Y .

2.6.2 Spearman correlation

The Spearman correlation coefficient is a measure of the monotonic relationship between two variables. The formula for calculating the Spearman correlation coefficient is the same as for Pearson's correlation coefficient. However, while the Pearson correlation is calculated based on the actual values assumed by the variables X and Y , Spearman correlation uses their relative ranks.

$$\rho_{r_X, r_Y} = \frac{\text{cov}(r_X, r_Y)}{\sigma_{r_X} \sigma_{r_Y}} \quad (2.25)$$

Here r_X and r_Y refers to the ranked values of the parameters X and Y . Perfect Spearman correlation ($\rho_{r_X, r_Y} = 1$) indicates a perfect monotonic relationship, linear or non-linear.

2.6.3 Kendall's tau

Just as the Spearman correlation, Kendall's tau is a statistic used to measure ordinal association between two random variables. Kendall's [67] tau is based on comparison of pairs of measurements from the two variables. For two variables X and Y , (x_i, y_i) and (x_j, y_j) are compared. If $x_i > y_i$ and $x_j > y_j$, or the other way around, the pair is said to be *concordant*. If this is not the case, and $x_i < y_i$ for $x_j > y_j$ for instance, the pair is called

⁷Technically the amount of work required increases with the number of cores due to increased overhead.

discordant. By determining the number of concordant pairs n_c and discordant pairs n_d , Kendall's tau is given by:

$$\tau = \frac{n_c - n_d}{\binom{n(n-1)}{2}} \quad (2.26)$$

with the denominator representing the number of different ways of combining the n measurements of the variables X and Y . A problem in both Spearman correlation and Kendall's tau is dealing with identical measurements of a variable. For variables containing many such ties ($x_i = x_j$ or $y_i = y_j$), alternate versions of Kendall's tau exist that compensates for this. An example of this is the τ_B statistic:

$$\tau_B = \frac{n_c - n_d}{\sqrt{(n_c + n_d + n_{t,X})(n_c + n_d + n_{t,Y})}} \quad (2.27)$$

Here $n_{t,X}$ and $n_{t,Y}$ is the number of ties in X and Y respectively. This modification effectively normalizes the τ -scores to the interval $\tau \in [-1, 1]$ even in the presence of large amounts of tied variable values.

2.6.4 Biweight midcorrelation

Biweight midcorrelation is a median-based similarity measure. The fact that it is median based makes it more robust to outliers than Pearson correlation, for example. It has therefore been suggested as a more robust alternative for evaluating co-expression patterns in genes [51].

The biweight midcorrelation incorporates weights w_i for each value assumed by the variables X and Y , in order to remove outliers, and assigns more weight to values close to the medians. The weights are calculated from median based parameters defined as follows:

$$u_i = \frac{x_i - \text{med}(x)}{9 \cdot \text{mad}(x)} \quad (2.28a)$$

$$v_i = \frac{y_i - \text{med}(y)}{9 \cdot \text{mad}(y)} \quad (2.28b)$$

Here $\text{med}(x)$ is the median of x , and $\text{mad}(x)$ is its median absolute deviation ($\text{mad}(x) = \text{med}(|X_i - \text{med}(X)|)$). The weights are then defined as;

$$w_i^{(x)} = (1 - u_i^2)^2 \cdot I(1 - |u_i|) \quad (2.29a)$$

$$w_i^{(y)} = (1 - v_i^2)^2 \cdot I(1 - |v_i|) \quad (2.29b)$$

where,

$$I(x) = \begin{cases} 1 & x > 0 \\ 0 & \text{otherwise} \end{cases} \quad (2.30)$$

The deviation from the median is calculated and normalized,

$$\tilde{x}_i = \frac{(x_i - \text{med}(x)) \cdot w_i^{(x)}}{\sqrt{\sum_{j=1}^m [(x_j - \text{med}(x)) \cdot w_j^{(x)}]^2}} \quad (2.31a)$$

$$\tilde{y}_i = \frac{(y_i - \text{med}(y)) \cdot w_i^{(y)}}{\sqrt{\sum_{j=1}^m [(y_j - \text{med}(y)) \cdot w_j^{(y)}]^2}} \quad (2.31b)$$

and the biweight midcorrelation is given by:

$$\text{bicorr}(x, y) = \sum_{i=1}^m \tilde{x}_i \cdot \tilde{y}_i \quad (2.32)$$

2.6.5 Mutual information

Mutual information is a measure of variable dependency. The mutual information (MI) between variable X and variable Y gives the amount of information (in units of bits or shannons) that is obtained about the variable X through variable Y or vice versa. Mutual information is defined as follows [68]:

$$I(X, Y) = \sum_{x \in X} \sum_{y \in Y} p(x, y) \log \left(\frac{p(x, y)}{p(x)p(y)} \right) \quad (2.33)$$

Here $p(x, y)$ is the joint probability distribution of X and Y , and $p(x)$ and $p(y)$ the corresponding marginal distributions. As mutual information is defined in terms of discrete variables, binning procedures for the estimation of the discrete probability distributions is needed for application on continuous data [69].

Mutual information has the advantage that it does not make any assumptions about how two variables are related, it simply measures their dependency. A measure of zero mutual information implies statistical independence between the variables, provided the estimated probability distributions are correct.

2.7 Statistics

2.7.1 Fisher exact test

Fisher's exact test [70] is a test used for evaluating dependencies in nominal variables. Consider for instance a case of a 2×2 contingency table, where the first variable represents the gender, and the second variable whether or not a person is a student. The test assumes constant margins sums in the table.

Table 2.1: Example of contingency table for two nominal variables.

	Male	Female	Total:
Student	a	b	$a + b$
Not student	c	d	$c + d$
Total:	$a + c$	$b + d$	$n = a + b + c + d$

Fisher's exact test is based on a null hypothesis of independence between the variables. The hypergeometric distribution is then used to calculate the probability of getting the observed data or more extreme observations. The resulting probability is the p-value of the test. For the two-tailed test, the probability of getting equally extreme observations but in the opposite direction is also added. The probability of a given observation is calculated from the hypergeometric probability function:

$$p = \frac{\binom{a+b}{a} \binom{c+d}{c}}{\binom{n}{a+c}} = \frac{(a+b)!(c+d)!(a+c)!(b+d)!}{a!b!c!d!n!} \quad (2.34)$$

A multivariate generalization of the above function can be expressed in terms of the row sums R_i , columns sums C_j and matrix elements m_{ij} as follows:

$$p = \frac{\Pi_i(R_i!) \cdot \Pi_j(C_j!)}{n! \cdot \Pi_{i,j}(m_{ij}!)} \quad (2.35)$$

For the estimation of the p-value of this test, the tables must be ordered by some criteria to determine which observations are considered further away from independence.

Fisher's exact test is mostly used when the number of observations is small, due to the number of computations that needs to be performed. For cases with many observations, or approximately equal number of observations in each category (resulting in many computations), the *chi-squared test of independence* can be used as a good approximation.

2.7.2 Bonferroni correction

Bonferroni correction [71] is a simple and conservative method for compensating for multiple testing in statistics. Let H_1, \dots, H_m be a set of hypothesis with a corresponding set of p-values p_1, \dots, p_m . Here m is the total number of null hypotheses. Given m_0 true null hypotheses, the Bonferroni correction limits the *familywise error rate* (FWER)⁸ to less than or equal to α by requiring that $p_i \leq \frac{\alpha}{m}$:

$$FWER = P\left\{\bigcup_{i=1}^{m_0} \left(p_i \leq \frac{\alpha}{m}\right)\right\} \leq \sum_{i=1}^{m_0} \left\{P\left(p_i \leq \frac{\alpha}{m}\right)\right\} = m_0 \frac{\alpha}{m} \leq m \frac{\alpha}{m} = \alpha \quad (2.36)$$

⁸The probability of making a false discovery (type 1 error)

2.7.3 Benjamini-Hockberg method

The *Benjamini-Hockberg method* [72] for compensating for multiple testing is based on limiting the *false discovery rate* (FDR) at a level α . The false discovery rate is defined as the expected number of false discoveries among the discoveries. For a set of hypotheses H_1, \dots, H_m with corresponding p-values $P_{(1)}, \dots, P_{(m)}$, sorted in ascending order of p-values, the Benjamini-Hockberg procedure is done the following way:

1. For a given α find the largest value k such that $P_{(k)} \leq \frac{k}{m} \alpha$
2. Reject all null hypothesis $H_{(i)}$ for $i = 1, 2, \dots, k$

The Benjamini-Hockberg method is valid for independent tests, but also for some cases of dependency [73]. A modification to the method exists that compensates for dependence [73]. The concept of controlling the false discovery rate, provides less stringent compensation of type 1 error than controlling family wise error rate such as the Bonferroni correction method. The increased power of the FDR-based methods comes at the cost of increased probability of type 1 errors.

Chapter 3

Methods: Software

The main goal of this thesis was the development of a program for fast and easy generation of differential co-expression networks using the CSD-framework. The result is the program *CSD-CS*, short for *CSD-C++ Software*. The entire software has been developed as a part of this thesis, and so everything described in this chapter refers to work performed by the author. The software can be found on [github](#)¹, along with all related files. As the name indicates, the program was implemented using the C++ programming language. It comes with a GNU make file as the software was written and developed on Ubuntu 16.04. Due to the computationally expensive nature of the CSD-method, a simple parallelization scheme of the sections with highest temporal complexity was performed using OpenMP. It should be noted that OpenMP is for shared memory systems only. The software have been thoroughly tested during development, and appears to behave according to its specifications. Potential users should however make themselves acquainted with the program's limitations and requirements before using it. The networks generated by the program are written to file in a format compatible with the *Cytoscape* [74] software environment. This makes it easy to both visualize and analyze the network using the *network analyzer* [75] package.

The *CSD-CS* project is open source and comes with an MIT-License. Some effort has been made to facilitate further development, and suggestions for possible projects for extended functionality and alternate solutions can be found in the discussion chapter. The sections below contains a thorough description of the program. Hopefully this chapter will make it easier for others to read and understand the code, and encourage further development.

¹www.github.com/magnusolavhelland/CSD-Software

3.1 Program overview

Following from the CSD-method, the CSD-software consists of three main processes; calculating similarity scores within each dataset, determining CSD-scores, and determining the adjacency matrices using the importance thresholds. The CSD-Software was developed in a modular fashion, where each of the processes above behaves as an independent subprogram to the user. Any sequential combination of these subprograms are valid executions of the program. The operation of the program is specified through the combination of input parameters supplied by the user². Which of the processes is executed depends on the supplied input data and the requested output data. The program can only take one type of input-data, but the user can request multiple output-files containing the results from each of the "subprograms" executed.

The full program, using expression data as input and returning a CSD-network, performs the following steps:

1. Preprocessing of input data
2. Reading from files
3. Calculating correlation coefficients
4. Generating subsamples and estimating variances
5. Calculating CSD-scores
6. Determining importance thresholds for C-,S- and D-type interactions
7. Calculating adjacency matrices
8. Writing network to file

The first step of the software involves preprocessing of the input data. The purpose of this step is to enable more flexible use of the program. The preprocessing algorithm removes all genes not present in all input-files, and eliminates the need for identically sorted gene-sequences in the input-files. As the three processes operates on different data-formats, so does the program-modules. Although, the current version of the program writes all output-data in a format the next module can read, only the first software module contains a preprocessing function. This limits independent use of the other modules, as the program can only use two sets of correlation data (or CSD-scores), provided that they were analyzed together by module 1. The result is a somewhat limited modular functionality in the current version of the software.

3.2 Setting up and using the program

After downloading *CSD-CS* repository from github and putting it in an appropriate folder, the only remaining step before using the program is compiling it. The program was compiled using the g++ compiler (version 4.5.0). In the linux terminal, the program can be

²See section on "Program parameters".

Figure 3.1 The figure illustrates the program modularity. Each module represents one of the one of the main processes of the program, and can be performed independently of each other.

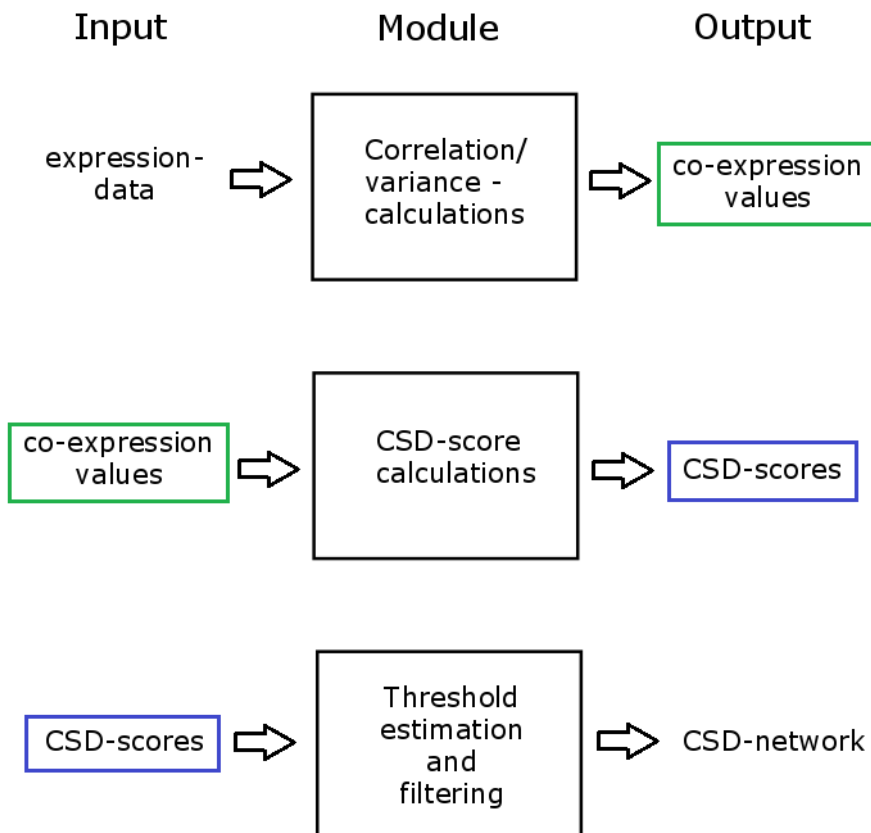
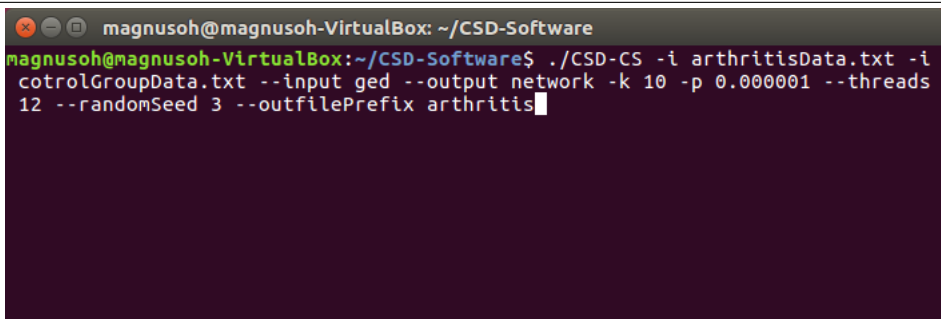


Figure 3.2 The figure illustrates how the program can be executed using the linux terminal. Notice how all parameters are preceded by an appropriate identifier.



```
magnusoh@magnusoh-VirtualBox: ~/CSD-Software
magnusoh@magnusoh-VirtualBox:~/CSD-Software$ ./CSD-CS -i arthritisData.txt -i
cotrolGroupData.txt --input ged --output network -k 10 -p 0.000001 --threads
12 --randomSeed 3 --outfilePrefix arthritis
```

compiled by moving to the directory where the program is located and enter the command *make*. This should create an executable called *CSD-CS*. There is currently no make file for Windows operating systems.

The operation of the program is specified by the parameters described in the sections below. The program searches for all input files in the same folder as the executable is located in. The program also places all output files, including a file containing mismatched genes from the input files, in this same folder. Some of the output files have generic default names, such as *CSD_Network.txt* for the filtered CSD-network. For this reason the user should make sure to remove or copy any previous results to a different folder before running the program, or the files will be overwritten.

3.3 Program parameters

The *CSD-CS* program have a total of 13 input parameters that can be specified by the user. A parameter is set upon calling the executable in the terminal, by following it by an *indicator* and then a suitable argument. All elements should be separated by a space, and all arguments should be preceded by an indicator. Table 3.1 shows an overview of the indicators and arguments used by the program. The program verifies all keywords before assigning them to their parameter, and returns an error if the argument does not match one of the legal keywords for the parameter. If invalid file names are passed, the program will terminate when trying to read the file and return an error. The program also interprets the combination of parameters supplied by the user and validates its logic. For instance, attempting to return correlation-data when using CSD-scores as input will cause the program to terminate before execution and return an error. Assigning meaningful values to the numerical parameters is left for the user.

3.3.1 Input file

Input files are added to the program by passing the *-i* indicator followed by the file name when calling the program. The program can take an arbitrary number of input files. It is however required that all the input files are of the same type. When receiving more than

Table 3.1: Overview of the indicators, and arguments used by *CSD-CS*. Upon calling the program in the terminal, it should be followed by some combination of these arguments to specify the execution of the program. Each argument should be preceded by the appropriate indicator to specify which parameter it sets. The arguments given in *italic* are keywords specific for the indicator. Attempting to set one of these parameters using anything other than a keyword will return an error.

Indicator	Parameter	Argument(-s)
-i	input file	filename
-k	subsample size	integer
-p	importance level	float
- -input	type of input data	<i>ged</i> <i>corr</i> <i>csd</i>
- -output	type of output data	<i>corr</i> <i>csd</i> <i>network</i> <i>all</i>
- -corrMethod	measure of coexpression	<i>spearman</i>
- -format	orientation of gene expression data in input file	<i>vertical</i> <i>horizontal</i>
-maxSubsamples	maximum number of subsamples to be generated by the subsampling algorithm	integer
-terminationLimit	the maximum number of failed attempts at generating a subsample	integer
-randomSeed	seed for random number generators in program	integer
-iSamples	number of samples drawn in importance threshold estimation	integer
-threads	number of parallel threads spawned by the program	integer
-outfilePrefix	prefix for names of output files	string

two input files, the program generates one network for each possible combination of the datasets. The memory and time consumption of the program when receiving more than two files has not been verified, but both are assumed to scale linearly with the number of ways of combining the x input files ($\frac{x(x-1)}{2}$).

3.3.2 Subsample size

The subsample size is set by through the $-k$ indicator. The argument must be an integer in the interval $\langle 0, m \rangle$ where m is the number of data points per gene in the gene-expression file with the lowest number of data points. As explained in the theory chapter the subsample size should be at least 7 and ideally $k \leq \sqrt{m}$, thus the input files should contain at least 49 data points per gene.

3.3.3 Importance level

The importance level of the method can be set by using the $-p$ indicator. The argument should be a decimal number in the interval $\langle 0, 1 \rangle$.

3.3.4 Input data and file format

The input data parameter is used to specify the data type in the input files. There are three possible keywords that can be assigned to this parameter; *ged* (gene-expression data), *corr* (correlation and variation data), or *csd* (all CSD-scores). The format of the input data used by the program, is different for each case. The two latter cases have some resemblance in that they give a sorted list of all possible gene pairs, along with the correlation and variance, or CSD-scores respectively. For the case of gene-expression data, the input file should contain a tab-separated two dimensional matrix, where the first row and column should contain the gene-IDs and measurement-IDs. In the case of *ged*-input, the format parameter can be used to specify along which axis the genes are oriented in the matrix. Two possible arguments exist for the format parameter; *vertical*, and *horizontal*. *Vertical* means that each row contains all the measurements of a single gene, while *horizontal* means that the measurements of a gene are found along a column.

For examples of the data-formats see appendix B.

3.3.5 Output data

The output data parameter is used to specify which types of output the program should return. The output data parameter is different from the other parameters in that it can take multiple assignments, thus allowing the user to request multiple outputs. The keywords for this parameter are: *corr*, *csd*, *network* and *all*. The *corr* keywords requests correlation and variance data to be written to a file. *csd* tells the program to return the CSD-scores before thresholding, while *network* returns the list of gene-pairs in CSD-network after importance-filtering. Using the *all*-keyword writes correlation, variance and CSD-scores

for all gene-pairs to a single file. Note that requesting *corr*, *csd*, or *all* for datasets containing a large number of genes results in large output files.³

3.3.6 Correlation method

The correlation method parameter are used to specify which type of correlation function that should be used as a similarity measure for co-expression. Currently only Spearman correlation is supported, but a discussion on other correlation measures relevant for future versions of *CSD-CS* is supplied in chapter 7.

3.3.7 Subsampling parameters

Two different subsampling schemes is used by the software⁴. The first subsampling scheme is used for estimating correlation variances, and the second for determining the importance thresholds. The subsampling scheme used for variance estimation uses the parameters *maxSubsamples*, *terminationLimit* and *randomSeed*. *maxSubsamples* sets an upper limit for the number of subsamples that can be generated. This is useful in cases where two datasets have a large difference in the number of data points. For such cases, the dataset with few measurement points will limit the subsample size, and cause a large number of subsamples to be generated for the bigger dataset. This in turn increases the running time of the program. By introducing a maximum number of subsamples, the subsampling algorithm terminates when a sufficient number of subsamples have been generated.

As the first subsampling function is random based, a termination criterion for cases in which the maximum number of subsamples is not reached is also required. This is provided through the *terminationLimit*-parameter. When the subsampling algorithm has unsuccessfully attempted to draw a subsample as many times as specified by the *terminationLimit*, the algorithm terminates.

Finally, the *randomSeed* is used by the random number generator to generate reproducible results.

The second subsampling function is used for threshold estimation. Similar to the first subsampling function, it uses a random number generator in the drawing procedure. The *randomSeed* is thus used by this function as well. In contrast to the first subsampling algorithm, the second one draws the subsamples with replacement. In other words; after successfully generating a subsample, the drawn data-points can be re-drawn in later subsamples. Thus, as long as the number of values drawn in each subsample is less then the total number of values in the underlying distribution, the algorithm can generate an arbitrary number of subsamples. The parameter *iSamples* is therefore used to specify the number of subsamples drawn before termination.

3.3.8 Threads

The program is parallelized for shared memory systems in order to reduce the wall-time. This feature is used through the *threads*-parameter. The program spawns the number of

³The *all*-file generated for the application in chapter 5 had a size of 9.4 GB. In this case the total number of genes were 18490.

⁴See section 3.5 for description of the algorithms.

Parameter	Default value
Subsample size	7
Importance level	0.001
Input data type	<i>ged</i>
Correlation measure	<i>spearman</i>
Data format	<i>vertical</i>
Max subsamples	100
Termination limit	1000
Importance samples	10000
Outfile prefix	"CSD"
Threads	1

Table 3.2: The table shows the default values of the input parameters of the *CSD-CS*-program.

threads requested through the *threads*-parameter, and as long as the number of threads does not exceed the number of cores, all threads are executed in parallel by relevant program sections.

3.3.9 The *outfilePrefix* parameter

As mentioned earlier in this chapter, some of the output files have general names as default. In order to avoid overwriting older files, a prefix for the output files can be specified by the user. For instance, if the datasets analyzed is related to breast cancer, the prefix "*breast_cancer*" can be supplied, for which the output files will be called "*breast_cancer_corr.txt*", "*breast_cancer_csd.txt*", "*breast_cancer_network.txt*" and "*breast_cancer_all.txt*", for the four different output types.

3.3.10 Default values and unnecessary parameters

All parameters have default values except for the output parameter, input file-list and random seed. The default values have been chosen to accommodate a minimalist case with respect to number of data points. The default values are listed in table 3.2.

Although all parameters will have some assigned value when the program is executed, default or user specified, they will not necessarily have an impact. For instance if the user feeds the program gene-expression data and requests correlation-data as the only output, the program will terminate after writing the requested data to file. This means that the importance level parameter will not be used by the program regardless of whether the user specifies it or not. The program will not return any warnings or errors in such a case as it does not affect the results in any way.

3.4 Data formats

Examples of the data formats used by the program can be found in appendix B. The program assumes that all input files matches the specified format, and is sensitive to variations. For example, when using gene expression data as input, the program does not

tolerate empty space at the end of a line as this will cause it to overestimate the number of columns. The program is then likely to crash, or worse; it will use uninitialized values in its calculations. Empty space should also be avoided at the bottom of the file, although it has not been registered to have any consequences.

Currently the program only performs any preprocessing⁵ in the case of gene expression data as input. The preprocessing algorithm filters out all gene-data that does not occur in all input files. The software does not have an equivalent functionality for correlation-data or CSD-scores as input, and so the flexibility in using these sections separately is restricted. The user is then required to remove any gene-pairs that does not occur in all files himself/herself, in order for the similarity and adjacency matrices to have the same dimensions. It is also important that the gene-pairs are sorted for these data-types, as this required by the file-reading algorithm.

3.5 Algorithms

3.5.1 Preprocessing gene-IDs

As a measure to increase the flexibility of the program in dealing with differences in data from different sources, a preprocessing algorithm was implemented during this work. The purpose of this algorithm is to remove the genes that do not occur in all input files. At the same time the algorithm sorts the remaining genes by their identifiers, allowing the subsequent function, that reads the data from each of the input files, to store the resulting data-matrices in the same order. This is useful as it makes it easier to keep track of the genes and their combinations later on in the program, and ensures the similarity and adjacency matrices contain the same gene pairs at each position $[i, j]$.

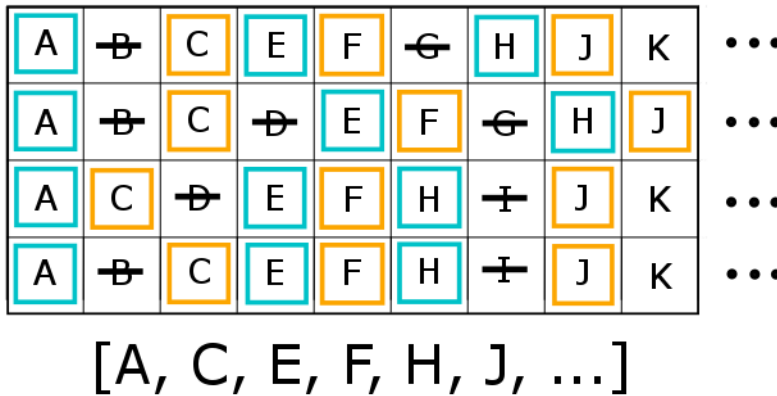
The algorithm simply starts out by reading the gene-IDs from each input file into separate C++-vectors. All of them are then sorted using the quick-sort algorithm [66] by performing character-by-character comparisons between the strings, and giving short strings precedence before long strings. The vectors are then iterated through looking for matching elements, and the intersection (i.e. the conjunction) of the gene-ID vectors is returned. The algorithm is estimated to have an asymptotic running time of $O(m \cdot n_{max} \cdot \log(n_{max}))$. Here m is the number of input files, and n_{max} is the largest number of genes in any of the input files. The asymptotic running time is caused by the quick-sort algorithm running in $O(n \log(n))$. The intersection part of the algorithm will run in $O(n_1 + n_2 + \dots + n_m) \approx O(m \times n_{avg})$, which is negligible in the asymptotic limit. Figure 3.3 shows the general idea behind the filtering algorithm. All gene-IDs not found in the intersection vector, are written to a separate file by the program. The file is called "mismatchedGenes.txt" and is put in the program's directory.

3.5.2 Correlation calculations

The correlation calculations uses the Spearman correlation measure. Thus it consists of two steps; ranking the measurements for each gene, and calculating the correlation coefficient for each different gene-pair. The *merge-sort* algorithm [66] was used for ranking the

⁵Algorithm described in the next section

Figure 3.3 Illustration of the idea behind the preprocessing algorithm, for the case of four input files. Each row is a sorted gene-list from one of the input-files. The gene lists are iterated over incrementally, and all genes not present in all data-sets are ignored. By sorting the gene-lists first, the algorithm only needs to iterate over all elements once, allowing the comparison to run in linear time. The color is added to highlight the successful comparisons and make them easier to separate from one another.



data. Although both quick-sort and merge-sort have expected asymptotic time complexities of $\Theta(m \cdot \log(m))$, merge-sort has a larger constant factor causing quick-sort to perform better on average. Merge-sort is however a stable sorting algorithm which was considered important for consistent ranking, in particular for cases with multiple identical measurement values. Merge-sort was therefore chosen over quick-sort for this purpose. For n genes and m measurements, this first step results in a time complexity of $\Theta(n \cdot m \cdot \log(m))$.

The second step is calculating the correlation parameters. The formula is the same as for Pearson correlation, and can be performed in linear time ($\Theta(m)$). It does however need to be repeated for all possible combinations of the n genes ($n(n-1)/2$) resulting in a time complexity of $\Theta(m \cdot n^2)$ for the second step.

For the majority of cases, the number of genes n will be much bigger than the number of measurements m . Therefore the total time-complexity of this algorithm is $\approx \Theta(n^2)$, but with a large constant factor.

3.5.3 Subsampling algorithms for variance estimation

The CSD-method estimates the variance in the correlation measures by subsampling the measurement points and calculating the correlations for each of the subsamples. From the generated distributions the variance is then calculated. The problem is therefore to generate a sufficient number of subsamples to get enough points in each distribution to get a good estimate of the variance. At the same time, the number of subsamples should not be too high, as this makes the method computationally expensive. The current solution is to generate independent subsamples, so that no two measurements occur in the same

subsample twice. In the article describing the method [1], the following algorithm for generation of independent subsamples was described:

1. Enumerate the objects in the set from which the sub-samples will be drawn.
2. Divide the total set of objects into groups of size k by their assigned values, thus generating a set of initial subsamples.
(Ex: $k = 4$ will yield $\{(1, 2, 3, 4), (5, 6, 7, 8), (9, 10, 11, 12), \dots\}$)
3. Now, starting at the object with the lowest index $i = 1$, sequentially iterate through the objects and add to the current sub-sample any objects that have not previously co-occurred in a sub-sample with any of the selected objects. Continue this process until the sub-sample is of size k , or there are no more valid objects.
4. Repeat the step above until there are no more possible sub-samples including i .
5. Repeat steps 3 and 4, increasing i , until there are no more possible sub-samples to be drawn.

During program and algorithm testing it was discovered that this structured method of drawing subsamples did not generate an optimal amount of subsamples. Although the algorithm is simple and predictable, it tended to draw the subsamples so that some of the measurement points were left with many unused combinations. The reason seemed to be that this form of systematic drawing favoured some of the data points over others. For instance data points with low indices are favoured over high-index data points, as the low-index ones have their domains exhausted earlier, and are thus less restricted by the domains of the other variables. There also seemed to be some regular interval at which unfavoured variables would occur, seemingly related to the grouping performed in step 2 of the algorithm above. For this reason three different approaches to independent subsampling were attempted. In addition to the algorithm above, a similar method omitting step number 2 was implemented, and a random based subsampling algorithm. The current version of *CSD-CS* uses the random subsampling algorithm described below, as it turned out to perform better than the sequential algorithms (see section 4.1).

Implementation of sequential algorithms

The sequential subsampling algorithms was implemented using the backtracking-search algorithm [76]. It is a depth-first search algorithm, that systematically traverses a search tree looking for a solution. The search-tree traversed by the algorithm in this case has a maximum depth of k and an initial branching factor of $(m - 1 - d)$, where k is the subsample size, m is the number of measurements, and d is the depth in the search-tree. Finding a solution (generating a valid subsample) consist in arriving at depth $d = k$ in the search-tree. Upon finding a valid solution, the subsample is stored and the search-tree modified, removing all paths conflicting with the generated subsample (in terms of the independence requirement). This guarantees that all consecutive subsamples will be independent. The backtracking search algorithm is guaranteed to find a valid solution given its existence, and so when it fails to generate a new subsample, the subsampling procedure terminates. The backtracking-search algorithm was instructed to always evaluate the measurements

points with lowest index first. The only difference between the two sequential algorithms is whether or not step 2 in the algorithm described above is performed before initiating the backtracking search algorithm.

The worst-case time complexity of the algorithms is $O(k \cdot s \cdot m^2)$, where k is the subsample size, m is the number of measurements at each gene, and s is the number of subsamples generated. The memory complexity of the algorithm is $O(m^2)$.

Random subsampling algorithm

The random subsampling algorithm works much the same way as the incremental one. The main difference is that instead of using the backtracking search-algorithm, it simply chooses a path randomly down the search-tree. If the algorithm reaches a dead end (a path shorter than k), it starts from the top again.

Two different termination criteria are used by the function. The first one (set by *terminationLimit* program parameter) causes the algorithm to terminate once a certain number of unsuccessful paths has been explored. The algorithm has no memory of the paths traversed, and so there is a possibility that a dead-end path may be attempted several times, although this is unlikely to be a big problem. The second termination criterion (the one set through the *maxSubsamples* parameter) causes the algorithm to terminate when a certain number of successful paths have been explored.

The random subsampling algorithm runs in $O(k \cdot m \cdot (s_{max} + f_{max}))$ where k is the subsample size, m is the number of measurements and s_{max} and k_{max} refers to the two termination limits above. The memory complexity is $O(m^2)$.

3.5.4 Variance estimation

After generating the subsamples using the algorithm above, the Spearman correlation parameter is calculated for each of the subsamples for all possible gene-pairs. Thus, for each pair of genes, s (the number of subsamples) correlation parameters are calculated. The variance in the correlation between the relevant gene-pair is then calculated from the s subsample-correlations. As the variance estimation runs in linear time ($\Theta(s)$), and the Spearman correlation in $\Theta(k \cdot \log(k))$ (due to the merge sort algorithm), the total time complexity of this procedure is $\Theta((k \cdot \log(k) + 1) \cdot s \cdot n^2)$, as the Spearman correlation must be calculated for each of the s subsamples and each of the $\propto n^2$ gene pairs, while the variance only needs to be calculated for each gene-pair.

3.5.5 Calculating CSD-scores

The CSD-scores are calculated from the correlation and variance using equations 2.22. Assuming that the calculation of a square-root is performed in constant time, the time-complexity of calculating CSD-scores is proportional to the number of gene-pairs ($\propto n^2$). The calculation of the CSD-scores thus runs in $\Theta(n^2)$.

3.5.6 Estimation of importance thresholds

As explained in the previous chapter (section 2.2.2) the thresholds are estimated as the average maximum C-, S- and D-scores drawn in a set of subsamples of size $1/p$, where p is the selected importance level. Each of the subsamples are drawn with replacement. The current implementation of this subsampling scheme does not keep track of which elements have been drawn, thus it does not guarantee that all elements in a subsample are separate elements. In other words, it is possible to draw the same value twice in a subsample. For the example analysis performed in this thesis, a datasets of approximately 20 000 genes were used, resulting in approximately $2 \cdot 10^8$ C-, S-, and D-scores. An importance level of $p = 10^{-5}$ requires the subsamples to contain 10^5 elements each. This means that each subsample only contains 0.05% of the total number of elements. Although it is likely that some elements will be drawn multiple times, the effect of this is assumed negligible as long as $1/p \ll n(n-1)/2$. The advantage of implementing the method this way is increased speed and reduced memory consumption, as the algorithm does not need to keep track of the elements drawn.

The time complexity of this algorithm is $\Theta(s_p \cdot \frac{1}{p})$, where s_p is the number of subsamples generated by the algorithm set through *iSamples*.

3.6 Parallelization

The human genome contains about 20 000-25 000 protein coding genes [77], and so the software must accommodate data sets of this size. As the previous subsection illustrated, several of the processes runs in $\Theta(n^2)$. Most notably, the correlation and variance estimations has a time complexity proportional to n^2 with high constant factors. With $n = 2 \cdot 10^4$, the computational complexity is proportional to $4 \cdot 10^8$ operations, resulting in high wall-times for the program. As a measure to reduce time-consumption parallelization of the most computationally expensive processes was performed using *OpenMP*⁶ [80, 81].

Parallelization of both the correlation and variance calculations was performed by parallelizing the outer *for*-loop in the algorithms. The parallelization uses a cyclic division of the loop-indices over the threads. This means that if the indices 1 to 16 were to be divided over 4 threads t_1, t_2, t_3, t_4 , the division would be as follows:

t_1	1,5,9,13
t_2	2,6,10,14
t_3	3,7,11,15
t_4	4,8,12,16

In the current implementation of the program all loops iterating over the adjacency or similarity matrices are constructed so that the workload associated with each index i is proportional to $n-i$. This results in a slightly uneven distribution of the workload over the

⁶“OpenMP is a specification for a set of compiler directives, library routines, and environment variables that can be used to specify high-level parallelism in Fortran and C/C++ programs.” [78] “The OpenMP ARB (Architecture Review Boards) mission is to standardize directive-based multi-language high-level parallelism that is performant, productive and portable.”[79]

threads, when using the cyclic division described above. However, the loss in efficiency of the parallelization is assumed to be small to negligible.

Although parallelizing the correlation and variance calculations was considered most important, other program sections also lends itself to simple parallelization. Step 1 and 2 involving data preprocessing and file reading, can be performed independently of each other, and so, if more threads are spawned by the program than the number of input files given to the program, these processes are performed in parallel. Step 5 in the program, calculating the CSD-scores, is also parallelized in a manner similar to the correlation calculations; by cyclic division of the outer for-loop over available threads.

Chapter 4

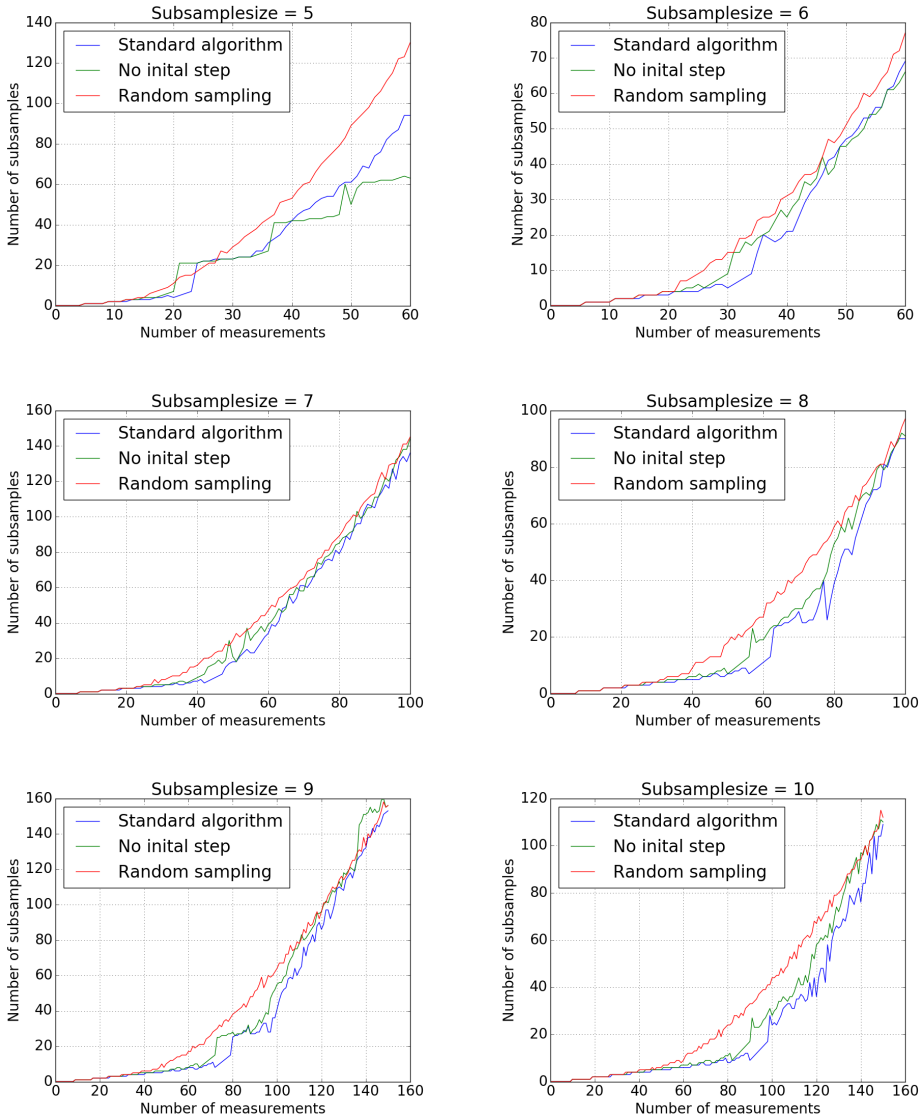
Results: Software

As CSD-CS is a program developed for streamlined generation of differential co-expression networks, it is interesting to evaluate the software's performance in terms of resource requirements. Aside from ease of use, an important aspect is the system requirements for running the program. Limiting maximum memory usage is a key factor in determining which systems are able to run the software, and how big data sets are supported. The following sections will present an analysis of the program performance with a special focus on the time- and memory-complexity of the program.

4.1 Subsampling algorithms

As described in chapter 3, three different subsampling algorithms were implemented in order to investigate possible improvements to the algorithm described in [1]. When testing the algorithms the random subsampling algorithm was set to terminate after 1000 unsuccessful attempts at drawing a subsample. For this termination limit the random subsampling algorithm generally outperformed the other two methods for the subsample sizes and number of measurements tested here (figure 4.1). As seen from the figures, the number of subsamples generated by the random subsampling follows the number of measurements more steadily and with fewer variations than the two other methods. This result might seem surprising, as one would expect that random drawing would introduce a certain level of variability in the number of subsamples generated. Although this is likely to be the case, this result provides some evidence for the observed problems related to systematic drawing described in chapter 3. Even if the exact mechanisms that cause this behaviour are not fully known, it seems to limit the number of subsamples that can be drawn to such an extent that random drawing is more successful. As a result the CSD-CS software was implemented using the random subsampling algorithm described in section 3.5.3.

Figure 4.1 The number of subsamples generated by the different subsampling algorithms. The random subsampling algorithm generally outperforms the other two methods.



4.2 Time consumption and parallel efficiency

4.2.1 Time complexity analysis

A summary of the time complexity of different program sections is given in table 4.1.

Table 4.1: The time-complexity of different processes in the *CSD-CS* software.

Process	Time-complexity
Preprocessing	$O(m \cdot n \cdot \log(n))$
File reading	$\Theta(m \cdot n)$
Correlations	$\Theta(m \cdot n^2)$
Subsampling	$O(k \cdot m \cdot (s_{max} + f_{max}))$
Variance estimation	$\Theta((k \cdot \log(k) + 1) \cdot s \cdot n^2)$
CSD-scores	$\Theta(n^2)$
Threshold estimation	$\Theta(s_p \cdot \frac{1}{p})$
Adjacency matrix	$\Theta(n^2)$
File writing	$\Theta(n^2)$

As seen from table 4.1 the dominating factor for determining the time-complexity of the program is the number of genes n in the data sets. Since n is also expected to be larger than all other parameters (except for maybe $1/p$), the program is thus expected to have a running time proportional to n^2 . Of the listed processes, the correlation and variance calculations are responsible for the major contribution to the wall-time. To verify the expected time-complexity, the CPU-time used by the program for datasets containing a varying number of genes n was tested along with the corresponding CPU-time related to the correlation and variance calculations. The results can be seen in figure 4.2.

The program was run using two input files containing $m = 100$ data points for each of the n genes. The other parameters were $k = 10$ and $p = 10^{-5}$. As seen in figure 4.2, for low n the time used by the program is not dominated by the correlation and variance calculations. The majority of the time-consumption in these cases is caused by the threshold estimations, whose time-complexity is independent of n . As the time-complexity of the threshold estimation is independent of n , the effect of these calculations on the overall time-consumption of the program quickly becomes negligible as n increases. Based on the asymptotic time-complexities presented in table 4.1 one would expect the contribution from the correlation and variance calculations to dominate the overall time-consumption. This is verified by the observations from figure 4.2.

Figure 4.3 confirms that the time-complexity of the program is in fact proportional to n^2 . The blue line shows the CPU-time of the program as a function of n , when run on 10 cores, with $m = 100$, $k = 10$, $t = 1000$, $p = 10^{-5}$ and $s_{imp} = 10000$. The red line shows a second degree polynomial fit on the form:

$$y = (3.78 \cdot 10^{-5}) \cdot n^2 + (2.47 \cdot 10^{-2}) \cdot n + 178.8 \quad (4.1)$$

Figure 4.2 The figure shows the CPU-time used by CSD-CS for input data containing 100 measurements for a various number of genes $n \in [1000, 25000]$. The CPU-time is initially dominated by the contribution from the threshold estimation algorithm, which runs in approximately constant time for a given combination of p and s_{imp} . The figure shows the strong influence of the number of genes in the data set on the time consumption of the program. For large n the time consumption is purely exponential and approximately equal the time consumption of the correlation calculations.

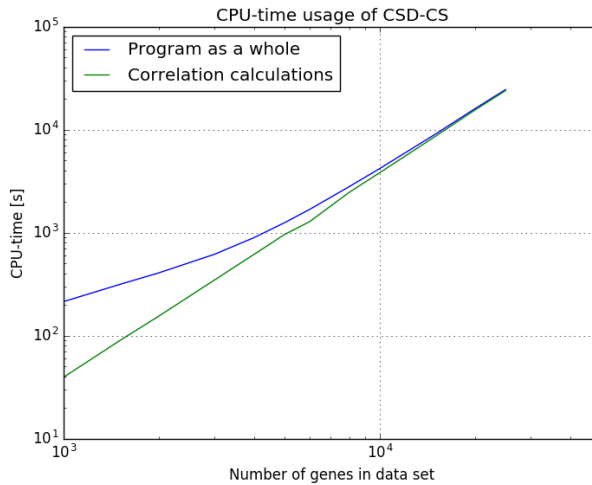


Figure 4.3 The figure shows how the CPU-time of the program depends on the number of genes in the analyzed data-sets. A second degree polynomial fit (the red line) verifies the estimated time-complexity of the program $O(n^2)$.

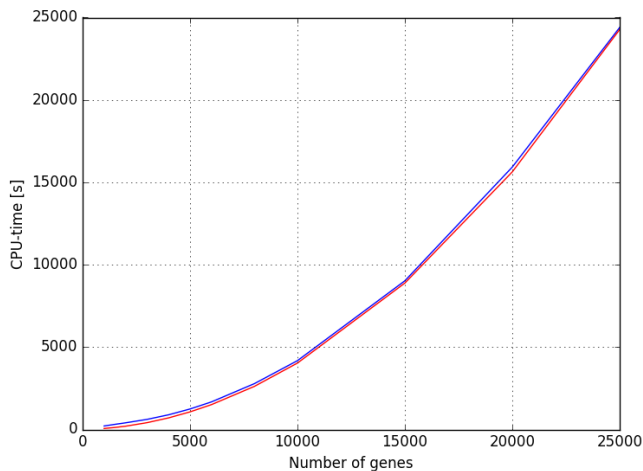
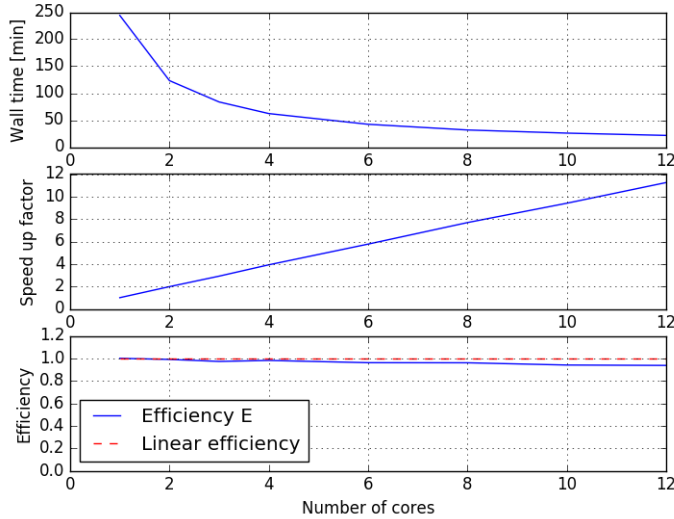


Figure 4.4 Result of parallelizing the correlation and variance calculations. The top figure shows how the wall-time of the program section follows the the number of cores used. The middle plot show the speed-up-factor S , and the bottom figure shows the efficiency E of the parallelization.



4.2.2 Parallelization of correlation and variance calculations

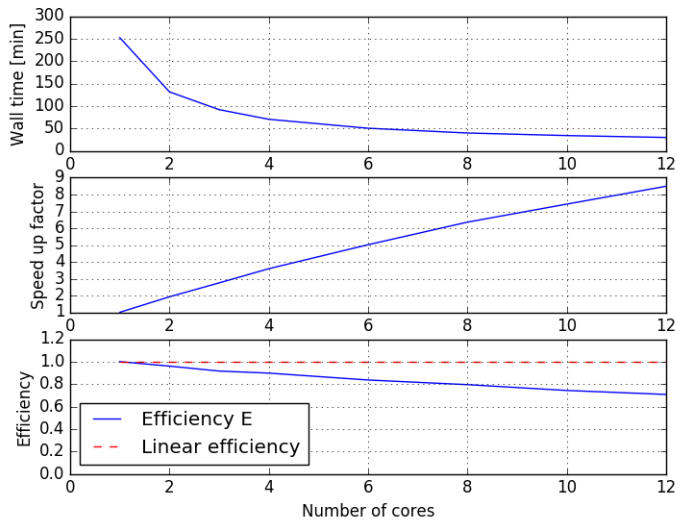
As a response to the strong influence from the correlation and variance calculations on the time consumption of the program, a parallelization scheme was implemented to reduce the wall-time of the program. In evaluating the effect of the parallelization two parameters are of main interest; *speed-up factor* S and *efficiency* E . They both convey similar information, and are defined as follows:

$$S = \frac{T(1)}{T(c)} \quad (4.2)$$

$$E = \frac{S}{c} \quad (4.3)$$

Here $T(c)$ is the wall-time of the program when run on c cores. The speed-up factor is thus a measure of how much faster the program terminates when run on c cores, while the efficiency gives the relative speed-up of adding another core. Figure 4.4 shows the wall-time, speed-up, and efficiency associated with the parallelizing the correlation calculations.

Based on the way that the correlation calculations are parallelized, the highest expected efficiency would be linear. At linear efficiency the speed-up is the same as the number of cores used to execute the code. Figure 4.4 shows a slight deviation from linear efficiency. Two possible explanations for this are related to increased overhead, and imperfect loop-division. When parallelizing code there are always some associated overhead, related

Figure 4.5 The effect on the overall program performance from all the parallelized code.

to communication, copying memory and setting up the jobs for each of the cores. The amount of overhead generally increases with the number of cores. An additional factor comes from imperfect division of the work load over the cores. As described in section 3.6, the parallelization scheme uses a cyclic division of the loop-indices over the cores, with uneven workload associated with each index. Although this may cause a slightly uneven distribution of labour over the cores, it is less than differences in work load caused by assigning a different number of indexes to each core (as will happen with for instance 5 indices divided over 2 cores), and is not expected to reduce the efficiency significantly. As the loss in efficiency is fairly small and the observed efficiency is close to linear, the parallelization is considered successful.

4.2.3 Further parallelization

As explained in section 3.6, step 1, 2, and 5 in the program¹ was also parallelized. While section 1 and 2 have 0 efficiency for cases with more cores than input-files, process 5 is assumed to have the same efficiency as the parallelization of the correlation and variance estimation. The effect of all parallelization on the wall-time used by the program as a whole, as well as the overall speed-up and efficiency of parallelizing the different program sections can be seen in figure 4.5.

As seen from figure 4.5, a program consisting of both serial and parallel code results in a drop in efficiency. As the efficiency is reduced, the effect on the wall-time from adding further cores diminishes and the wall-time converges towards its minimum value T_{min} . The wall-time can be modeled by $T(c) = T_s + T_p(c)$, where $T_p(c)$ is the wall time of

¹See list in section 3.1.

the parallelizable parts of the program, while T_s is the wall time of the serial parts and $T_s = T_{min}$. This model results in the following formula for the speed-up factor:

$$S = \frac{T(1)}{T(p)} = \frac{T_s + T_p(1)}{T_s + T_p(c)} = \frac{T_{CPU}}{T_s + T_p(c)} \quad (4.4)$$

This formula predicts a drop in speed-up factor and thereby efficiency, as a result of the unparallelized code in the program, which fits well with observations.

4.2.4 Memory usage

An important factor in the programs portability is its maximum memory usage. How large data sets are supported by the machine executing the code, depends mainly on how much memory (RAM) the system has. Although the program may execute even if it requires more memory than available, the increased overhead caused by memory allocation and reading and writing to long term memory makes the program run significantly slower. As an example, a former version of the program had a wall-time of approximately 11.5 minutes for data sets of $m = 100$ and $n = 1.5 \cdot 10^4$ when executed on 10 cores. When increasing n to $2 \cdot 10^4$, the peak memory use exceeded the amount available on the system, resulting in a wall-time of approximately 4.7 hours. The expected wall-time for data sets of this size were estimated to 20 minutes by extrapolating the growth observed in scaling n for lower values.

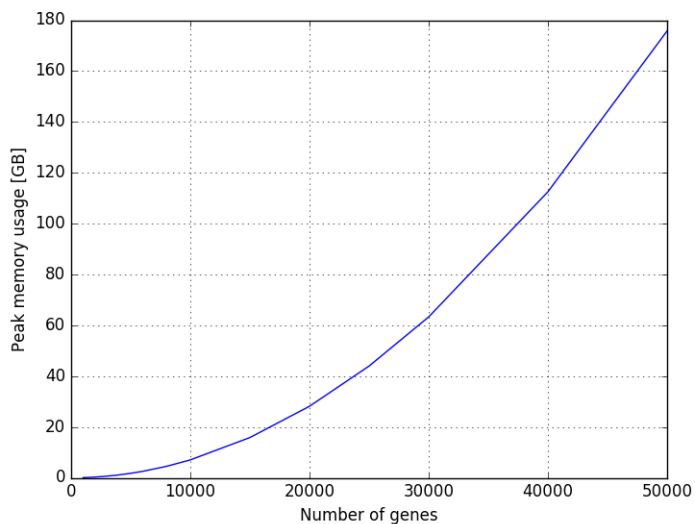
The memory usage of the current version of *CSD-CS* scales primarily with the number of genes n in the data sets. Figure 4.6 shows the peak memory usage as a function of n . As the majority of memory consumption comes from storing the similarity-matrices and adjacency-matrices, the maximum required memory scales proportional to n^2 . This is typically much larger than $m \cdot n$ for most data sets, and so the the memory usage is approximately independent of m . The software is also implemented so that number of subsamples generated, s , does not contribute significantly to memory consumption.

4.3 Program testing

The program was debugged and tested in several steps. All functions have been tested individually to verify their behaviour. Each step of the program (as listed in section 3.1) was also tested before assembling the program. All valid combinations on input and output files have been tested, to verify the program's behaviour.

After finishing the program, an artificial test case was created to check that the software calculations generates correct results. Two 8×8 matrices with tailor-made test-data were fed to the program, and the *output*-parameter was set to *all*. For both files, all of the $n(n - 1)/2 = 28$ gene-pairs, had their correlation and variance verified, along with their corresponding CSD-scores. All results were as expected.

Figure 4.6 The maximum memory usage of the program as a function of the number of genes in the data sets n .



Results: Application to empirical gene expression data

This chapter features an analysis of the rheumatoid arthritis network generated using *CSD-CS*. The goal is to illustrate how the developed software, in combination with tools from network science and systems biology, can be used to analyze gene expression patterns across appropriately chosen data-sets. Although the identification and implication of unknown genetic factors is the target of the analysis, the purpose of this thesis is not to perform an extensive biological analysis. RA is a highly heterogeneous and complex autoimmune disease arising from both genetic and environmental factors. An extensive literature review and thorough analysis of the network is possibly required in order to identify disease related biology in the network. As the focus is on the methodology rather than the actual results, the analysis will be restricted to superficial evaluations. The analysis will be limited to examining the presence of genes of known associations and related pathways.

5.1 Data collection

5.1.1 Rheumatoid arthritis expression data

The expression data for rheumatoid arthritis used in this thesis, was collected by Walsh et al. in their article on “Integrative genomic deconvolution of rheumatoid arthritis GWAS loci into gene and cell type associations” [82]. The data consist of whole-blood RNA expression data collected from a cohort of 377 patients with active rheumatic arthritis despite Methotrexate¹ therapy. The paragraph below is taken from the article, describing the exact procedure used for collecting the RNA-expression data.

“Peripheral whole blood was collected in PAXgene tubes (Preanalytix, Switzerland). RNA was isolated using the Qiagen Biorobot (Qiagen, Valencia, CA, USA), which fol-

¹Methotrexate is an anti-inflammatory drug used for treatment of autoimmune disease, in particular rheumatic diseases and myositis.

lowed the protocol from the Qiagen PAXgene MDX kit (cat# 752431), and was modified to collect both total and microRNA. Subject cDNA was amplified through utilization of the NuGEN Ovation Pico WTA System V2 (NuGEN, San Carlos, CA, USA). Microarray hybridization was performed on GeneChip Human Genome U133 Plus 2.0 Array according to the manufacturers protocol (Affymetrix, Santa Clara, CA, USA). Data were normalized using Robust Multi-array Average (RMA) algorithm and log base 2 transformed using R. Data are available from NCBI GEO with accession number GSE74143.” [82]

The expression data from the Affymetrix GeneChip Human Genome U133 Plus 2.0 array consisted of expression values for 54715 probes for each individual. This data set was reduced to 20741 points by mapping the expression values for all probes to their associated gene, and throwing away all measurements from probes related to non-coding DNA-segments. For genes with multiple associated probes, the expression value was set as the average of the expression values of the probes.

5.1.2 Control group expression data

In order to effectively probe the regulatory mechanisms underlying RA using the CSD-framework, a reference data set was needed. This was acquired from the *Genotype-Tissue Expression (GTEx) Project* [83], a database containing various types of genetic data². GTEx data is collected post mortem from donors fulfilling the following criteria:

1. $21 \leq \text{Age (years)} \leq 70$
2. $18.5 < \text{Body Mass Index} < 35$
3. Time between death and tissue collection less than 24 hours
4. No whole blood transfusion within 48 hours prior to death
5. No history of metastatic cancer
6. No chemotherapy or radiation therapy within the 2 years prior to death
7. Generally unselected for presence or absence of diseases or disorders, except for potentially communicable diseases that disqualify someone to donate organs or tissues would also be disqualifying for GTEx.

The data is thus considered to originate from healthy individuals. The expression data used in this thesis originates from whole-blood samples from 191 individuals in GTEx v4. The expression data was collected using an Affymetrix GeneChip Human Gene 1.0 ST Array, and reduced from probe expression to gene expression in a similar manner as for the RA-data, resulting 18490 protein coding genes.

²<https://www.gtexportal.org/home/>

Table 5.1: The input parameters used by CSD-CS in creating the network analyzed in this thesis.

Parameter	Value	Description
-i	arthritisData.txt	Gene expression data from rheumatoid arthritis patients
-i	controlData.txt	Gene expression data from from healthy control group
-input	"ged"	Specifying input type
-output	"all"	Requesting all data to be written to file
-output	"filtered"	Requesting file containing network node pairs
-k	10	Subsample size
-p	0.00001	Importance level
-iSamples	10 000	Number of samples to be drawn for threshold estimation
-threads	10	The number of parallel threads spawned by the program
-format	"vertical"	Specifying the orientation of the expression data in the input files
-randomSeed	3	Seed for random number generator
-maxSubsamples	100	Upper bound for number of subsamples generated by the subsampling algorithm
-terminationLimit	1 000	Maximum number of failed attempts at drawing a subsample allowed by the subsampling algorithm before termination

5.2 Network construction

The differential co-expression network analyzed in this thesis was generated using the purpose-built software; *CSD-CS*. The network was constructed using the two data-sets described in the sections above, originating from rheumatoid arthritis patients and a healthy control group. The program was executed with the parameters listed in table 5.1.

5.3 Network analysis: tools and procedure

The main tools used in analyzing the network was the *Cytoscape v2.8.1* [74] software environment, and *python*. The format of the output file containing the network generated by *CSD-CS* is intended for easy use with Cytoscape. Cytoscape was used for visualization of the network, calculation of node- and network-parameters using the *network analyzer*-module [75], as well as construction of degree-preserving random networks. The degree-preserving random networks were used to identify which properties of the network arise purely from the nature of the degree distribution, and which properties can be attributed

to other organizing principles. The resulting data was exported and further analyzed using *python*.

The *NetworkX*-python package combined with an implementation of the *Louvain* community detection algorithm³ was used to identify network modules. The algorithm assigns an index to each node indicating their module affiliation. These indices were transported back into Cytoscape for further analysis of the modules.

The *scipy*-library was used to estimate correlations between node-parameters within the network. The Spearman rank-based correlation was used here to evaluate any potential relations between the node parameters. Spearman correlation was chosen over Pearson correlation to reduce the constraint on observable relations, and test for any monotonous correlation between the node-parameters. This choice opens up for capturing potential non-linear correlations which might be relevant to the network structure. Not all the node parameters calculated by Cytoscape were included in the correlation matrices. Several parameters have such similar definitions that they show almost perfect linear Pearson correlation and perfect Spearman correlation. For these cases one of the parameters were omitted in the analysis, as they gave rise to pairs of identical correlation scores, thus providing no new information to the analysis. The omitted parameters were *stress*, *average shortest path-length*, and *radiality* (on account of *betweenness centrality*, *closeness centrality*, and *betweenness centrality* respectively).

Next enrichment analysis was performed for network components and modules of interest, using the *Enrichr* web-site [84, 85]. Enrichment analysis consists of matching a set of genes, with genes of known biological associations. A significant enrichment means that the number of genes present, related to the specific biological factor/function, is higher than would be expected by chance. The *Enrichr* web page takes a list of genes as input, and matches it with transcription-, pathway-, disease-, cell-, and ontology-associated genes from various databases. Four different enrichment values are calculated. The first value is a p-value based on the Fisher exact test. The test assumes a binomial distribution and independent probabilities for any gene belonging to any set. The second score is an adjusted p-value (here referred to as the q-value), calculated by using the Benjamini-Hochberg method for correction for multiple hypothesis testing. The third score is a rank based ranking. It is calculated by running the Fisher exact test for many random gene sets in order to estimate the mean rank and its standard deviation. The corresponding z-score is then calculated to evaluate the deviation from the expected rank. The last score is a combination of scores 1 and 3, calculated as the product of the logarithm of the p-value and the z-score. The enrichment-scores may give valuable information about the underlying biology of the network, and its modules.

As a further step in analyzing the biological information captured by the co-expression network, the function and associations of central nodes in the network were evaluated. Nodes with high, yet non-trivial, parameter scores were identified. The most prominent ones were evaluated for possible links to the disease state. The *genecards* web-resource [86] was used to acquire information about the identified genes.

Finally, a search for genes of known relevance to the disease state were performed. Their distribution in the network were visualized, in order to evaluate possible regions containing a high density of associated genes. Such regions might capture important bi-

³<https://github.com/taynaud/python-louvain/>

ological functions relating to the disease state, and implicate roles of non-associated adjacent genes in the disease state. A special interest is given to *specific* (S) and *diverging* (D) interactions, as these capture differences in gene regulation across healthy and disease states.

5.4 Network properties

Figure 5.1 shows the full CSD-network generated using the *CSD-CS*. One striking feature of the network is that it consists of two major components. These components are separated from one another, and appears different by visual inspection. The left component consist of a majority of S-interactions, and appears to have a more connected and dense structure than its neighbor. As the right component consists almost exclusively of C- and D-type interactions, the two components will from now on be referred to as the S- and the CD-components respectively.

Table 5.2 shows number of nodes and edges, and the distribution of interaction types for the network and its main components. Although the two components consist of approximately the same number of nodes, the S-component contains over 50% more edges than the CD-component. This confirms the observation that the S-component is more connected than its neighbor. Notably, the CD-component only contain one single S-interaction, while the S-component consists of a majority of them. In the context of analyzing a disease versus non-disease state, S- and D-type interactions are of greatest interest. The reason is that a change in co-expression pattern across the two cohorts is more likely to be related to phenotypic differences. Thus both the identified components are of interest for further analysis, as they both may convey relevant biological information.

To further analyze the differences between the different components, a set of network parameters were calculated for the network as a whole and its two main components individually. The results can be seen in table 5.3. The calculated parameters supports the notion of structural differences between the two network components. While the CD-component has a lower density, average degree, radius, diameter and characteristic path-length, it shows higher centralization, heterogeneity, and clustering coefficient. Thus, despite the component being more sparse and less connected, it seems to have more prominent clustering around hubs.

Table 5.2: The number of nodes and edges in the networks and its major components, along with the distribution of specific interaction types.

	Full network	S-component	CD-component
Nodes:	1064	481	428
Edges:	2164	1267	778
<i>C-type</i> :	672	343	263
<i>S-type</i> :	783	767	1
<i>D-type</i> :	708	157	514

Figure 5.1 The figure shows the full CSD-network produced by CSD-CS as visualized by Cytoscape. Conserved interactions are indicated by blue edges, while specific and diverging are shown in green and red respectively.

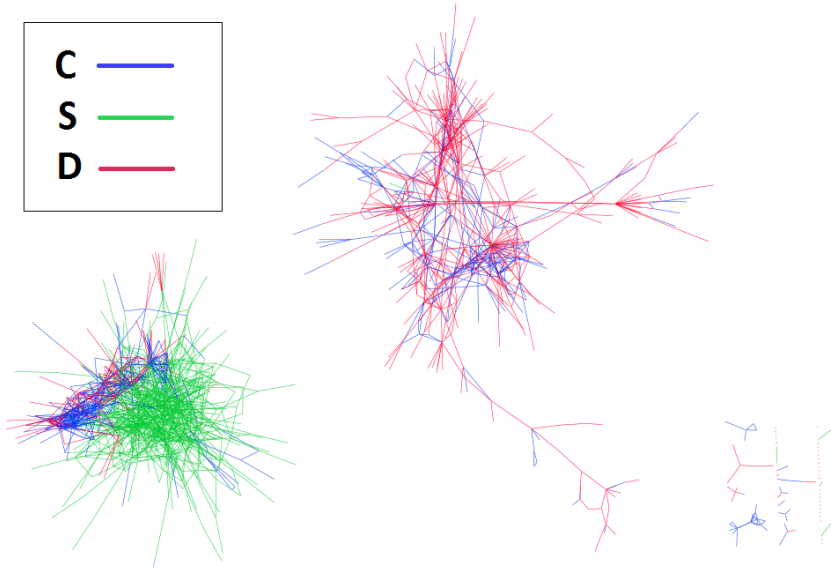


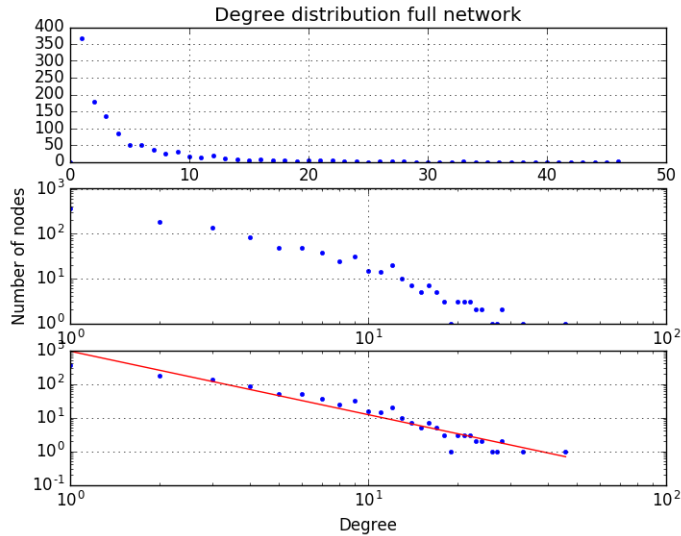
Table 5.3: The network parameters for the full network and its two main components.

	Full network	S-component	CD-component
Radius	-	6	10
Diameter	18	9	18
Characteristic pathlength	4.813	4.113	5.669
Average degree	4.068	5.261	3.598
Network density	0.004	0.011	0.008
Network centralization	0.040	0.058	0.097
Network heterogeneity	1.120	0.944	1.207
Clustering coefficient	0.121	0.111	0.147

The degree distributions of the full network and the two main components were plotted and inspected. There were no major differences in the degree distributions. Figure 5.2 shows the degree distribution of the full network. The degree distribution follows approximately a power law with $\gamma = 1.886$ (subplot 3), indicating a scale-free network.

The neighborhood-connectivity distribution of the network and its major components was also evaluated. While the full network and the S-component show positive assortativity, the CD-component shows disassortativity (the fitted power-law has a negative exponent). What this means is that on average high-degree nodes in the CD-network tend to favour connections with lower degree-nodes, and vice-versa. This trend is opposite for the S-component. This further confirms the observations of a more dispersed, hub-centered

Figure 5.2 The figure shows the degree distribution of the full network. Subplot 2 and 3 shows log-log plots of the degree-distribution. As indicated by the fitted line in subplot 3, the degree-distribution follows a heavy-tailed power-law (with $\gamma = 1.886$).



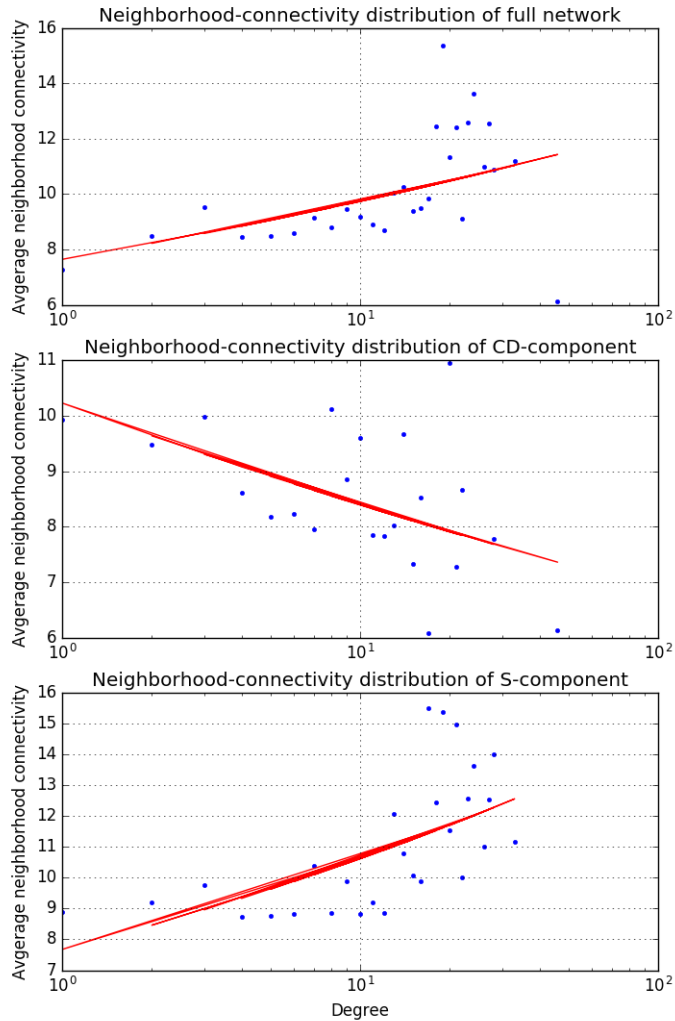
topology for the CD-component.

5.4.1 Component analysis

Although the two network components are part of the same regulatory network, and only appear distinct due to the choice of importance level used in the CSD-CS, a further analysis of their differences was deemed to be of interest. The idea was to detect any differences in their structure and organization, and evaluate whether this was reflected in the properties of the underlying biology. Another advantage of analyzing the two components individually is the ability to more accurately probe local differences in network structure when analyzing intra-component modules. Further comparisons with the full network will not be considered, as the information encoded in the full network is heavily dependent on both the main components, and thus will mostly be an average of the their properties. It will therefore yield a minimal amount of new information.

An interesting question in the evaluation of network parameters is; how do the components differ from random networks of equal degree distribution? This question yields information about which parameter values are expected due to the nature of the degree distribution, hence illuminating whether the observed parameter values can be attributed to organizing principles rather than randomness. To examine this, 10 degree-preserving random networks were generated for each of the two main components. The same network parameters as in the section above were calculated for each of the random networks. Tables 5.4 and 5.5 shows comparisons between the average network parameters acquired

Figure 5.3 The neighborhood-connectivity distributions of the full network and the two main components, along with a fitted power law (red line) . Interestingly the CD- and S-components show opposite assortativity.



from the random networks, and the corresponding component-parameters. The full set of network parameters from all the random networks can be found in appendix C along with an example of a random networks generated from each of the components.

Table 5.4: The table shows the average of the network parameters calculated for the degree-preserving random networks based on the S-component. Difference in parameter values between the S-component and the random networks are highlighted in red. The radius is not comparable as the random network give rise to pairs of nodes separated from the main component yielding a radius of 1.

	S-component	S-based random
Radius	6	-
Diameter	9	8.3
Characteristic pathlength	4.113	3.682
Average degree	5.261	5.261
Network density	0.011	0.011
Network centralization	0.058	0.058
Network heterogeneity	0.944	0.944
Clustering coefficient	0.111	0.025

Table 5.5: The table shows the average of the network parameters calculated for the degree-preserving random networks based on the CD-component. Difference in parameter values between the CD-component and the random networks are highlighted in red. The radius is not comparable as the random network give rise to pairs of nodes separated from the main component yielding a radius of 1.

	CD-component	CD-based random
Radius	10	-
Diameter	18	9.6
Characteristic pathlength	5.669	4.081
Average degree	3.598	3.598
Network density	0.008	0.008
Network centralization	0.097	0.097
Network heterogeneity	1.207	1.207
Clustering coefficient	0.147	0.023

The results from the two tables shows that the real networks have a larger diameter, characteristic pathlength and clustering coefficient than than their random equivalents. This indicates that the real networks show a more distributed, cluster based topology. Figures 6 and 7 in appendix C supports this notion, as the topology of these networks is more revolved around a single central cluster.

To take this analysis a step further, the Spearman correlation between different node parameters were calculated for the two components and their respective random networks. The results can be seen in tables 5.6 and 5.7. Once again, the presented values for the random networks is the average of those obtained from each of the random networks.

For the CD-component there are several notable differences between the node-parameter correlations in the component and its randomized equivalents. The clustering coefficients of nodes in the CD-component show a much higher correlation with the topological coefficient than for the random networks. On the other hand clustering coefficient is more correlated with the betweenness centrality in the random networks. The first of these observations can be seen as an confirmation of the clustering tendency of the CD-components as discussed above. What this means is that for the CD-component a high level of clustering around a node implies a much higher level of sharing of neighboring nodes, than for both the random networks and for the S-component. This might account for the more separated, localized clustering observed in the CD-component than in any of the others.

There are some other minor changes between both the CD-component and its random networks, and the S-components and its random networks, but these are limited to around 0.2 difference in correlation scores, from which it is difficult to draw conclusions about changes in network properties.

Table 5.6: The table shows a comparison between the Spearman correlation between different node parameters within in the CD-dominated network component (top half) and degree-preserving random networks based on the CD-component (bottom half). Blue indicates strong correlations, while red indicates notable differences in correlation scores.

	C_C	C_i	k_i	ϵ	k_{ii}	T_i
C_B	0.4608	0.3303	0.8920	-0.3564	-0.0647	0.4757
C_C	-	0.5195	0.6067	-0.6068	0.6637	0.2185
C_i	-	-	0.6457	-0.3479	0.2964	0.4873
k_i	-	-	-	-0.4305	0.0589	0.5243
ϵ	-	-	-	-	-0.3371	-0.2383
k_{ii}	-	-	-	-	-	0.0579
C_B	0.5731	0.5009	0.9567	-0.4358	0.1318	0.5291
C_C	-	0.5097	0.6160	-0.8392	0.4300	0.1783
C_i	-	-	0.5689	-0.4027	0.2737	0.0855
k_i	-	-	-	-0.4653	0.1691	0.5243
ϵ	-	-	-	-	-0.3261	-0.1198
k_{ii}	-	-	-	-	-	0.1383

Table 5.7: The table shows a comparison between the Spearman correlation between different node parameters within in the S-dominated network component (top half) and degree-preserving random networks based on the S-component (bottom half). Blue indicates strong correlations, while red indicates notable differences in correlation scores.

	C_C	C_i	k_i	ϵ	k_{ii}	T_i
C_B	0.8011	0.1962	0.8937	-0.6564	-0.0788	-0.0572
C_C	-	0.4384	0.8833	-0.7934	0.3630	-0.0264
C_i	-	-	0.4954	-0.2496	0.5623	0.1537
k_i	-	-	-	-0.6669	0.1427	-0.0105
ϵ	-	-	-	-	-0.2358	-0.0138
k_{ii}	-	-	-	-	-	0.1601
C_B	0.7998	0.5148	0.9506	-0.6002	0.0636	-0.0200
C_C	-	0.5980	0.8593	-0.7737	0.4223	-0.1071
C_i	-	-	0.5855	-0.4317	0.2539	-0.1527
k_i	-	-	-	-0.6387	0.1020	-0.0490
ϵ	-	-	-	-	-0.3566	0.0033
k_{ii}	-	-	-	-	-	0.0956

5.4.2 Module analysis

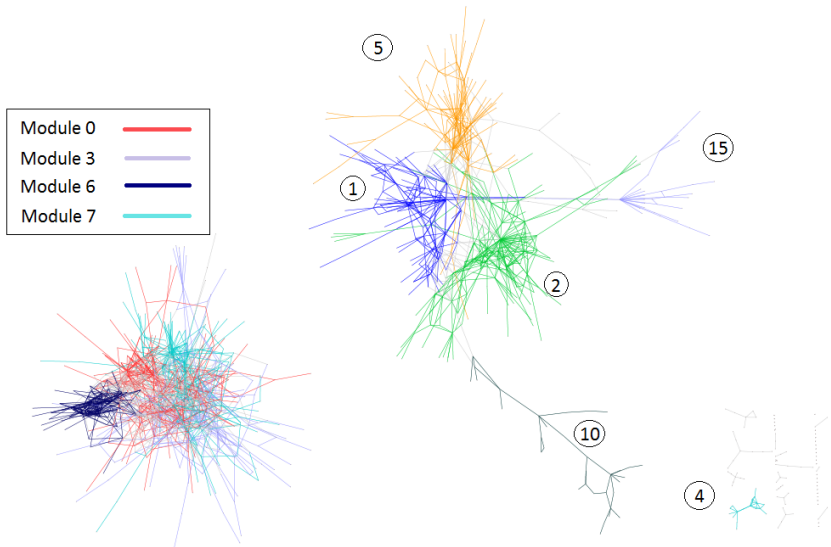
In order to further refine the topological analysis of the network, the Louvain community detection algorithm was applied to identify network modules. 58 modules were detected, but many of them were of negligible sizes, such as those created by the minor groups of nodes separated from the main components. The 10 biggest modules, listed in table 5.8, can be seen plotted in figure 5.4.

Table 5.8: The 10 biggest modules identified by the Louvain algorithm sorted by the number of nodes.

Module ID	Nodes	Location
2	155	CD-component
0	154	S-component
3	134	S-component
7	126	S-component
5	107	CD-component
1	98	CD-component
6	64	S-component
10	34	CD-component
15	30	CD-component
4	18	Separate

Node-parameter correlations were calculated for all the modules (except for module 4 as this is separated from the main components) and a comparative analysis similar to the one described in the section above was performed. The comparison-tables can be found in appendix D, and the corresponding significance scores can be found in appendix

Figure 5.4 The 10 largest modules detected in the CSD-network by using the Louvain community detection algorithm.



E. Due to the high number of nodes involved in the most of the modules, the detected correlations have high significance even for very low correlation scores, and are thus of little practical interest. The modules showing the least similarity with their component and neighboring modules are modules 10 and 15 for the CD-component, and module 6 for the S-component. Figure 5.4 shows that both modules 10 and 15 are peripheral modules with a sparse connections, that visually distinguish themselves from the other CD-modules. Module 6 in the S-component is harder to distinguish from its neighbors visually, except for observation that it seem less interconnected with the other modules of the component. Interestingly module 6 seems to be localized to the area of the S-component containing a high level of C- and D-type interactions. Upon inspection, module 6 turned out to consist of 174 C-interactions, 13 S-interactions and 37 D-interactions. As argued earlier, a high level of conserved relationships is less likely to explain phenotypic differences related to disease. It is interesting however to see what biological information is captured by module 6 as its neighboring modules show a highly S-dominated co-expression pattern, which the biology of module 6 might be related to.

5.5 Biological analysis

5.5.1 Enrichment analysis

The Enrichr [84, 85] web site was used for enrichment analysis of all detected modules, as well as the two components and the network as a whole. In addition, the C-,S- and D-interaction networks were separated and tested for enrichment individually. Main fo-

cus was given to the categories, *Pathways*, *Ontologies* and *Disease/Drugs*, and processes related to inflammation, immune system or other processes relevant for the disease state.

The general trend was that large groups of genes, such as the full network, the large components, and the interaction networks showed lower enrichment values than the modules. Usually the same biology was highlighted, but with lower scores, indicating that the modules in fact show more specific biological associations than the larger groups of genes. It should be noted however, that the requirement for high enrichment scores, in terms of number of matching genes, increase with the number of genes assessed, adding to this trend. Another feature observed where very high enrichment scores for tissue types when assessing large groups of genes. Typically many different organs were highlighted with high enrichment scores, indicating that a lot of the genes in the network are related to general cellular processes and are not tissue specific.

Many enrichment scores were considered too low to be of interest. Typically, biology related to cell signaling pathways, blood coagulation and clotting, T- and B-cells, and other blood related factors were slightly enriched. In order to assess whether this is caused by the fact that whole-blood samples were used, or the disease state, the interaction-specific networks are useful. In general the S- and D-networks are assumed to capture more of the disease related biology, while the C-network will highlight conserved biological processes.

For the S-network the majority of prominent enrichments were related to blood clotting cascade, prothrombin activation, and blood coagulation. A strong signal for carcinoma was also detected. The C-network showed slight enrichment for signaling pathways related to T-cells, and an equivalent carcinoma signal to the S-network. Similar to the C-network, the D-network also showed slight enrichment for processes related to the immune system. Here B-cell and T-cell receptor signaling showed some enrichment, but at the same levels as for the C-network. Interestingly transcription related to the gene PRDM1, an RA-associated gene [87], showed some enrichment. Also TYK2-knockdown showed some enrichment, with TYK2 being another RA-related gene [88].

Module level enrichment were also performed, to evaluate more local enrichment patterns. No RA-specific associations were detected, but more general traits showed local enrichments. Modules 3,4,6, and 10 showed immune system related enrichments. Notably module 6 showed enrichment for apoptosis, death receptor signaling and abnormal NK-cells, all of which were not detected elsewhere. Module 3 showed strong enrichment for T-cell receptor complex, while module 10 showed strong enrichment for B-cell receptor complex and immunoglobulin complex. Module 4 showed enrichment for T-lymphocytes. Another interesting result, is the enrichment in T-cell receptor signaling for CD4+ T-cells in module 3 (although the enrichment value is a bit low). These T-cells have been assumed to have a role in RA pathogenesis [89]. Module 3 also turned out to have a strong signal for the blood clotting related biology observed in the S-network.

The enrichment analysis showed disease related enrichments, relevant for the systemic effects of RA. Module 3 showed enrichment for venous thrombosis, and module 4 for myocardial infarction. Module 6 showed enrichment for carcinoma, but also a small enrichment for osteoporosis, which might be related to the bone degrading nature of RA.

The discussed enrichment scores can be found in tables 5.9 and 5.10.

Table 5.9: The table shows the enrichment scores for the C-, S- and D-networks most relevant for RA. The listed scores are not the maximum observed scores, but are the ones most relevant with respect to RA.

Location	Description	p	q	z	Combined score
C-network	T-cell activation	0.004358	0.2136	-1.60	2.47
	T-cell receptor signaling pathway	0.001598	0.2479	-1.88	2.62
	Carcinoma	0.0002970	0.1485	-5.87	11.19
S-network	Blood clotting cascade	0.001012	0.2246	-2.02	3.01
	Formation of fibrin clot	0.001235	0.3114	-2.15	2.51
	Blood clotting, intrinsic pathway	0.0004520	0.2802	-3.19	4.06
	Intrinsic prothrombin activation pathway	0.00009493	0.007310	-1.63	8.03
	Carcinoma	0.0007571	0.1315	-5.76	11.69
D-network	PRDM1	0.0002810	0.08092	-1.74	4.37
	TYK2 knockdown	0.0003239	0.04616	-1.77	5.45
	T-cell receptor signaling pathway	0.01363	0.2800	-1.66	2.11
	B-cell receptor signaling pathway	0.001288	0.1803	-1.84	3.14

5.5.2 Biological functions of central nodes

An interesting analysis is whether central nodes of the network are related to the disease. If genes with key locations, both in terms of information transmission and connectivity, are in fact related to the disease, they may convey clues as to where in the network a potential disease signal is located. It will further support the working hypothesis that the network in fact captures biological information related to the disease state.

As the goal of this thesis is simply to illustrate the procedure for analyzing a differential gene co-expression network, the list of nodes assessed were restricted to the top 5 nodes for each chosen parameter. For a more thorough analysis additional nodes could be included, but this was not considered viable given the time available.

The top 5 nodes for each of the node parameters treated in the correlation analysis above, spare the eccentricity, were identified and are listed in tables 5.11-5.16. Only the two major components were included in order to avoid identifying nodes with trivially high parameter values due to the low number of neighbors. For some of the parameters a degree threshold was also selected in order to avoid identifying nodes with high parameter values caused by their low connectivity.

Table 5.10: The table shows the most interesting findings from the modular enrichment analysis. The listed scores are not the maximum observed scores, but the ones most relevant with respect to RA.

Location	Description	p	q	z	Combined score
Module 3	Blood clotting cascade	0.0004126	0.03032	-1.94	6.78
	Intrinsic prothrombin activation pathway	0.0004721	0.01558	-1.63	6.80
	Role of MEF2D in T-cell apoptosis	0.001046	0.01726	-1.71	6.96
	T-cell receptor complex	0.0001545	0.01746	-3.09	12.52
	T-cell receptor signaling in naive CD4+ T-cells	0.009098	0.1913	-1.67	2.76
	T-cell receptor signaling in naive CD8+ T-cells	0.005395	0.1913	-1.65	2.73
	Module 4	Helper T-cell	0.02703	0.3080	-7.33
Cytotoxic T-cell		0.02520	0.3080	-7.22	8.51
Blood clotting, Myocardial infarction		0.0004520	0.2802	-3.19	4.06
		0.001065	0.01810	-2.44	9.79
Module 6	Death receptor signaling	0.0004877	0.04150	-2.06	6.56
	Apoptosis signaling pathway	0.004283	0.02141	-1.70	6.53
	Abnormal NK-cell physiology	0.0004300	0.07495	-2.62	6.80
	Carcinoma	0.03149	0.1207	-4.95	10.47
	Osteoporosis	0.03465	0.03465	-1.02	3.43
Module 10	IgA immunoglobulin complex	0.00004578	0.009820	-2.48	11.48
	IgA B-cell receptor complex	0.00006482	0.009820	-2.26	10.43

Table 5.11: The highest degree nodes from the network.

Node	Degree (k_i)
PPM1G	46
PSPC1	33
MTTP	28
IFT88	28
DNAJB5	27

Table 5.12: The nodes with the highest clustering coefficient and degrees higher than 13.

Node	Degree	C_i
BAG4	15	0.4190
C17orf64	17	0.4118
ACVR1B	18	0.3922
PXMP2	14	0.3846
AMIGO2	16	0.325

Table 5.13: The nodes with the highest betweenness centrality from within one of the two largest components.

Node	C_B
PPM1G	0.33573447
TMEM18	0.19230653
FOXD1	0.15981781
MTTP	0.15617865
MSX2	0.14097653

Table 5.14: The nodes with the highest closeness centrality from within one of the two largest components.

Node	C_C
PDPK1	0.32454361
CD72	0.32193159
PSPC1	0.32150033
PLEKHG3	0.32042724
GOLIM4	0.31978681

Table 5.15: The nodes with the highest neighborhood connectivity of the nodes with a degree higher than 15.

Node	Degree	k_{ii}
C17orf64	17	18.167
CD72	24	16.167
GOLIM4	21	15.439
PDPK1	19	15.368
ACVR1B	18	15.056

Table 5.16: The nodes with the highest topological coefficient from nodes with degree higher than 15.

Node	Degree	T_i
ACVR1B	18	0.1955
AMIGO2	16	0.1909
C17orf64	17	0.1731
C1orf227	16	0.1707
GPR4	20	0.1598

The web-resource *genecards*⁴ [86] was used as a primary step to identify the function of the identified genes. None of the genes had any known relevance to the disease state, but were related to various biological processes such as transcription, cell cycle, cell growth, protein transport and signaling pathways.

Of the identified genes, the most promising prospects to be related to RA are FOXD1 and CD72. The CD gene-family is a group of genes associated with cell surface molecules, often coding for receptors or ligands. Many CD-molecules are thus related to the function of the immune system, and several are associated with RA including CD2, CD21, CD28, CD40 and CD58. The CD72 is assumed to play a role in both B-cell [90] and T-cell proliferation [91] proliferation and differentiation. It is also shown to play important roles in other autoimmune diseases [92, 93, 94].

⁴genecards.org

The FOX (forkhead boxes) proteins are a family of transcription factors that play important roles in regulating gene expression related to cell growth, proliferation, and differentiation. Forkhead transcription factors also plays important roles in maintaining immune homeostasis, and FOXD1 have been shown to coordinate the regulation of $NF - \kappa B$ and $NF - \kappa B$ which are two key inflammatory transcription factors. It has been suggested to possibly prevent autoimmunity by directly regulating anti-inflammatory regulators of the $NF - \kappa B$ -pathway [95].

5.5.3 Localizing disease-associated genes

The final step in this analysis was evaluating the locations of genes with known biological associations to the disease state in the network. The idea is much the same as for the previous two steps; to identify potential disease signals in the network. The presence of disease-associated genes and their locations may convey interesting biological information. Potential clustering of associated genes may also imply roles of neighboring genes through so-called guilt-by-association.

In order to do this, the *Online Mendelian Inheritance in Man* (OMIM) [64] database, and several articles on the genetics of RA [58, 59, 60, 61, 62, 63] was consulted to identify genes with associations to RA. A list of 55 genes were identified (appendix A), and matched against the constructed network. Of these, only three genes were found to be present in the network. The relevant genes are CCL21, CCR6 and CD5. Of these three genes, only CD5 was located in one of the major network components. Also in this section the *genecards* web-resource [86] was used to identify the functions of genes.

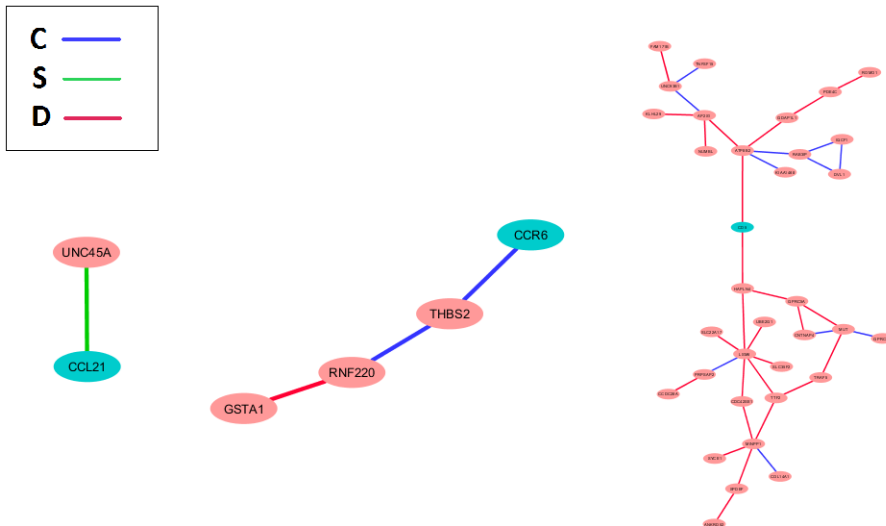
CD5 was found to be located in module 10 with two neighbors (HAPLN4 and ATP8B2) both connected to CD5 through D-interactions. CD5 is a gene codes for a member of the *scavenger receptor cysteine rich* (SRCR) superfamily. The protein is a type 1 transmembrane glycoprotein found on the surface of thymocytes, T-cells and a subset of B-cells. It is assumed to have roles in T-cell proliferation and B-cell development.

Its first neighbor, ATP8B2, is connected through a D-type interaction. Interestingly, AP8B2 is also connected to KIAA1468. KIAA is a group of protein-coding sequences of uncharacterized human genes. The gene KIAA1468 has been associated with chronic recurrent multifocal osteomyelitis (CRMO) [96]. CRMO is a rare autoinflammatory skeletal disorder of unknown aetiology [97]. AP8B2 is thus connected to 2 different genes with associations to inflammatory skeletal disorders. A further investigation of AP8B2 and KIAA1468 in relation to RA, could yield interesting information about the genetics of RA.

The other gene connected to CD5 is HAPLN4 (Hyaluronan And Proteoglycan Link Protein 4). Also HAPLN4 is connected through a D-interaction. The relevant gene has GO-annotations [98] for *extracellular matrix structural constituent* and *hyaluronic acid binding*. The association between CD5 and a hyaluronic acid binding protein is interesting in relation to RA. Hyaluronic acid is a major constituent of synovial fluid, and has been related to autoimmune inflammation and disease progression in autoimmune insulinitis patients with type 1 diabetes [99] and autoimmune encephalomyelitis in mouse [100, 101]. Some studies have implicated a role of hyaluronic acid and hyaluronan synthesis in RA pathogenesis [102, 103], while others indicates anti-inflammatory effects of hyaluronic acid [104, 105]. Hyaluronic acid definitely plays some role in RA pathology, and thus it

would be interesting to evaluate how HAPLN4 and CD5 relates to this. The gene CD44, coding for the hyaluronan receptor [106] and with roles in inflammation [107], was also found in the network close to module 10.

Figure 5.5 The figure shows the two small components where the RA-associated genes CCL21 and CCR6 are located, as well as module 10 where CD5 is located. The relevant genes are highlighted by a different color.



Both CCL21 (C-C Motif Chemokine Ligand 21) and CCR6 (C-C Motif Chemokine Receptor 6) are genes coding for proteins related to cytokine signaling and play immunoregulatory roles. The neighbor of CCL21, UNC45A codes for a myosin chaperone. The neighbor of CCR6, THBS2 codes for a glycoprotein that mediates cell-cell and cell-matrix interactions. One could ask whether the function of the glycoprotein coded by THBS2 is somehow related to the function of the CCR6 receptor, but no obvious connections were found in literature. The same was the case for CCL21 and UNC45A. Further, as both CCR6-THBS2 and THBS2-RNF220 shows a conserved co-expression pattern, it is reasonable to assume that CCR6 and RNF220 also shows a certain level of conserved co-expression, but less than that required by the chosen importance level. The RNF220 codes for an E3 ubiquitin ligase, a group of proteins that assists in ligation of ubiquitin. Ubiquitin in turn serves diverse functions such as signaling of degradation by the proteasome, intracellular transport, protein interactions, and protein activity [108]. No known relations to neither CCR6 or THBS2 were found in published literature.

Discussion and further work

6.1 CSD-CS

6.1.1 Code parallelization

The current parallelization schemes successfully reduces the wall-time of the program, allowing for fast construction of differential co-expression networks using the CSD-framework. As seen from figure 4.5 however there is room for increased efficiency by reducing the constant time contribution from serial parts of the program. Steps 6 and 7 listed in section 3.1, determining the importance thresholds and calculating the adjacency matrix respectively, are examples of unexplored program sections in terms of parallelization. The first one should be possible to parallelize by exchanging the outer while-loop with a for-loop and introducing OpenMP reduction variables in the parallel domain. Step 7 should be possible to parallelize the same way that the correlations calculations were parallelized; by cyclic division of the outer for-loop that iterates over the index i in the adjacency matrix A_{ij} . Due to a lack of time these changes are left for further work.

The program have currently been tested on two different computer systems. On the one used to generate the results presented in chapter 5, the parallelization behaved exactly as expected. On the other system however, a dysfunction occurred. The correlation and variance calculations are repeated for all input-files in a sequential manner. This is done is through a for-loop that iterates over all the list of input-files and calls the function calculating the correlation and variance for each one. For some yet unidentified reason, the parallelization within the correlation/variance-function fails for the first input file on this other system. For the second input-file however, it appears to function as expected. This matter require further investigation, but as the second computer-system is a fairly new high-performance system and recently assembled, it is possible that the problem lies in the assembly of the system and not in the code itself. For the older and more thoroughly tested computer system the code performed as expected.

6.1.2 Memory reduction

For increased program portability, further reduction in memory consumption should be performed, as the program still requires around 28 GB RAM to evaluate data sets containing the approximately 20 000 protein coding genes in the human genome. A peak memory consumption below 16 GB is an ideal first target, as this is the memory-capacity of many modern desktop systems.

A first step towards limiting the memory consumption has been performed, exchanging the datatype of the similarity and adjacency matrices from *double* to *float*. As these matrices constitutes the majority of memory requirement, this step was expected to effectively reduce memory consumption by 50%. The program-version using *double* as data type required approximately 35 GB of memory for two datasets containing 20 000 genes, while the current version uses 28 GB. Thus the actual observed memory reduction from this step was about half of that expected.

An attempt at relating the peak memory consumption to size of the input data was performed. For two input files, the maximum memory usage was estimated to be approximately $7 \cdot (n \times n)$ multiplied by the size of the data-type of the matrices (2 matrices for correlation values, 2 matrices for variance values, and 3 matrices for CSD-scores; $2 + 2 + 3 = 7$). For input data of size $n = 2 \cdot 10^4$, and using *float* as the data-type, the estimated maximum memory usage, should be 11.2 GB. Although this is a slight underestimate, the fact that the program peaks at approximately 28 GB indicates that the program does not behave entirely as intended in terms of memory management. The memory peak appears towards the end of the program execution, later than assumed in the estimation above. This gives rise to a couple of hypotheses as to where the problem lies. One possible explanation is that de-allocation of some matrix or matrices is forgotten or somehow fails, resulting in a longer than intended lifetime for some of the data. Another possibility is that uncautious implementation of one of the latter procedures of the program causes the program to consume more memory than intended. A systematic analysis of the latter program sections and evaluation of their memory consumption is left for further work. Finding the cause of the memory leak, and reducing the peak memory consumption is considered high priority for future program versions, as it will increase the software portability.

Another possibility for memory reduction, is only storing half of the elements in the similarity and adjacency matrices. As the networks generated using the CSD-framework are undirected, the similarity and adjacency matrices are symmetric across the diagonal. This means that all information is conserved upon discarding one of the matrix halves. The current version stores the entire matrices as this was easier to implement, and getting a reliable, working version of the program was more important than the actual performance of the program for this first version. As all matrices of size $n \times n$ in the program share this property, and the peak memory use of the program is dominated by the contribution from these matrices, this has the potential to reduce the memory consumption with 50%.

Successful identification of the memory leak, and removing half of the matrices as described above, have the potential of reducing the memory consumption of the program to around 5.6 GB for data-sets containing 20 000 genes. This is well within the target of 16 GB. It is even below the next target of 8 GB, which will make the program feasible for most mid-price range laptops. This makes the program highly portable, and opens up for analyzing expression data for full probe-sets of $n \approx 50000$. This will however still be

reserved for high-performance systems, as it will require ≈ 70 GB of memory.

6.1.3 Increased functionality

Preprocessing algorithms

The current version of CSD-CS only contains a preprocessing algorithm for gene-expression data as input. This is somewhat limiting for the functionality of the program. Ideally it should be possible to run the program using gene-expression data, correlation data or CSD-scores, and return anything from correlation data to a finished network. In order for this to be fully supported, it is necessary to implement preprocessing algorithms for correlation data and CSD-scores. This is so that the user will be able to calculate the correlation values or CSD-scores for data-sets independently of each other, and then feed the results into the program at a later stage. The current version has no preprocessing algorithms for these input types however, requiring the input files to consist of identical and sorted sets of genes in order to for the similarity and adjacency matrices to match. This property is guaranteed if the the relevant data sets are fed into the program together as gene-expression data. If not, the user need to manipulate the data sets him-/herself, which is cumbersome and error-prone, and thus should be avoided. Implementation of these additional preprocessing algorithms is left as further work.

Correlation measures

The choice of similarity measure will affect the output of the method. Table 6.1 illustrates this, by showing a comparison between 5 different similarity measures for a set of 4 randomly chosen genes from the rheumatoid arthritis data set. It might be of interest to generate CSD-networks using different similarity measures, as a potential extension to the method itself, and the software. The current program version uses Spearman's rank correlation, and have an implementation of Pearson's correlation, currently unused. Other interesting prospects are for instance Kendall's tau [109], Biweight mid-correlation [51] or mutual information [69, 110]. The program is designed to accommodate integration of other similarity measures, despite only supporting Spearman correlation at the moment. Implementation and integration of other similarity measures is left as further work.

Table 6.1: A comparison of different similarity scores between 4 randomly chosen genes (A , B , C and D) in the rheumatoid arthritis data set. The binning of the values for calculation of MI-scores was chosen to generate scores in the same size range as the rest of the similarity measures (for the purpose of comparison).

	A-B	A-C	A-D	B-C	B-D	C-D
Pearson	0.1285	-0.0069	0.1179	0.1571	0.4427	0.1682
Spearman	0.1084	-0.0184	0.0985	0.1507	0.4301	0.1508
Kendall's tau	0.0720	-0.0146	0.0661	0.1023	0.2962	0.0998
Biweight midcorrelation	0.1099	-0.0020	0.1030	0.1492	0.4390	0.1609
Mutual information	0.3056	0.2205	0.2463	0.1572	0.1849	0.1066

Multiple files as input

Although not discussed in this thesis, the CSD-framework is generalizable to an arbitrary number of data sets. This has been taken into account when writing the program, and so it accommodates an arbitrary number of input files. The current version simply generates a network for each combination of the input files, resulting in a rapid increase in memory and time consumption when increasing the number of files. The genes in the networks are filtered by the preprocessing algorithm so that only the genes/probes occurring in all data sets are included. Further development of this feature, including specifying how to relate the resulting networks, alternatively calculating higher dimension CSD-scores, is left for further work.

Graphical user interface

Although the software is fairly simple to use, the long list of input parameters requires the user to have good knowledge about the functionality of the software. As some users might feel more comfortable using a graphical user interface (GUI), and program portability and ease of use is in focus, an idea for further work might be to develop a GUI. This will make it easier for users to identify which parameter options are available and, and the program can display all default values. As the program functionality is limited, implementing a GUI should not be a demanding task, and can be considered for future program versions.

Support for other operating systems

The current version of the software is developed solely for linux-based operating systems. For program portability, support for other operating systems, most noticeably Windows and OS X, would be useful extensions. Facilitating use with these environments is left as further work.

6.2 Network analysis

RA is a highly heterogeneous disease, with multiple genetic associations and environmental factors playing a role in the disease development. This makes it unlikely to identify a clear disease signal in the network, as the underlying processes are complex. As RA is an autoimmune disease, one would expect to detect a change in immune response between the healthy and infected cohorts. Especially the S- and D-interaction networks are of interest in this respect. The S-network showed highest enrichment for blood clotting and coagulation related biology. It could therefore be interesting to evaluate how the observed coagulation-signal relates to the disease state, and whether these genes are more or less co-expressed in the disease dataset. Both the C- and the D-network, showed some enrichment for immune system related biology, but fairly low scores. When performing enrichment on the two major components, a slight enrichment of T-cell related signaling was observed in the S-component. The module level enrichment supported this last observation by showing fairly high enrichment scores for T-cell receptor complex and binding in module 3. Module 3 contains only 12 C-interactions, and 4 D-interactions, while consisting of 168 S-interactions. This is thus a strong indication of a change in immune regulation captured

by this module. Further analysis on the localization and interactions of the relevant genes should be performed in order to verify this. Whether the apparent change in co-expression is linked to a general inflammatory response or possible mechanisms behind RA needs to be evaluated in future projects as well.

Module 6 showed moderate enrichment for apoptotic signaling, abnormal NK-cell physiology, and increased T-cell proliferation. Module 6 does however consist of a large majority of C-type interactions, which makes it unlikely that the observed enrichment is related to changes in immune system response. It is however interesting to observe the proximity of module 3 and 6 given their related biology, supporting the hypothesis that network structure is related to biological function.

Furthermore module 4 showed high ontological enrichment for helper T-lymphocytes and cytotoxic T-lymphocytes, and module 10 for IgA immunoglobulin complex and IgA B-cell receptor complex. As module 4 consists exclusively of C-interactions it is not of great interest to the disease state. Module 10 however contains 11 C-type and 26 D-type interactions, and are thus an interesting candidate for RA related biology. The fact that CD5 is located in this module, and its neighboring genes are related to RA-biology, makes this module an interesting prospect for further analysis. As module 10 is fairly sparse, an interesting analysis would be to generate a new CSD-network with a lower importance threshold. New genes and new interactions would then likely occur in and around module 10, possibly conveying additional RA-related biology.

An interesting finding from the enrichment analysis was that PRDM1-transcription related enrichment was found for the D-network. PRDM1 (also known as BLIMP1; B lymphocyte induced maturation protein 1) is a transcription factor important in antigen-dependent activation of both T-cells and B-cells (although it plays different roles in the two cell types) [111]. PRDM1 have been associated with RA [87], and although the gene itself is not present in the network it could be interesting to evaluate the location and co-expression pattern of the related genes. For completeness, and to facilitate further analysis, a list of the PRDM1-transcription related genes have been added in appendix F.

The detected TYK2-knockdown enrichment was also an interesting result from the D-network enrichment analysis. TYK2 (Tyrosine Kinase 2) codes for a protein that activates cytokine signals by phosphorylating subunits of cytokine receptors. The protein is also involved in type 1 and 3 interferon signaling, and is suspected to protect against RA and autoimmune disease [88]. This finding may indicate a change in TYK2 expression or similar changes in expression patterns for RA patients. A similar analysis as described in the paragraph above could be of interest here, and so the relevant genes are listed in appendix G and the analysis itself left for further work.

Other procedures that can be interesting for further network analysis include; reducing importance level and retry guilt-by-association, investigating relations between the CSD-network and PPI-networks¹, and identification and analysis of nodes involved in a lot of S- and D-interactions. In addition the genes CD72 and FOXD1 was found in the network, both of which play regulatory roles in relation to autoimmunity [93, 92, 95]. Investigating their locations and co-expression patterns may yield interesting information. Further analysis of modules with high immunological enrichment such as module 6 may also be of interest.

¹Protein-protein interaction network

Although the results commented on so far are mainly those with probable connections to the disease state, other nodes, such as those identified in section 5.5.2 are all of interest. The fact that the CSD-scores are based on both correlation scores and variance weighting, ensures that only gene-pairs showing predictable and tight co-expression across all the measurements in a data-set are highlighted by the method. The highlighted genes are thus likely to be related to central biological processes, or cellular responses. The S- and D-interactions in a network will therefore effectively capture responses that may explain phenotypic differences. A CSD-network thus provides a good tool for generating hypotheses about the genetics behind phenotypic differences, and may identify new genes of interest.

Conclusion

This thesis have successfully generated a new software-tool for the generation of differential co-expression networks using the CSD-framework. An example of a co-expression network analysis have also been provided, on a network generated from rheumatoid arthritis expression data. The analysis identified RA-related biology in the network, such as immune system and inflammation related biology. Some RA-associated genes were found in the network, while others were implied through enrichment-analysis. Several prospective genes for further analysis, including CD72, FOXD1, AP8B2 and HAPLN4 were identified. The findings from this preliminary analysis give some clues as to where disease-related information may be localized in the network, and provide a solid basis for further analysis.

Although the target of the CSD-method and the illustrated application was on gene expression data, the methodology lends itself to a much broader set of problems. By using a network formalism to visualize complex correlation patterns in one or more data-sets, a researcher can apply all the tools of network science in probing the underlying structure of the data. The biological analysis performed illustrates how case-specific knowledge can be applied to further refine the analysis.

Bibliography

- [1] A Voigt, K Nowick, and E Almaas. A composite network of conserved and tissue specific gene interactions reveals possible genetic interactions in glioma. *submitted*.
- [2] A-L Barabási. *Network Science Book*. Cambridge University Press, 2015.
- [3] P Erdős and a Rényi. On random graphs. *Publicationes Mathematicae*, 6:290–297, 1959.
- [4] A-L Barabási and E Bonabeau. Scale-free networks. *Scientific American*, 3(1):50–59, 2003.
- [5] M Kivela. Multilayer Networks Library for Python, http://people.maths.ox.ac.uk/kivela/mln_library/visualizing.html, Accessed: 2017-08-01.
- [6] A-L Barabási. The network takeover. *Nature Physics*, 8(1):14–16, 2011.
- [7] R Albert and A-L Barabasi. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74(1):47–97, 2002.
- [8] Y Y Liu and A-L Barabási. Control principles of complex systems. *Reviews of Modern Physics*, 88(3), 2016.
- [9] B F Zhan. Three fastest shortest path algorithms on real road networks: Data structures and procedures. *Journal of Geographic Information and Decision Analysis*, 1(1):70–82, 1997.
- [10] J Leskovec, K J Lang, and M Mahoney. Empirical comparison of algorithms for network community detection. *Conference on World Wide Web {WWW}*, pages 631–640, 2010.
- [11] S Wasserman and K Faust. *Social Network Analysis: Methods and Applications*, 1994.
- [12] H Jeong, B Tombor, R Albert, Z N Oltvai, and A-L Barabasi. The large-scale organization of metabolic networks. *Nature*, 407(6804):651–654, 2000.

-
- [13] J Schwender. *Plant Metabolic Networks*. 2009.
- [14] B H Liu, H Yu, K Tu, C Li, Y X Li, and Y Y Li. DCGL: An R package for identifying differentially coexpressed genes and links from gene expression microarray data. *Bioinformatics*, 26(20):2637–2638, 2010.
- [15] D Amar, H Safer, and R Shamir. Dissection of Regulatory Networks that Are Altered in Disease via Differential Co-expression. *PLoS Computational Biology*, 9(3), 2013.
- [16] D Kostka and R Spang. Finding disease specific alterations in the co-expression of genes. In *Bioinformatics*, volume 20, 2004.
- [17] E Pierson, D Koller, A Battle, and S Mostafavi. Sharing and Specificity of Co-expression Networks across 35 Human Tissues. *PLoS Computational Biology*, 11(5), 2015.
- [18] Z Xue, K Huang, C Cai, L Cai, C Y Jiang, Y Feng, Z Liu, Q Zeng, L Cheng, Y E Sun, and Et Al. Genetic programs in human and mouse early embryos revealed by single-cell RNA sequencing. *Nature*, 500(7464):593–597, 2013.
- [19] S van Dam, U Vösa, A van der Graaf, L Franke, and J P de Magalhães. Gene co-expression analysis for functional classification and genedisease predictions. *Briefings in Bioinformatics*, 2017.
- [20] B Zhang and S Horvath. A General Framework for Weighted Gene Co-Expression Network Analysis A General Framework for Weighted Gene. *Statistical applications in genetics and molecular biology*, 4(1), 2005.
- [21] M E J Newman. The structure and function of complex networks. *Siam Review*, 45(2):167–256, 2003.
- [22] R Albert, H Jeong, and A-L Barabasi. Diameter of the World Wide Web. *Nature*, 401:130–131, 1999.
- [23] M Faloutsos, P Faloutsos, and C Faloutsos. On power-law relationships of the Internet topology. *ACM SIGCOMM Computer Communication Review*, 29(4):251–262, 1999.
- [24] R Albert, I Albert, and G L Nakarado. Structural vulnerability of the North American power grid. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 69(2 2), 2004.
- [25] H Jeong, S P Mason, A-L Barabási, and Z N Oltvai. Lethality and centrality in protein networks. *Nature*, 411(6833):41–42, 2001.
- [26] S Roy, D K Bhattacharyya, and J K Kalita. Reconstruction of gene co-expression network from microarray data using local expression patterns. *BMC bioinformatics*, 15(7):S10, 2014.

-
- [27] K-I Goh, M E Cusick, D Valle, B Childs, M Vidal, and A-L Barabási. The human disease network. *Proceedings of the National Academy of Sciences of the United States of America*, 104(21):8685–8690, 2007.
- [28] M E J Newman. Spread of epidemic disease on networks. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 66(1), 2002.
- [29] R Albert, H Jeong, and A-L Barabási. Error and attack tolerance of complex networks. *Nature*, 406: 378482, 2000. *Nature*, 406(6794):378–382, 2000.
- [30] A E Motter. Cascade control and defense in complex networks. *Physical Review Letters*, 93(9), 2004.
- [31] S Dorogovtsev, A Goltsev, and J Mendes. Critical phenomena in complex networks. *Reviews of Modern Physics*, 80(4):1275–1335, 2008.
- [32] A Arenas, A Diaz-Guilera, J Kurths, Y Moreno, and C Zhou. Synchronization in complex networks, 2008.
- [33] R Olfati-Saber, J A Fax, and R M Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, 2007.
- [34] E N Gilbert. Random Graphs. *The Annals of Mathematical Statistics*, 30(4):1141–1144, 1959.
- [35] A-L Barabasi and R Albert. Emergence of Scalling in Random Networks. *Science*, 286(5439):509–512, 1999.
- [36] V D Blondel, J-L Guillaume, R Lambiotte, and E Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, (10):10008, 2008.
- [37] G Bianconi and A-L Barabási. Competition and multiscaling in evolving networks. *Europhysics letters*, 54(4):436–442, 2001.
- [38] X He and J Zhang. Why do hubs tend to be essential in protein networks? *PLoS Genetics*, 2(6):826–834, 2006.
- [39] T G Lewis. Network science - theory and applications. chapter 13. Wiley, 2009.
- [40] D M Wolf, M E Lenburg, C Yau, A Boudreau, and L J Van't Veer. Gene co-expression modules as clinically relevant hallmarks of breast cancer diversity. *PLoS ONE*, 9(2), 2014.
- [41] G J Tawa, M D M AbdulHameed, X Yu, K Kumar, D L Ippolito, J A Lewis, J D Stallings, and A Wallqvist. Characterization of chemically induced liver injuries using gene co-expression modules. *PLoS ONE*, 9(9), 2014.
- [42] J Jiang, P Jia, Z Zhao, and B Shen. Key regulators in prostate cancer identified by co-expression module analysis. *BMC genomics*, 15(1):1015, 2014.
-

-
- [43] B Cai, C-H Li, and J Huang. Systematic identification of cell-wall related genes in Populus based on analysis of functional modules in co-expression network. *PLoS one*, 9(4), 2014.
- [44] U Brandes. A faster algorithm for betweenness centrality. *Ulrik Brandes The Journal of Mathematical Sociology*, 25(252):163–177, 2001.
- [45] M E J Newman. A measure of betweenness centrality based on random walks. *Social Networks*, 27(1):15, 2003.
- [46] U Stelzl, U Worm, M Lalowski, C Haenig, F H Brembeck, H Goehler, M Stroedicke, M Zenkner, A Schoenherr, S Koeppen, and Et Al. A human protein-protein interaction network: A resource for annotating the proteome. *Cell*, 122(6):957–968, 2005.
- [47] J Dong and S Horvath. Understanding network concepts in modules. *BMC Systems Biology*, 1(1752-0509):24, 2007.
- [48] U Ala, R M Piro, E Grassi, C Damasco, L Silengo, M Oti, P Provero, and F Di Cunto. Prediction of human disease genes by human-mouse conserved coexpression analysis. *PLoS Computational Biology*, 4(3), 2008.
- [49] J Nie, R Stewart, H Zhang, J A Thomson, F Ruan, X Cui, and H Wei. TF-Cluster: a pipeline for identifying functionally coordinated transcription factors via network decomposition of the shared coexpression connectivity matrix (SCCM). *BMC systems biology*, 5(1):53, 2011.
- [50] C-H Zheng, L Yuan, W Sha, and Z-L Sun. Gene differential coexpression analysis based on biweight correlation and maximum clique. *BMC bioinformatics*, 15 Suppl 1(Suppl 15):S3, 2014.
- [51] L Song, P Langfelder, and S Horvath. Comparison of co-expression measures: mutual information, correlation, and model based indices. *BMC Bioinformatics*, 13(1):328, 2012.
- [52] S Persson, H Wei, J Milne, G P Page, and C R Somerville. Identification of genes required for cellulose synthesis by regression analysis of public microarray data sets. *Proceedings of the National Academy of Sciences of the United States of America*, 102(24):8633–8638, 2005.
- [53] M Schena. *Microarray analysis*. John Wiley and Sons Inc., 1st editio edition, 2003.
- [54] I B McInnes and G Schett. The Pathogenesis of Rheumatoid Arthritis. *New England Journal of Medicine*, 365(23):2205–2219, 2011.
- [55] A J MacGregor, H Snieder, A S Rigby, M Koskenvuo, J Kaprio, K Aho, and A J Silman. Characterizing the quantitative genetic contribution to rheumatoid arthritis using data from twins. *Arthritis & Rheumatism*, 43(1):30–37, 2000.

-
- [56] D Van Der Woude, J J Houwing-Duistermaat, R E M Toes, T W J Huizinga, W Thomson, J Worthington, A H M Van Der Helm-Van Mil, and R R P De Vries. Quantitative heritability of anti-citrullinated protein antibody-positive and anti-citrullinated protein antibody-negative rheumatoid arthritis. *Arthritis and Rheumatism*, 60(4):916–923, 2009.
- [57] C M Deighton, D J Walker, I D Griffiths, and D F Roberts. The contribution of HLA to rheumatoid arthritis. *Clinical genetics*, 36(3):178–182, 1989.
- [58] J Kurkó, T Besenyei, J Laki, T T Glant, K Mikecz, and Z Szekanecz. Genetics of rheumatoid arthritis - A comprehensive review, 2013.
- [59] E A Stahl, S Raychaudhuri, El F Remmers, G Xie, S Eyre, B P Thomson, Y Li, F A S Kurreeman, A Zhernakova, A Hinks, and Et Al. Genome-wide association study meta-analysis identifies seven new rheumatoid arthritis risk loci. *Nature genetics*, 42(6):508–514, 2010.
- [60] S Viatte, D Plant, and S Raychaudhuri. Genetics and epigenetics of rheumatoid arthritis. *Nature reviews. Rheumatology*, 9(3):141–53, 2013.
- [61] S Eyre, J Bowes, D Diogo, A Lee, A Barton, P Martin, A Zhernakova, E Stahl, S Viatte, K McAllister, and Et Al. High-density genetic mapping identifies new susceptibility loci for rheumatoid arthritis. *Nature genetics*, 44(12):1336–1340, 2012.
- [62] S Raychaudhuri, C Sandor, E A Stahl, J Freudenberg, H-S Lee, X Jia, L Alfredsson, L Padyukov, L Klareskog, J Worthington, and Et Al. Five amino acids in three HLA proteins explain most of the association between MHC and seropositive rheumatoid arthritis. *Nature genetics*, 44(3):291–296, 2012.
- [63] D Diogo, F Kurreeman, E A Stahl, K P Liao, N Gupta, J D Greenberg, M A Rivas, B Hickey, J Flannick, B Thomson, and Et Al. Rare, Low-Frequency, and Common Variants in the Protein-Coding Sequence of Biological Candidate Genes from GWASs Contribute to Risk of Rheumatoid Arthritis. *American Journal of Human Genetics*, 92(1):15–27, 2013.
- [64] MD) McKusick-Nathans Institute of Genetic Medicine, Johns Hopkins University (Baltimore. Online Mendelian Inheritance in Man, OMIM®.
- [65] L Carmona, M Cross, B Williams, M Lassere, and L March. Rheumatoid arthritis. *Best practice & research. Clinical rheumatology*, 24(6):733–45, 2010.
- [66] T H Cormen, C C E Leiserson, R R L Rivest, and C Stein. *Introduction to Algorithms, Third Edition*, volume 7. The MIT Press, 2009.
- [67] M G Kendall. Biometrika Trust A New Measure of Rank Correlation. *Source: Biometrika*, 30(12):81–93, 1938.
- [68] T M Cover and J A Thomas. *Elements of Information Theory 2nd Edition*. 2006.
-

-
- [69] C O Daub, R Steuer, J Selbig, S Kloska, M Schena, D Shalon, R W Davis, P O Brown, V E Velculescu, I Zhang, and Et Al. Estimating mutual information using B-spline functions an improved similarity measure for analysing gene expression data. *BMC Bioinformatics*, 5(1):118, 2004.
- [70] R. A. Fisher. *Statistical methods for research workers*. Number V. 1934.
- [71] O J Dunn. Estimation of the Medians for Dependent Variables. *The Annals of Mathematical Statistics*, 30:192–197, 1959.
- [72] Y Benjamini and Y Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing, 1995.
- [73] Y Benjamini and D Yekutieli. The control of the false discovery rate in multiple testing under dependency. *Annals of Statistics*, 29(4):1165–1188, 2001.
- [74] P Shannon, A Markiel, O Ozier, N S Baliga, J T Wang, D Ramage, N Amin, B Schwikowski, and T Ideker. Cytoscape: A software Environment for integrated models of biomolecular interaction networks. *Genome Research*, 13(11):2498–2504, 2003.
- [75] N T Doncheva, Y Assenov, F S Domingues, and M Albrecht. Topological analysis and interactive visualization of biological networks and protein structures. *Nature Protocols*, 7(4):670–685, 2012.
- [76] S J Russell and P Norvig. *Artificial intelligence: a modern approach (Global Edition)*. Pearson Education Ltd., 3rd edition, 2016.
- [77] International Human Genome Sequencing Consortium. Finishing the euchromatic sequence of the human genome. TL - 431. *Nature*, 431(7011):931–945, 2004.
- [78] OpenMP - FAQ, <http://www.openmp.org/about/openmp-faq/>, Accessed: 2017-08-02.
- [79] OpenMP - About us, <http://www.openmp.org/about/about-us/>, Accessed: 2017-08-02.
- [80] B Chapman, G Jost, and R Van Der Pas. *Using OpenMP: Portable Shared Memory Parallel Programming*, volume 10. 2008.
- [81] P Pacheco. *An Introduction to Parallel Programming*. 2011.
- [82] A M Walsh, J W Whitaker, C C Huang, Y Cherkas, S L Lamberth, C Brodmerkel, M E Curran, R Dobrin, I B McInnes, G Schett, and Et Al. Integrative genomic deconvolution of rheumatoid arthritis GWAS loci into gene and cell type associations. *Genome Biology*, 17(1):79, 2016.
- [83] J Lonsdale, J Thomas, M Salvatore, R Phillips, E Lo, S Shad, R Hasz, G Walters, F Garcia, N Young, and Et Al. The Genotype-Tissue Expression (GTEx) project. *Nature genetics*, 45(6):580–585, 2013.

-
- [84] E Y Chen, C M Tan, Y Kou, Q Duan, Z Wang, G V Meirelles, N R Clark, and A Ma'ayan. Enrichr: interactive and collaborative HTML5 gene list enrichment analysis tool. *BMC bioinformatics*, 14(1):128, 2013.
- [85] M V Kuleshov, M R Jones, A D Rouillard, N F Fernandez, Q Duan, Z Wang, S Koplev, S L Jenkins, K M Jagodnik, A Lachmann, and Et Al. Enrichr: a comprehensive gene set enrichment analysis web server 2016 update. *Nucleic acids research*, 44(1):90–97, 2016.
- [86] M Rebhan, V Chalifa-Caspi, J Prilusky, and D Lancet. GeneCards: A novel functional genomics compendium with automated data mining and query reformulation support. *Bioinformatics*, 14(8):656–664, 1998.
- [87] S Raychaudhuri, B P Thomson, E F Remmers, S Eyre, and A Hinks. Genetic variants at CD28, PRDM1 and CD2/CD58 are associated with rheumatoid arthritis risk. *Nature Genetics*, 41(12):1313–1318, 2009.
- [88] D Diogo, L Bastarache, K P Liao, R R Graham, R S Fulton, J D Greenberg, S Eyre, J Bowes, J Cui, A Lee, and Et Al. TYK2 protein-coding variants protect against rheumatoid arthritis and autoimmunity, with no evidence of major pleiotropic effects on non-autoimmune complex traits. *PLoS ONE*, 10(4), 2015.
- [89] A P Cope, H Schulze-Koops, and M Aringer. The central role of T cells in rheumatoid arthritis. *Clinical and experimental rheumatology*, 25(5):S4–11, 2007.
- [90] H J Wu, C Venkataraman, S Estus, C Dong, R J Davis, R a Flavell, and S Bonada. Positive signaling through CD72 induces mitogen-activated protein kinase activation and synergizes with B cell receptor signals to induce X-linked immunodeficiency B cell proliferation. *Journal of immunology (Baltimore, Md. : 1950)*, 167:1263–1273, 2001.
- [91] X Jiang, N K Björkström, and E Melum. Intact CD100-CD72 Interaction Necessary for TCR-Induced T Cell Proliferation. *Frontiers in immunology*, 8:765, 2017.
- [92] C Akatsu, K Shinagawa, N Numoto, Z Liu, A Konuskan Ucar, M Aslam, S Phoon, T Adachi, K Furukawa, N Ito, and T Tsubata. CD72 negatively regulates B lymphocyte responses to the lupus-related endogenous toll-like receptor 7 ligand Sm/RNP. *Journal of experimental medicine*, 213:2691–2706, 2016.
- [93] T Tsubata. CD22 und CD72 sind in BLymphozyten dominant exprimierte inhibitorische Rezeptoren und regulieren systemische Autoimmunerkrankungen. *Zeitschrift für Rheumatologie*, 75(1):86–89, 2016.
- [94] M Lyu, Y Hao, Y Li, C Lyu, W Liu, H Li, F Xue, X Liu, and R Yang. Upregulation of CD72 expression on CD19+ CD27+ memory B cells by CD40L in primary immune thrombocytopenia. *British journal of haematology*, 2017.
- [95] L Lin and S L Peng. Coordination of NF- κ B and NFAT antagonism by the forkhead transcription factor Foxd1. *Journal of immunology (Baltimore, Md. : 1950)*, 176:4793–4803, 2006.
-

-
- [96] A Golla, A Jansson, J Ramser, H Hellebrand, R Zahn, T Meitinger, B H Belohradsky, and A Meindl. Chronic recurrent multifocal osteomyelitis (CRMO): evidence for a susceptibility gene located on chromosome 18q21.3-18q22. *European Journal of Human Genetics*, 10(3):217–221, 2002.
- [97] D Aygun, K Barut, and Y Camcioglu. Chronic recurrent multifocal osteomyelitis: A rare skeletal disorder, 2015.
- [98] Gene Ontology Consortium. The Gene Ontology Consortium. Gene ontology: tool for the unification of biology. *Nature Genetics*, 25(1):25–29, 2000.
- [99] N Nagy, G Kaber, P Y Johnson, J A Gebe, A Preisinger, B A Falk, V G Sunkari, M D Gooden, R B Vernon, M Bogdani, and Et Al. Inhibition of hyaluronan synthesis restores immune tolerance during autoimmune insulinitis. *Journal of Clinical Investigation*, 125(10):3928–3940, 2015.
- [100] A M Mueller, B H Yoon, and S A Sadiq. Inhibition of hyaluronan synthesis protects against central nervous system (CNS) autoimmunity and increases CXCL12 expression in the inflamed CNS. *Journal of Biological Chemistry*, 289(33):22888–22899, 2014.
- [101] H F Kuipers, M Rieck, I Gurevich, N Nagy, M J Butte, R S Negrin, T N Wight, L Steinman, and P L Bollyky. Hyaluronan synthesis is necessary for autoreactive T-cell trafficking, activation, and Th1 polarization. *PNAS*, 113(5):1339–1344, 2016.
- [102] X Chang, R Yamada, and K Yamamoto. Inhibition of antithrombin by hyaluronic acid may be involved in the pathogenesis of rheumatoid arthritis. *Arthritis research & therapy*, 7(2):268–273, 2005.
- [103] Y Yoshioka, E Kozawa, H Urakawa, E Arai, N Futamura, L Zhuo, K Kimata, N Ishiguro, and Y Nishida. Suppression of hyaluronan synthesis alleviates inflammatory responses in murine arthritis and in human rheumatoid synovial fibroblasts. *Arthritis and Rheumatism*, 65(5):1160–1170, 2013.
- [104] C-L Chou, H-W Li, S-H Lee, K-L Tsai, and H-Y Ling. Effect of intra-articular injection of hyaluronic acid in rheumatoid arthritis patients with knee osteoarthritis. *Journal of the Chinese Medical Association : JCMA*, 71(8):411–415, 2008.
- [105] K Masuko, M Murata, K Yudoh, T Kato, and H Nakamura. Anti-inflammatory effects of hyaluronan in arthritis therapy: Not just for viscosity, 2009.
- [106] A Aruffo, I Stamenkovic, M Melnick, C B Underhill, and B Seed. CD44 is the principal cell surface receptor for hyaluronate. *Cell*, 61(7):1303–1313, 1990.
- [107] S Misra, V C Hascall, R R Markwald, and S Ghatak. Interactions between Hyaluronan and Its Receptors (CD44, RHAMM) Regulate the Activities of Inflammation and Cancer. *Frontiers in Immunology*, 6(May):201, 2015.
- [108] O Kerscher, R Felberbaum, and M Hochstrasser. Modification of proteins by ubiquitin and ubiquitin-like proteins. *Annual review of cell and developmental biology*, 22:159–180, 2006.

-
- [109] S Kumari, J Nie, H Chen, H Ma, R Stewart, X Li, M-Z Lu, W M Taylor, and H Wei. Evaluation of gene association methods for coexpression network construction and biological knowledge discovery. *PloS one*, 7(11):e50411, 2012.
- [110] I Priness, O Maimon, and I Ben-Gal. Evaluation of gene-expression clustering via mutual information distance measure. *BMC bioinformatics*, 8:111, 2007.
- [111] K Calame. Activation-dependent induction of Blimp-1, 2008.

Appendix A

Table 1: This is a list of 52 genes with suspected association to rheumatoid arthritis.

AFF3	FCGR3A	IRAK1	RBPJ
ARID5B	FCGR3B	IRF5	RCAN1
C5	GATA3	IRF8	REL
CCL21	HLA-B	KIF5A	RUNX1
CCR6	HLA-DPB1	PADI4	SPRED2
CD2	HLA-DRB1	PIP4K2C	STAT4
CD5	IKZF4	POU3F1	TAGAP
CD28	IL2	PRDM1	TLE3
CD40	IL2RA	PRKCQ	TNFAIP3
CD58	IL2RB	PTPN22	TNFRSF14
CTLA4	IL6R	PTPRC	TRAF1
FCGR2A	IL6ST	PYK	TRAF6
FCGR2B	IL21	RASGRP1	TYK2

Appendix B

Figure 1 Example illustrating the *ged* file-format.

///	M0	M1	M2	M3	M4	M5	M6	M7	M8
G0	67.7018	62.2004	32.9133	2.30199	51.0603	2.6937	102.451	85.8533	101.881
G1	13.1174	67.3498	98.9909	43.6737	108.145	0.642852		104.724	101.838
G2	64.7056	29.3808	23.3282	60.8735	108.213	101.479	32.7166	104.028	27.6895
G3	87.1239	35.2746	16.2567	27.3469	83.0659	102.605	52.5045	10.3039	79.2695
G4	2.87402	99.4195	26.5149	98.4131	51.7014	57.1076	31.0701	4.31542	106.643
G5	57.3801	102.911	99.1284	43.0768	68.5062	87.1031	20.7713	47.0499	98.8018
G6	98.3397	46.1126	12.067	60.3223	59.8552	77.9779	90.1432	69.0127	81.0512
G7	57.6359	57.3377	54.6405	30.0818	2.26597	92.0216	102.532	24.4968	75.6608
G8	101.163	77.028	19.396	31.7934	79.1143	40.8562	69.8342	28.8057	24.7176
G9	41.5719	100.061	83.1148	85.0419	84.2186	84.4583	40.8372	33.9304	13.2982
G10	43.402	21.7304	108.846	40.4912	36.8702	68.3048	3.98987	38.7479	82.1542
G11	106.049	91.0002	73.4376	109.591	49.3091	59.5614	85.2954	13.7365	93.158
G12	78.6624	100.258	1.95172	83.2465	65.6916	12.2372	87.079	53.8619	48.1316
G13	49.7853	41.2435	65.4334	53.2643	48.0484	61.3567	18.3598	92.1212	104.568
G14	54.831	88.7325	29.7668	30.323	28.4679	66.2466	72.945	105.587	26.4932
G15	109.639	49.4268	12.575	11.3498	48.8605	14.5198	12.3157	45.6368	31.4198
G16	73.5359	1.88459	24.178	105.042	13.015	71.4492	77.1969	89.3507	0.22101
G17	40.3803	84.1218	88.0423	6.79967	71.2877	3.82474	89.4264	83.4129	41.3008
G18	107.569	20.0771	39.5704	32.5524	69.0868	27.2393	17.5384	69.5722	1.11742
G19	73.8079	19.3502	28.9105	51.2521	106.002	107.608	80.4303	86.9533	62.0848
G20	84.7008	46.0942	73.5407	84.6468	66.7856	39.3257	79.708	27.766	72.7424
G21	90.2464	48.5196	55.3863	105.939	82.3747	11.5664	71.5592	12.8517	46.7498
G22	80.5488	67.7575	9.41205	73.9852	1.97837	88.7037	59.4997	2.21028	79.3802
G23	110.193	3.25086	34.8562	82.4895	87.9107	12.5862	2.14054	85.8237	13.5115
G24	4.45818	12.8949	101.968	59.4098	47.3321	88.7989	32.4914	41.2138	31.4351
G25	91.8096	58.1497	64.8269	50.6438	33.2802	26.899	82.2198	32.6136	98.4228
G26	79.6672	95.5717	54.2331	18.8283	91.6732	35.2218	24.6481	103.911	20.1083
G27	38.665	50.3596	87.806	94.1607	85.4788	31.2111	64.435	109.428	89.4062
G28	47.659	96.7948	25.4605	86.9627	52.7051	79.4969	60.9706	42.0871	19.0372
G29	59.304	35.3124	88.9712	43.2307	74.0305	69.133	107.789	104.628	6.95037
G30	42.4342	96.3248	68.0602	57.5007	76.5697	50.6305	14.8208	52.7512	108.09

Figure 2 Example illustrating the *corr* file-format.

	Genes	Correlation	Variance
G0	G1	0.00919292	0.120822
G0	G10	0.0722352	0.106228
G0	G100	-0.00253225	0.1061
G0	G101	0.0894089	0.0646841
G0	G102	0.0862046	0.11391
G0	G103	-0.150303	0.13394
G0	G104	0.118812	0.132093
G0	G105	-0.0582898	0.0976216
G0	G106	0.0192979	0.0709034
G0	G107	-0.039712	0.181778
G0	G108	-0.0525893	0.114026
G0	G109	0.00277228	0.128466
G0	G11	0.133993	0.126678
G0	G110	-0.100102	0.0883812
G0	G111	0.0647585	0.0815505
G0	G112	0.0848485	0.0734709
G0	G113	-0.0728113	0.0838207
G0	G114	0.182418	0.094951
G0	G115	0.0442724	0.0930236
G0	G116	-0.172493	0.124252
G0	G117	-0.00411641	0.104944
G0	G118	0.00568857	0.114492
G0	G119	0.234467	0.125555
G0	G12	0.129529	0.109903
G0	G120	-0.112295	0.10977
G0	G121	0.0944854	0.118301
G0	G122	0.0972697	0.109228
G0	G123	0.0580858	0.141031
G0	G124	0.138422	0.11856
G0	G125	-0.101314	0.0824208
G0	G126	0.0664987	0.146638
G0	G127	0.0317672	0.0796832
		1,1	Top

Figure 3 Example illustrating the *csd* file-format

	Genes	C-score	S-score	D-score
G0	G1	0.167952	0.127867	0
G0	G10	0.397162	0.0969776	0
G0	G100	0.221284	0.221284	0.011109
G0	G101	0.499928	0.0502764	0
G0	G102	0.124919	0.124919	0.0882528
G0	G103	0.333351	0.252804	0
G0	G104	0.196015	0.196015	0.104718
G0	G105	0.516148	0.249325	0
G0	G106	0.482682	0.38767	0
G0	G107	0.0710369	0.0710369	0.0103159
G0	G108	0.0392227	0.0392227	0.222203
G0	G109	0.178075	0.165928	0
G0	G11	0.301085	0.239803	0
G0	G110	0.624188	0.160432	0
G0	G111	0.300379	0.00163325	0
G0	G112	0.18283	0.18283	0.0326171
G0	G113	0.155494	0.155494	0.335784
G0	G114	0.976883	0.133882	0
G0	G115	0.48431	0.279739	0
G0	G116	0.3464	0.3464	0.0807872
G0	G117	0.0787412	0.0787412	0.016997
G0	G118	0.0857639	0.0857639	0.0229868
G0	G119	0.865467	0.0884216	0
G0	G12	0.383461	0.193321	0
G0	G120	0.530997	0.0551744	0
G0	G121	0.0944677	0.0944677	0.179528
G0	G122	0.511304	0.106031	0
G0	G123	0.118365	0.118365	0.22497
G0	G124	0.55116	0.0758019	0
G0	G125	0.282755	0.181194	0
G0	G126	0.401106	0.117602	0
G0	G127	0.395284	0.256992	0
--	INSERT	--	1,1	Top

Figure 4 Example illustrating the *all* file format.

Gene 1:	Gene 2:	corr 1:	corr 2:	Variance 1:	Variance 2:	C-score:	S-score:	D-score:
G0	G1	0.00919292	0.0678428	0.120822	0.0895635	0.167852	0.127867	0
G0	G10	0.0722352	0.118908	0.106228	0.125395	0.397162	0.0969776	0
G0	G100	-0.00253225	0.103414	0.1061	0.101739	0.221284	0.221284	0
G0	G101	0.0894089	0.109403	0.0646841	0.0934663	0.499928	0.0502764	0
G0	G102	0.0862046	-0.02252023	0.11391	0.146139	0.124919	0.124919	0
G0	G103	-0.150303	-0.0206541	0.13394	0.129069	0.333351	0.252804	0
G0	G104	0.118812	-0.0250465	0.132093	0.0967349	0.196015	0.196015	0
G0	G105	-0.0582898	-0.167225	0.0976216	0.0932764	0.516148	0.249325	0
G0	G106	0.0192979	0.176778	0.0709034	0.0941119	0.482682	0.38767	0
G0	G107	-0.039712	0.00268827	0.181778	0.0898604	0.0710369	0.0710369	0
G0	G108	-0.0525893	0.0711551	0.114026	0.110029	0.0392227	0.0392227	0
G0	G109	0.00277228	0.0785118	0.128466	0.07989	0.178075	0.165928	0
G0	G11	0.133993	0.0151815	0.126678	0.1188	0.301085	0.239803	0
G0	G110	-0.100102	-0.169361	0.0883812	0.0979849	0.624188	0.160432	0
G0	G111	0.0647585	0.0654665	0.0815505	0.106402	0.300379	0.00163325	0
G0	G112	0.0848485	-0.0069487	0.0734709	0.108071	0.18283	0.18283	0
G0	G113	-0.0728113	0.140246	0.0838207	0.104257	0.155494	0.155494	0
G0	G114	0.182418	0.24036	0.094951	0.0923502	0.976883	0.135882	0
G0	G115	0.0442724	0.165353	0.0930236	0.0943201	0.48431	0.279739	0
G0	G116	-0.172493	0.0180138	0.124252	0.0746261	0.3464	0.3464	0
G0	G117	-0.00411641	0.0422562	0.104944	0.12967	0.0787412	0.0787412	0
G0	G118	0.00568857	-0.0481368	0.114492	0.130477	0.0857639	0.0857639	0
G0	G119	0.234467	0.190999	0.125555	0.116119	0.865467	0.0884216	0
G0	G12	0.129529	0.0427003	0.109903	0.0918271	0.383461	0.193321	0
G0	G120	-0.112295	-0.138338	0.10977	0.113019	0.530997	0.0551744	0
G0	G121	0.0944854	-0.04600366	0.118301	0.144726	0.0944677	0.0944677	0
G0	G122	0.0972697	0.148167	0.109228	0.121192	0.511304	0.106031	0
G0	G123	0.0580858	-0.119208	0.141031	0.125626	0.118365	0.118365	0
G0	G124	0.138422	0.10495	0.11856	0.0764186	0.55116	0.0758019	0
G0	G125	-0.101314	-0.0221782	0.0824208	0.108327	0.282755	0.181194	0

1,1
Top

Figure 5 Example illustrating the *network* file-format.

Genes	Interaction-	
	type	
G101	G505	d
G102	G707	c
G112	G484	s
G118	G177	d
G130	G473	d
G138	G651	d
G139	G951	d
G141	G551	c
G142	G636	s
G149	G922	d
G158	G855	d
G164	G312	d
G166	G909	d
G167	G336	d
G169	G498	d
G176	G413	d
G202	G805	s
G208	G580	s
G210	G761	c
G211	G944	c
G22	G367	s
G227	G651	s
G231	G699	c
G250	G296	s
G252	G942	c
G26	G980	c
G262	G45	c
G28	G925	s
G282	G309	s
G285	G626	d
G287	G357	d
G289	G640	s
	1,1	Top

Appendix C

Table 2: Network parameters for the degree-preserving random networks generated from the S-component.

Network nr.	1	2	3	4	5	6	7	8	9	10
Radius	1	1	1	1	1	1	1	1	1	1
Diameter	9	9	8	8	8	7	10	8	8	8
Char.pt.len	3.704	3.685	3.666	3.671	3.647	3.697	3.734	3.656	3.707	3.658
Avg.degree	5.261	5.261	5.261	5.261	5.261	5.261	5.261	5.261	5.261	5.261
Density	0.011	0.011	0.011	0.011	0.011	0.011	0.011	0.011	0.011	0.011
Centralization	0.058	0.058	0.058	0.058	0.058	0.058	0.058	0.058	0.058	0.058
Heterogeneity	0.944	0.944	0.944	0.944	0.944	0.944	0.944	0.944	0.944	0.944
Clust.coeff.	0.025	0.017	0.025	0.024	0.032	0.021	0.031	0.024	0.021	0.027

Table 3: Network parameters for the degree-preserving random networks generated from the CD-component.

Network nr.	1	2	3	4	5	6	7	8	9	10
Radius	1	1	1	1	1	1	1	1	1	1
Diameter	10	11	10	9	9	9	9	10	10	9
Char.pt.len	4.203	4.029	4.084	4.076	4.049	4.084	4.072	4.080	4.099	4.034
Avg.degree	3.598	3.598	3.598	3.598	3.598	3.598	3.598	3.598	3.598	3.598
Density	0.008	0.008	0.008	0.008	0.008	0.008	0.008	0.008	0.008	0.008
Centralization	0.097	0.097	0.097	0.097	0.097	0.097	0.097	0.097	0.097	0.097
Heterogeneity	1.207	1.207	1.207	1.207	1.207	1.207	1.207	1.207	1.207	1.207
Clust.coeff.	0.024	0.021	0.022	0.019	0.032	0.021	0.026	0.021	0.020	0.023

Figure 6 Example of a degree-preserving random network generated from the S-dominated component in the arthritis-network.

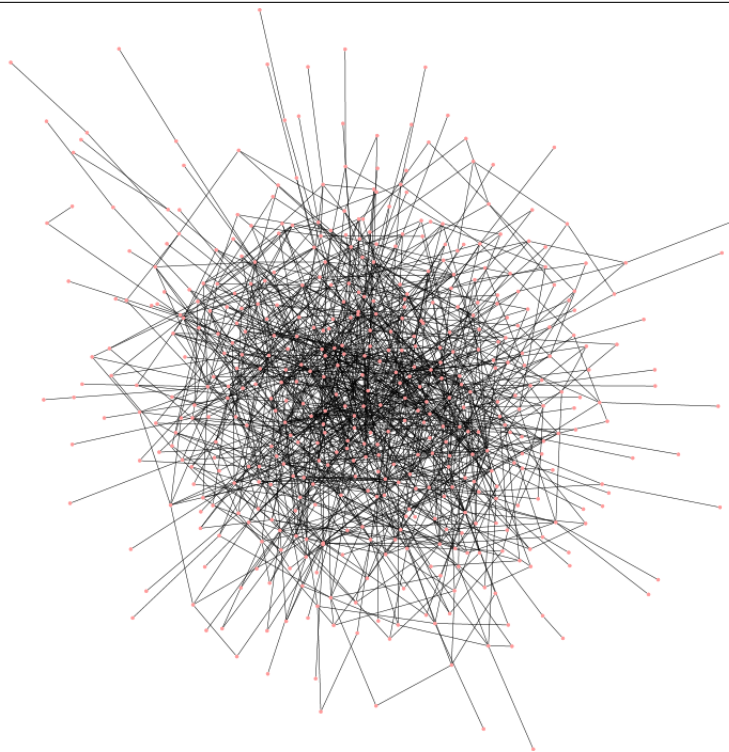
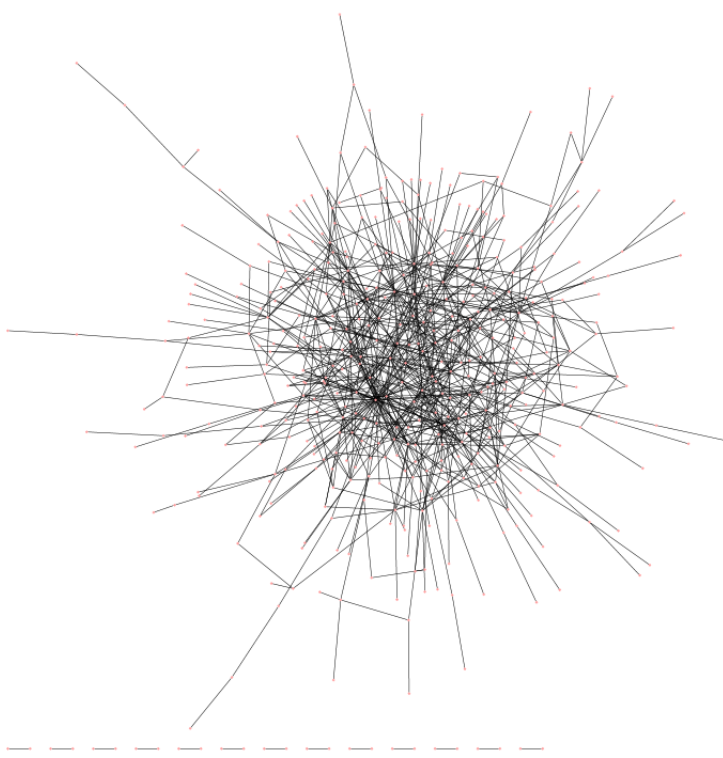


Figure 7 Example of a degree-preserving random network generated from the CD-dominated component in the arthritis-network.



Appendix D

Table 4: Spearman correlation between different node parameters within in the CD-dominated network component.

Module		C_C	C_i	k_i	ϵ	k_{ii}	T_i
CD	C_B	0.4608	0.3303	0.8920	-0.3564	-0.0647	0.4757
	C_C	-	0.5195	0.6067	-0.6068	0.6637	0.2185
	C_i	-	-	0.6457	-0.3479	0.2964	0.4874
	k_i	-	-	-	-0.4305	0.0589	0.5243
	ϵ	-	-	-	-	-0.3371	-0.2383
	k_{ii}	-	-	-	-	-	0.0579
	1	C_B	0.6549	0.3043	0.9149	-0.4495	-0.0714
C_C		-	0.3332	0.6477	-0.6484	0.5403	0.0698
C_i		-	-	0.5879	-0.2127	0.1274	0.5525
k_i		-	-	-	-0.4557	-0.0738	0.4122
ϵ		-	-	-	-	-0.3212	-0.1401
k_{ii}		-	-	-	-	-	-0.0661
2		C_B	0.6198	0.2950	0.8997	-0.3665	0.0001
	C_C	-	0.5455	0.7215	-0.5619	0.6690	0.3182
	C_i	-	-	0.5851	-0.2951	0.3372	0.5489
	k_i	-	-	-	-0.4372	0.1128	0.3713
	ϵ	-	-	-	-	-0.4536	-0.1832
	k_{ii}	-	-	-	-	-	0.2343
	5	C_B	0.5914	0.5223	0.9470	-0.4870	-0.1138
C_C		-	0.6308	0.6809	-0.7406	0.6041	0.4171
C_i		-	-	0.7173	-0.3270	0.3180	0.5198
k_i		-	-	-	-0.5042	-0.0027	0.6884
ϵ		-	-	-	-	-0.3300	-0.2962
k_{ii}		-	-	-	-	-	0.0448
10		C_B	0.4471	-0.1132	0.8622	-0.3859	-0.1588
	C_C	-	-0.0087	0.3002	-0.9902	0.0827	0.1430
	C_i	-	-	0.2667	-0.0553	-0.3048	0.5096
	k_i	-	-	-	-0.2654	-0.4133	0.5569
	ϵ	-	-	-	-	-0.0649	-0.1228
	k_{ii}	-	-	-	-	-	-0.2151
	15	C_B	0.6277	0.4580	0.9423	-0.4863	-0.1441
C_C		-	0.5378	0.5992	-0.9150	0.5370	0.5109
C_i		-	-	0.6681	-0.3672	-0.0952	0.3114
k_i		-	-	-	-0.4474	-0.2203	0.8295
ϵ		-	-	-	-	-0.6596	-0.3243
k_{ii}		-	-	-	-	-	-0.0833

Table 5: Spearman correlation between different node parameters within in the S-dominated network component.

Module		C_C	C_i	k_i	ϵ	k_{ii}	T_i
S	C_B	0.8011	0.1962	0.8937	-0.6564	-0.0788	-0.0572
	C_C	-	0.4384	0.8833	-0.7934	0.3630	-0.0264
	C_i	-	-	0.4954	-0.2496	0.5623	0.1537
	k_i	-	-	-	-0.6669	0.1427	-0.0105
	ϵ	-	-	-	-	-0.2358	-0.0138
	k_{ii}	-	-	-	-	-	0.1601
	0	C_B	0.8413	0.2387	0.9148	-0.6717	-0.1114
C_C		-	0.5293	0.9182	-0.7881	0.3109	-0.0713
C_i		-	-	0.5124	-0.3267	0.5586	0.1183
k_i		-	-	-	-0.6932	0.0657	-0.0473
ϵ		-	-	-	-	-0.2312	0.0705
k_{ii}		-	-	-	-	-	0.1588
3		C_B	0.7800	0.2366	0.9341	-0.7212	0.0447
	C_C	-	0.3364	0.8691	-0.8184	0.4581	0.1548
	C_i	-	-	0.3333	-0.3164	0.2323	0.0672
	k_i	-	-	-	-0.7569	0.0791	0.0966
	ϵ	-	-	-	-	-0.3821	-0.2373
	k_{ii}	-	-	-	-	-	0.2323
	6	C_B	0.8315	-0.1611	0.8649	-0.6007	-0.4697
C_C		-	0.0424	0.7980	-0.8196	-0.1323	-0.2754
C_i		-	-	0.1626	-0.1869	0.3532	0.6485
k_i		-	-	-	-0.5966	-0.2944	-0.2667
ϵ		-	-	-	-	0.0190	0.0593
k_{ii}		-	-	-	-	-	0.1344
7		C_B	0.8510	0.2779	0.9478	-0.6468	-0.1561
	C_C	-	0.4810	0.9265	-0.7811	0.2624	-0.0619
	C_i	-	-	0.4454	-0.3145	0.4598	0.0695
	k_i	-	-	-	-0.7143	-0.0173	-0.0390
	ϵ	-	-	-	-	-0.2490	0.0571
	k_{ii}	-	-	-	-	-	0.1498

Appendix E

Table 6: The table shows the Bonferroni corrected p-values related to the Spearman correlations between different node parameters for nodes within the network as a whole (aside from minor components and single nodes).

	C_C	C_i	k_i	ϵ	k_{ii}	T_i
C_B	4.954e-73	4.382e-12	1.703e-298	1.024e-26	9.921e-01	3.704e-14
C_C	-	4.953e-16	8.200e-142	5.282e-246	1.408e-41	4.489e-05
C_i	-	-	8.065e-63	1.0	8.836e-37	3.569e-23
k_i	-	-	-	1.265e-50	1.113e-02	1.829e-16
ϵ	-	-	-	-	1.330e-08	2.246e-03
k_{ii}	-	-	-	-	-	1.960e-02

Table 7: The table shows the Bonferroni corrected p-values related to the Spearman correlation between different node parameters for nodes within the CD-dominated network component.

	C_C	C_i	k_i	ϵ	k_{ii}	T_i
C_B	4.399e-23	2.738e-11	3.902e-151	3.045e-13	1.0	8.649e-25
C_C	-	2.542e-30	4.614e-44	4.490e-44	1.239e-55	8.216e-05
C_i	-	-	1.035e-51	1.374e-12	5.181e-09	3.430e-26
k_i	-	-	-	7.205e-20	1.0	5.693e-31
ϵ	-	-	-	-	8.919e-12	9.464e-06
k_{ii}	-	-	-	-	-	1.0

Table 8: The table shows the Bonferroni corrected p-values related to the Spearman correlation between different node parameters for nodes within the S-dominated network component.

	C_C	C_i	k_i	ϵ	k_{ii}	T_i
C_B	9.824e-108	3.011e-04	7.149e-168	2.089e-59	1.0	1.0
C_C	-	9.774e-23	1.133e-158	3.129e-104	3.852e-15	1.0
C_i	-	-	6.870e-30	5.901e-07	3.268e-40	1.493e-02
k_i	-	-	-	5.428e-62	3.531e-02	1.0
ϵ	-	-	-	-	3.405e-06	1.0
k_{ii}	-	-	-	-	-	8.770e-03

Table 9: The table shows the Bonferroni corrected p-values related to the Spearman correlation between different node parameters for nodes within module 0.

	C_C	C_i	k_i	ϵ	k_{ii}	T_i
C_B	4.012e-41	6.032e-02	2.045e-60	3.132e-20	1.0	1.0
C_C	-	3.576e-11	1.086e-61	1.595e-32	1.830e-03	1.0
C_i	-	-	2.308e-10	7.437e-04	1.086e-12	1.0
k_i	-	-	-	4.681e-22	1.0	1.0
ϵ	-	-	-	-	8.237e-02	1.0
k_{ii}	-	-	-	-	-	1.0

Table 10: The table shows the Bonferroni corrected p-values related to the Spearman correlation between different node parameters for nodes within module 1.

	C_C	C_i	k_i	ϵ	k_{ii}	T_i
C_B	5.392e-12	4.865e-02	2.857e-38	7.203e-05	1.0	3.612e-02
C_C	-	1.684e-02	1.193e-11	1.111e-11	1.961e-07	1.0
C_i	-	-	4.110e-09	7.456e-01	1.0	7.745e-08
k_i	-	-	-	5.076e-05	1.0	5.182e-04
ϵ	-	-	-	-	2.645e-02	1.0
k_{ii}	-	-	-	-	-	1.0

Table 11: The table shows the Bonferroni corrected p-values related to the Spearman correlation between different node parameters for nodes within module 2.

	C_C	C_i	k_i	ϵ	k_{ii}	T_i
C_B	1.707e-16	4.091e-03	1.282e-55	5.719e-05	1.0	2.395e-03
C_C	-	4.499e-12	6.998e-25	5.908e-13	3.882e-20	1.145e-03
C_i	-	-	2.724e-14	4.055e-03	3.748e-04	2.985e-12
k_i	-	-	-	2.700e-07	1.0	4.152e-05
ϵ	-	-	-	-	6.449e-08	4.719e-01
k_{ii}	-	-	-	-	-	7.010e-02

Table 12: The table shows the Bonferroni corrected p-values related to the Spearman correlation between different node parameters for nodes within module 3.

	C_C	C_i	k_i	ϵ	k_{ii}	T_i
C_B	2.485e-27	1.241e-01	1.432e-59	1.812e-21	1.0	1.0
C_C	-	1.485e-03	7.442e-41	3.056e-32	5.471e-07	1.0
C_i	-	-	1.741e-03	4.115e-03	1.452e-01	1.0
k_i	-	-	-	7.805e-25	1.0	1.0
ϵ	-	-	-	-	1.098e-04	1.211e-01
k_{ii}	-	-	-	-	-	1.453e-01

Table 13: The table shows the Bonferroni corrected p-values related to the Spearman correlation between different node parameters for nodes within module 5.

	C_C	C_i	k_i	ϵ	k_{ii}	T_i
C_B	4.186e-10	1.664e-07	2.814e-52	2.182e-06	1.0	1.110e-13
C_C	-	6.907e-12	1.491e-14	1.627e-18	1.187e-10	1.647e-04
C_i	-	-	7.579e-17	1.230e-02	1.776e-02	2.021e-07
k_i	-	-	-	6.462e-07	1.0	5.351e-15
ϵ	-	-	-	-	1.087e-02	4.096e-02
k_{ii}	-	-	-	-	-	1.0

Table 14: The table shows the Bonferroni corrected p-values related to the Spearman correlation between different node parameters for nodes within module 6.

	C_C	C_i	k_i	ϵ	k_{ii}	T_i
C_B	3.746e-16	1.0	6.758e-19	3.235e-06	1.896e-03	8.536e-02
C_C	-	1.0	6.126e-14	2.590e-15	1.0	5.799e-01
C_i	-	-	1.0	1.0	8.820e-02	1.444e-07
k_i	-	-	-	4.131e-06	3e823e-01	6.956e-01
ϵ	-	-	-	-	1.0	1.0
k_{ii}	-	-	-	-	-	1.0

Table 15: The table shows the Bonferroni corrected p-values related to the Spearman correlation between different node parameters for nodes within module 7.

	C_C	C_i	k_i	ϵ	k_{ii}	T_i
C_B	3.690e-35	3.424e-02	4.222e-62	5.854e-15	1.0	1.0
C_C	-	2.495e-07	3.655e-53	8.237e-26	6.281e-02	1.0
C_i	-	-	3.641e-06	7.042e-03	1.279e-06	1.0
k_i	-	-	-	1.262e-19	1.0	1.0
ϵ	-	-	-	-	1.036e-01	1.0
k_{ii}	-	-	-	-	-	1.0

Table 16: The table shows the Bonferroni corrected p-values related to the Spearman correlation between different node parameters for nodes within module 10.

	C_C	C_i	k_i	ϵ	k_{ii}	T_i
C_B	1.686e-01	1.0	1.192e-09	5.079e-01	1.0	3.459e-01
C_C	-	1.0	1.0	1.385e-27	1.0	1.0
C_i	-	-	1.0	1.0	1.0	4.371e-02
k_i	-	-	-	1.0	3.172e-01	1.311e-02
ϵ	-	-	-	-	1.0	1.0
k_{ii}	-	-	-	-	-	1.0

Table 17: The table shows the Bonferroni corrected p-values related to the Spearman correlation between different node parameters for nodes within module 15.

	C_C	C_i	k_i	ϵ	k_{ii}	T_i
C_B	4.294e-03	2.295e-01	1.639e-13	1.350e-01	1.0	2.122e-09
C_C	-	4.573e-02	9.825e-03	3.144e-11	4.656e-02	8.212e-02
C_i	-	-	1.148e-03	9.637e-01	1.0	1.0
k_i	-	-	-	2.766e-01	1.0	3.072e-07
ϵ	-	-	-	-	1.543e-03	1.0
k_{ii}	-	-	-	-	-	1.0

Appendix F

Table 18: This is a list of 60 genes found in the network related to transcription of the PRDM1 gene.

C11orf30	FOXE3	ZCCHC10
FAM13B	SLC35F2	ACTG2
SLC8A1	AKAP12	CTSO
SLC22A17	PP2R5E	BBX
DLGAP1	CTNBL1	SPINK4
PDIA3	TRIM63	MINK1
ZNF10	SLC39A14	MUT
C12orf54	SCAMP2	CYP27A1
MINPP1	SHQ1	GPRC5A
PIK3CA	PSPC1	TRAF5
CMPK1	ATF7	PNLIPRP3
ASTN1	ICA1L	SP140
CXCR5	CD1A	MALT1
SLC5A4	BAG4	PDZD2
PRPSAP2	ASB10	MFN2
UQCC	HSF5	CCL17
SLC38A4	GRIA4	SPEN
P2RY12	XRCC4	PRRX2
SNF8	RCL1	SLC2A9
BHMT	CCNYL1	F13B

Appendix G

Table 19: This is a list of 19 genes found in the network related to knockdown of the TYK2 gene.

PUS3	MRPS27	SLC35F2
UBE2G1	URB2	IKZF1
THOC2	CD3D	C1orf123
PDLIM1	MRPL3	ACLY
NSDHL	PPP2R5E	C1QBP
ARHGDI1	BCAT1	CUTC
NUP37		