A network modelling approach to investigate methylation profiles for type 1 diabetics

TBT4500

Biotechnology, Specialisation Project

Amalie Mello

Supervisor: Professor Eivind Almaas

Co-supervisor: Postdoc. André Voigt

Department of Biotechnology and Food Science, NTNU

August 9, 2020

Summary

The prevalence of type 1 diabetes has increased among children and adolescents [1]. Rakyan et al. had a hypothesis that some of the non-genetic factors were due to epigenetic variation, and they generated deoxyribonucleic acid methylation profiles on monozygotic twins where one of the twins had type 1 diabetes [2]. In this specialisation project, this data set was used to investigate methylation profiles for type 1 diabetics.

A differential co-expression network was created using software programs developed by Voigt et al. The resulting network was analysed, and the aim of this project thesis was to identify possible type 1 diabetes associated network patterns. The network was sparse and scale free, which strengthened the assumption that the network was a good representation of a real biological network. Candidates for type 1 diabetes relevant CpG sites, were found based on node properties like degree, closeness centrality, clustering coefficient and which types of edges they were connected with. Only a few type 1 diabetes relevant properties were found among the genes associated with these CpG sites.

The most interesting pattern as a possible type 1 diabetes associated network pattern was the greatest hub in the network, $cg09736162$, and its 39 neighbours connected with edges that represented differentiated co-expression. This hub had the highest closeness centrality among the hubs, the clustering coefficient of the hub was a bit high compared with other nodes that was not connected to other hubs and the pattern was a part of the biggest component.

Other patterns that were less central in the network were also found. Further analysis that could be done on the network was proposed.

Preface

This specialisation project was carried out in the spring of 2020. The thesis is submitted as part of a Master's degree at the Department of Biotechnology and Food Science, Norwegian University of Science and Technology (NTNU).

The author would like to express her gratitude to Professor Eivind Almaas and Postdoc. André Voigt, for being helpful and supportive throughout the process.

Contents

List of Tables

List of Figures

Abbreviations

Symbol		definition
A	=	adjacency matrix
a_{ij}	=	place in adjacency matrix given by indices
BC	=	betweenness centrality
BFS	=	Breadth-First Search
C_{i}	=	the clustering coefficient
$\langle C \rangle$	=	the average degree of clustering for a network
CC	=	closeness centrality
CSD	=	systematic framework that takes conserved, specific and differentiated co-expression into account
d	=	the shortest path between two nodes
$\langle d \rangle$	=	the average path length in a network
d_{max}	=	the longest of the shortest path between two nodes in a network
DNA	=	Deoxyribonucleic acid
GO	=	Gene ontology
H	=	homogeneity of the nodes
k	=	degree of a node
$\langle K \rangle$	=	the average degree of a network
$k_{\mathrm{p}}^{\mathrm{C,S,D}}$	=	cut-off thresholds for C,S and D
L	=	the number of edges
$\hat{L}$	=	the size of the samples that are drown from the data set
l	=	a sub-sample of data points
M	=	the number of the data points of different gene pairs
m	=	the number of samples drown from the data set
MZ	=	monozygotic
N	=	the number of nodes
$\hat{N}$	=	the number of the data points of individuals
n	=	number of links on a path in a network
$\hat{n}$	=	the size of the sub-samples that the data points are divided into
P	=	path in network
p	=	importance level
p_{k}	=	the degree distribution
$r_{\mathrm{ij,k}}^{\mathrm{l}}$	=	the Spearman rank-correlation
S_{i}	=	a sample that is drown from the data set
s	=	strength
T1D	=	type 1 diabetes

Symbol		**definition**
T1D-MVP	=	type 1 diabetes methylation variable position
TF	=	transcription factor
TO	=	topological overlap
wTO	=	weighted topological overlap
X_{p}	=	the average of maximum values per sample
$\rho_{\mathrm{ij,k}}$	=	the standard error of the correlation
$\sigma_{\mathrm{ij,k}}$	=	the standard error of the mean
$\tau_i(j,k)$	=	the number of shortest paths between the nodes j and k which also is passing through node i
$\tau(j,k)$	=	the number of shortest paths between the nodes j and k

Chapter 1

Introduction

The prevalence of type 1 diabetes (T1D) has increased among children and adolescents [1]. Rakyan et al. had a hypothesis that some of the non-genetic factors were due to epigenetic variation, and therefore they generated deoxyribonucleic acid (DNA) methylation profiles on monozygotic (MZ) twins where one of the twins had T1D [2]. To understand complex biological systems, like the epigenetic variation of T1D, experimental and computational research are important contributors. This is in the field of systems biology, where computational biology, pragmatic modelling and theoretical exploration are used [5].

Differential co-expression network analyses is an important tool for investigation of differentiation and dysfunctional gene-regulation in diseases. To distinguish between various forms of differential co-expression, Voigt et al. presented a systematic framework for differential co-expression network analysis that incorporates different types of interactions [6]. They have made software programs to execute this available online [7]. This creates networks that can be analysed in order to find disease associated pathways.

This specialisation project will execute a differential co-expression network analysis using the software programs developed by Voigt et al. The data set that will be used is the DNA methylation profiles generated by Rakyan et al [2]. The resulting network will be analysed, and the aim of this project thesis will be to identify possible T1D associated

network patterns.

Chapter 2

Theory

The aim of this project is to investigate methylation profiles for T1D using a differential co-expression network analysis, and to identify possible T1D associated network patterns. In order to gain an understanding of this, it is needed to have knowledge about the data set used and existing literature in the field of network theory and the the differential co-expression network analysis. This section will provide an introduction to this.

2.1 Type 1 diabetes

T1D is a chronic disease [8]. The symptoms of untreated T1D are impaired general condition, polyuria, thirst and loss in weight [9]. The blood glucose levels can become so high that the patient becomes dizzy or falls into a coma [9]. Insulin is necessary for the transport of glucose from the blood and into the cells. Insulin is secreted by β-cells [10]. People with T1D have absolute insulin deficiency due to β-cell destruction [11]. The β-cell destruction is normally caused by the immunity mechanisms. Patients therefor need a life long treatment with insulin [8]. $5 - 10\%$ of all diabetes cases are type 1 diabetes [12]. The prevalence of T1D among children and adolescents has increased [1]. Dabelea et al. reported that in 2009, 6666 out of $3,4$ million youth were diagnosed with T1D for a prevalence of $1,93$ per 1000. Inheritance is a large part of the cause of why a person

develops T1D [12]. Nevertheless, the triggering factors of unset of the clinical disease is not fully understood. The MZ twin pair discordance for the complex autoimmune disease childhood-onset T1D is around 50% [2]. Rakyan et al. had a hypothesis that some of the non-genetic factors were due to epigenetic variation. From purified immune effector CD14+ monocytes, they generated genome-wide DNA methylation profiles. After array processing, identification of type 1 diabetes methylation variable positions (T1D-MVPs), pyrosequencing validation and analysis, it was suggested that very early in the etiological process to the onset of T1D, T1D-MVPs arises.

2.2 Cellular differentiation and gene-regulation

Transcription factors (TFs) are proteins that regulates gene expression. At the promoter of a gene, there are interactions of TFs. The sum of this determine whether the gene is activated, repressed, or not regulated [13].

DNA methylation is the biological process where a methyl group is covalently added to a cytosine, and gives 5-methylcytosine. The biological process is an important epigenetic mark in eukaryotes [14; 15]. The enzymes that carries out the biological process is called DNA methyltransferases. DNA metyhlation is affecting transcriptional activity, and this can be associated with diseases [16]. Recent genomic technological advances have made it possible to do large scale studies of human disease associated or tissue-specific epigenetic variation, such as comparing DNA methylation profiles [17]. DNA methylation as gene regulator may be a more complex process than repression of gene expression [14]. Even though, causality and the physiological explanation of DNA methylation level variance are not fully studied, network theory can be used to study differences between conditions.

2.3 Network theory

Components that are connected can be represented in a network where the components are nodes connected with edges. The number of nodes (N) is sometimes referred to as the size of the network. Each node has an index (i), that is an integer from 1 to N. The number

of edges (L) is counted, and each edge gets its name based on the nodes it goes from and to [18]. Edge $(1, 2)$ is e.g. connecting node 1 and 2. One of the node properties is its degree (k_i), which is how many edges that connects it with other nodes. Here i denotes the index of the node. The average degree ($\langle K \rangle$) for undirected networks is a network property shown in Equation (2.1). For undirected networks $\langle K \rangle$ is as shown in Equation (2.2). The number of edges can be expressed through all the node degrees as shown in Equation (2.3) [18].

$$\langle K \rangle = \frac{1}{N} \sum_{i=1}^{N} k_i = \frac{2L}{N} \tag{2.1}$$

$$\langle K \rangle = \langle K_{in} \rangle = \langle K_{out} \rangle \tag{2.2}$$

$$L = \frac{1}{2} \sum_{i=1}^{N} k_i \tag{2.3}$$

The edges of a network can be directed or undirected. A network is called directed if all of its links are directed, and it is called undirected if all links are undirected. For a directed network, degree is separated in incoming degree ($k_\mathrm{i}^{\mathrm{in}}$) and outgoing degree ($k_\mathrm{i}^{\mathrm{out}}$) [18]. The sum of those is the total degree, as shown in Equation (2.4), and the total number of edges for a directed network in Equation (2.5).

$$k_i = k_i^{in} + k_i^{out} \tag{2.4}$$

$$L = \sum_{i=1}^{N} k_i^{in} = \sum_{i=1}^{N} k_i^{out} \tag{2.5}$$

The degree distribution (p_k) is the probability that a random node in the network has a specific degree (k) [18]. A scale free network is a network where the degree distribution follows a power law, and many real networks are scale free. The normalised histogram that

is the degree distribution for a network with N nodes, can be expressed from the number of degree-k nodes (N_{k}) as shown in Equation (2.6).

$$p_k = \frac{N_k}{N} \tag{2.6}$$

The adjacency matrix is a matrix which represents all the nodes and edges in a network [18]. A network with N nodes has an adjacency matrix of size N x N. The values in the adjacency matrix have indices i and j. On position A_{ij}, there is an edge from j to i, if the value is 1. If the value is 0, there is no connection.

In directed networks, the incoming degree is found by calculating the sums over the rows of the adjacency matrix [18]. The outgoing degree is found by calculating the sums over the columns. Undirected networks are symmetric, so both the types of sums give the degree. In real networks most of the positions in the adjacency matrix are 0, meaning the number of edges (L) is far less than the maximum possible number of edges if all nodes were connected (L_{max}). This is called a sparse network. When storing a network on a computer, memory is therefore saved, if only the edges are retained [18].

A path (P) between two nodes i_0 and i_{n}, with a given number of links between them (n), can be represented as an ordered list of n edges, as shown in Equation (2.7) [18]. n is in other words the length of the path. The shortest path between two nodes (d) i and j is denoted d_{ij}. The longest of the shortest paths between two nodes in the network (d_{max}) is the network diameter [18]. In order to find the shortest path using a computer, Breadth-First Search (BFS) algorithm is used [18]. The algorithm starts with one node, and labels the nodes neighbours. Then the neighbour's neighbours are labeled until the target node is reached. The average path length ($\langle d \rangle$) is the average shortest path between all the node pairs. $\langle d \rangle$ for directed networks with N nodes are shown in Equation (2.8).

$$P = \{(i_0, i_1), (i_1, i_2), (i_2, i_3), ..., (i_{n-1}, i_n)\} \tag{2.7}$$

$$\langle d \rangle = \frac{1}{N(N-1)} \sum_{i,j=1,N:i\neq j} d_{ij} \tag{2.8}$$

To use the adjacency matrix to find paths, the product of the values on the possible path between two nodes have to be checked. The simplest example is to check if there is a path of unit length ($d_{ij} = 1$) between two nodes i and j. Then $A_{ij} = 1$, if not $A_{ij} = 0$. To check if there is a path of length two between nodes i and j, then the product $A_{ik}A_{kj} = 1$, if not this will be 0. The number of paths with length two between i and j, named $N_{ij^{(2)}}$ is shown in Equation (2.9) [18]. The number of paths with a length d, is shown in Equation (2.10). The number of the shortest paths between nodes i and j is denoted N_{ij}.

$$N_{ij}^{(2)} = \sum_{k=1}^{N} A_{ik}A_{kj} = A_{ij}^2 \tag{2.9}$$

$$N_{ij}^{(d)} = A_{ij}^d \tag{2.10}$$

Two nodes are connected if there is a path between them. A network is connected if all node pairs are connected. A part of a network where all the nodes within are connected with each other, but not with the rest of the network, is called a component or a cluster [18]. In large networks, this can be identified with computers using the BFS algorithm [18]. A bridge is an edge that would disconnect the network, if it was removed.

The clustering coefficient (C_{i}) is to which degree the neighbours of a given node i connects to other nodes, and is shown in Equation (2.11) [18]. The clustering coefficient is a number between 0 and 1. A value of 0 indicates that none of the neighbours of node i connects with each other, and a value of 1 indicates that all the neighbours of node i is connecting with each other, which is called a clique. The degree of clustering for a network ($\langle$ C$\rangle$) is the average clustering coefficient for all the nodes in the network, and is shown in Equation (2.12) for an undirected network [18].

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

$$\langle C \rangle = \frac{1}{N} \sum_{i=1}^{N} C_i \tag{2.12}$$

2.3.1 Centrality measures

The point centrality measures betweenness centrality (BC) and closeness centrality (CC) are based on intuitive ways to see a node as central [19]. The betweenness centrality of a node increases with the number of shortest paths between two nodes in a network the given node is located on, as shown in Equation (2.13). The node with the highest closeness centrality in a network is the node with the lowest average shortest path to all the other nodes. The calculation of closeness centrality is shown in Equation (2.14).

$$BC_i = \sum_{j,k=1; j \neq k \neq i}^{N} \frac{\tau_i(j,k)}{\tau(j,k)} \tag{2.13}$$

$$CC_i = \frac{1}{\frac{1}{N} \sum_{j=1}^{N} L(i,j)} = \frac{1}{\langle L_i \rangle} \tag{2.14}$$

$\tau_i(j,k)$ is the number of shortest paths between the nodes j and k which also is passing through node i. $\tau(j,k)$ is the number of shortest paths between the nodes j and k.

2.3.2 Network assortativity

In an assortative network nodes are connected to other nodes with similar properties. Because hubs have a large number of edges, they have larger probability to connect with each other. In neutral networks, hubs connect with hubs or non-hubs according to what is random. In an assortative network, hubs tend to connect with other hubs [18]. In a disassortative network hubs tend to connect with nodes with few neighbours.

2.4 Gene co-expression networks

A co-expression network analysis aims to study system-level functionality of genes [20]. It will result in covariance for genes that are behaving similarly. This means genes that influence each other or are influenced by the same genes. To obtain this, a network is constructed where the nodes are the genes and the edges are connections between genes that are significantly co-expressed. The node connectivity measures are correlations like Pearson correlation [20] or Spearman correlation [21]. Zhang describes a general framework for soft thresholding that gives each gene pair a weight score. This is what is called weighted gene co-expression networks. Co-expression measures can be converted into weighted edges in a network using adjacency functions.

A gene correlation network strategy is presented schematically by Langfelder and Horvath [22]. The first step is to construct a gene co-expression network. To do this the raw data first have to be put into a similarity matrix, that include correlations for all possible gene pairs and then converting this into an adjacency matrix based on a threshold [20]. The next step is to identify modules using clustering methods. This makes it possible to carry out pathway based analysis. Then modules should be related to external information, to secure biological interesting modules. Lastly, significantly strong key genes in these modules has to be identified [22].

2.4.1 Weighted topological overlap

When using Spearman correlation to make a similarity matrix, it is not possible to find a cut-off threshold that does not include a lot of false positives with less than 10 measurements per gene. When testing p-value in sufficient repeats, something significant will be found with a very high probability. To avoid this problem, a lower p-value could be used, but this is problematic using Spearman correlation. With weighted topological overlap, it is possible to set appropriate strict criteria, and it is a way to avoid spurious correlations [23].

The topological overlap matrix is a network transformation [24]. Identifying modules with strongly correlated genes is another challenge during gene expression network anal-

ysis, but topological overlap can significantly strengthen and bring to focus modules in a network. Topological overlap is similar to clustering (C_i). However, clustering is a node trait, while topological overlap calculates the fraction of nearest neighbour nodes shared by two nodes i and j. Equation (2.15) and (2.16) shows how to calculate topological overlap (TO) and weighted topological overlap (wTO) matrices respectively.

$$w_{ij}^{TO} = \frac{\sum_k A_{ik}A_{kj} + A_{ij}}{min(k_i, k_j) + 1 - A_{ij}} \tag{2.15}$$

$$w_{ij}^{wTO} = \frac{\sum_k w_{ik}w_{kj}A_{ik}A_{kj} + w_{ij}A_{ij}}{min(s_i, s_j) + 1 - |w_{ij}A_{ij}|} \tag{2.16}$$

w_{ij}^{TO} and w_{ij}^{wTO} are values between 0 and 1. In Equation (2.16), degrees (k) is replaced with strengths (s), and strength values (w_{ij}) are included in addition to adjacency matrix edges A_{ij} which are 0 or 1. The probability of perfect wTO is much lower than a perfect correlation, thus it is increased power at low sample sizes using wTO in favour of correlation [23]. The advantages that wTO has over correlation measures, decreases with larger data sets.

2.5 Differential co-expression network analyses

Within the research field of cellular differentiation and gene-regulation, the use of differential co-expression network analyses has become an important contribution [6]. Network analyses can suggest pathways as with a certain probability are associated with a disease.

There are two kinds of differential co-expression [6]. One type defines differential co-expression based on correlation solely to one condition. Specific co-expression (S) is when a gene pair is correlated under one condition only. The other type of differential co-expression defines differential co-expression based on net change in pairwise correlation. Differentiated co-expression (D) is when a gene pair is correlated in both conditions, but with opposite sign. If a gene pair that is significantly co-expressed in one condition, do not show differential co-expression between conditions, it is conserved co-expression (C).

2.6 Differential co-expression network methods

According to Voigt et al., there are two main types of differential co-expression methods [6]. The first group of methods has links between genes that are co-expressed based on statistical criteria of significance. The co-expression networks are specific for each condition. The network for each condition can be compared, and interactions that take place in only one condition may be found. Interactions among genes are not considered in this group of methods [25].

In the other group of methods that Voigt et al. describes, however, a score is assigned for each possible pair of genes. This score is the difference between the correlation of a gene pair over the conditions. The correlation can for example be Pearson [26]. It can then be found whether the change in co-expression is significant between the conditions or not.

There are also different types of data that are produced in various differential co-expression network methods, and Voigt et al. also defines these methods in two main groups [6]. Some methods only find differently co-expressed genes, while others also identifies the relations between genes. They look at the gene's position in the network. The latter methods can either produce unweighted or weighted networks [27]. In unweighted networks an edge is either present or not, while in a weighted network the edges have a given strength. In order to convert a weighted network to become unweighted, a cut-off value of the strength can be introduced. Strengths over this value then is changed to 1, and strengths under to 0.

2.7 The CSD method

As mentioned, differential co-expression networks can be conserved (C), specific (S) or differentiated (D). Conserved (C) co-expression between conditions is when the co-expression for a given gene pair is significantly similar in the two conditions. This corresponds to a normal correlation with the same sign for both conditions. Specific (S) differential co-expression between conditions occur when the co-expression is strong for a given gene pair in one condition, but not in the other condition. This corresponds to a high correlation, positive or negative, in one condition, and a correlation value around 0

in the other. Differentiated (D) differential co-expression between conditions is when the co-expression for a given gene pair is significantly opposite for the two conditions. This means the correlation is commonly, but with the opposite sign in the two conditions. Voigt et al. has developed a systematic framework that takes all the three interactions into account, and this is called CSD [6]. They have made software programs available to execute the framework [7].

2.7.1 Finding correlation and variance for gene pairs in each condition

The first software program in the CSD-method is a C++ code for computing the co-expression for all possible pairs of genes. The software calculates the Spearman rank-correlation $r^{l}_{ij,k}$ for a gene pair i and j in condition k, over the $\hat{N}$ gene expression data points, for each sub-sample l of size $\hat{n}$.

When selecting sub-samples, this approach aims to get the highest possible number of sub-samples, in order to increase the chance of matching interesting conditions and ensuring independence between sub-samples. The number of data points $\hat{N}$ per gene, that is the number of individuals, correspond to the columns in the expression data text file. These are ordered and sequentially numbered for the full sample. These $\hat{N}$ data points are divided into sub-samples of size $\hat{n}$. The initiating data point $n*$, is the first data point in each sub-sample, starting from $\hat{N} = 1$. It is iterated through the data points, and the data points that have not previously co-occurred in a sub-sample with any of the points that is in the current sub sample, are added to this sub-sample. This continues until the size of the current sub-sample reaches $\hat{n}$, and then the next data point is the initiating point for a new sub-sample. In cases where no valid sub-sample of size $\hat{n}$ can be drawn with $n*$ as the initiating data point, the initiating data point $n*$ will be replaced with $n * +1$. When the initiating data point $n*$ is equal to the number of data points $\hat{N}$, the approach is completed.

If a sub-sample size $\hat{n}$ is selected so that $\hat{n}^2 = \hat{N}$, it will increase the number of sub-samples that can be generated. Nevertheless, small sub-sample sizes result in less accuracy in the Spearman correlations. To obtain a three-digit accuracy, a size of 7 is recommended.

This means that $\hat{N} = 49$ data points are recommended as minimum. Then the standard error of the correlation $\rho_{\text{ij,k}}$ within a given condition can be determined within a reasonable accuracy.

No Spearman rank-correlation corresponds to $r^{\text{l}}_{\text{ij,k}} = 0$, maximum positive correlation to $r^{\text{l}}_{\text{ij,k}} = 1$ and maximum negative correlation to $r^{\text{l}}_{\text{ij,k}} = -1$. Standard error of the mean $\sigma_{\text{ij,k}}$, then is calculated from the $r^{\text{l}}_{\text{ij,k}}$ values. Pairs of genes with high variance are less co-expressed.

2.7.2 The difference between conditions

The next procedure in the CSD-method is to find the differences between conditions. The types of co-expression, conseved (C), spesific (S) and differentiated (D), needs to be calculated. Voigt et al. have developed a python code for this task and one code to create the network. The edges of the differential co-expression network will be one of the co-expression types. A significant change or conservation between conditions will create an edge. The differential co-expression network produced with the CSD-method , then will show differences in transcription-pattern between the conditions. C, S and D types of differential and conserved co-expressions between condition i and j, are given by Equation (2.17)-(2.19).

$$C_{ij} = \frac{|\rho_{ij,1} + \rho_{ij,2}|}{\sqrt{\sigma^2_{ij,1} + \sigma^2_{ij,2}}} \tag{2.17}$$

$$S_{ij} = \frac{||\rho_{ij,1}| - |\rho_{ij,2}||}{\sqrt{\sigma^2_{ij,1} + \sigma^2_{ij,2}}} \tag{2.18}$$

$$D_{ij} = \frac{|\rho_{ij,1}| + |\rho_{ij,2}| - |\rho_{ij,1} + \rho_{ij,2}|}{\sqrt{\sigma^2_{ij,1} + \sigma^2_{ij,2}}} \tag{2.19}$$

i and j are the two genes of a pair, and 1 and 2 are the two conditions. C_{ij}, S_{ij} and D_{ij} are values between 0 and ∞. To normalise the values, cut-off thresholds $k_p^{C,S,D}$ for all types are set. All three types of values are supposed to correspond to the same importance level

P. Gene co-expression scores lower than the threshold is discarded. The importance value is determined based on the likelihood of obtaining a value from the underlying distribution. The underlying distribution is based on M data points (which is the number of different gene pairs) m samples s_i are drown from the data set. Each of the samples s_i has the size $\hat{L} \ll M$. The threshold value X_p is the average of maximal values per sample as shown in Equation (2.20). The importance level P is not the significant threshold, and is specified in Equation (2.21).

$$X_p = \frac{1}{m} \sum_{i=1}^{\infty} \max_{\{s_i\}} X \tag{2.20}$$

$$P = \frac{1}{\hat{L}} \tag{2.21}$$

A trait of the nodes is the node homogeneity H. All the nodes inn the network will be connected to other nodes with C, S or D type of edges. Nodes that have only one type of edges have the highest node homogeneity, that is $H_{\mathrm{i}} = 1$. Nodes that have equal amount of the three edges, will have an $H_{\mathrm{i}} = \frac{1}{3}$. The node homogeneity can be calculated as shown in Equation (2.22).

$$H_i = \sum_{j \in \{C,S,D\}} (\frac{k_{j,i}}{k_i})^2 \tag{2.22}$$

Chapter 3

Experiment

The aim of this project was to investigate methylation profiles for T1D patients using a differential co-expression network analysis, and to identify possible T1D associated network patterns. It was therefore preformed a CSD analysis on a data set consisting of T1D methylation profiles generated by Rakyan et al. [2].

3.1 Materials

Rakyan et al. made genome-wide DNA methylation profiles out of purified CD14+ monocytes from twin pairs where one of the two had T1D, and control pairs where none of them had T1D [2]. CD14+ monocytes is a cell type associated with T1D onset. Some information was removed from the text file. In this project, 32 of the T1D-discordant monozygotic (MZ) twin pairs were used with $27,006$ of the $27,578$ CpG sites for each individual. In total 100 individuals were included in the original data set. The data set generated by Rakyan et al. was stored in a text file consisting of general information on the first lines, and then the $27,578$ lines with CpG sites and values for the 100 individuals.

A python script was created in order to make two new text files, one with only the ones with type 1 diabetes, and another file with only the ones with a twin that had diabetes. The

code can be found as Listing 2 in Appendix A. This wrote two text files with blanc headers and lines with CpG sites and methylation values for 32 individuals. Methylation values that were left out in the original data set, were set to 0, and CpG sites including invalid methylation values, such as "$-3.40 \cdot 10^{38}$", were removed.

3.2 Methods

3.2.1 CSD analysis on type 1 diabetes

The CSD method developed by Voigt et al. was then used on the T1D discordance twin pair data sets to generate co-expression networks, as described in Section 2.7 [6]. Some of the lines in the C++ code were changed before it was ran with the T1D data set, as shown in Listing 1 in Appendix A. The same was done for the other file with the methylation values for the same CpG cites, but for the healthy twins of those contained in the first file. It took around 10 hours to run each of the codes on the computer Cruncher.

The two output files from running the C++ code one time for each of the data sets, were the input files in the python code that found C, S and D scores. Three of the output files from this code, were input files for the next python code that created networks. These input files were three auxiliary files that contained C, S and D scores for all the gene pairs. The cut-off value for the network, was fixed by a parameter that indicated approximate proportion of selected nodes. Four networks with values for this parameter set to $100,000$, $150,000$, $250,000$, and $300,000$ were created. These gave CSD-networks with the number of nodes equal to $6,689$, $4,570$, $3,402$ and $2,898$, respectively. These were all networks including a large number of nodes, thus the last one with the smallest number of nodes was used in further analysis. The python code created four networks with the value $300,000$ as the proportion of selected nodes parameter. Three of the networks consisted of only C, S or D interaction, and one consisted of all interaction types combined. The networks will be described in the results and analysis section.

3.2.2 Analysis on the resulting network

A visual representation of the network was created using Cytoscape. Analysis was preformed using the Python library networkx, and the code is found in Listing 3 in Appendix A. The CpG sites were converted to associated gene names using a data table published by Illumina Inc. [28]. This was done in Python with the code in Listing 4 in Appendix A. The Gene Ontology Consortium tools [4] and the NCBI Gene tools [3] were used to access biological functions.

Chapter 4

Results and analysis

4.1 Results from CSD

Using the CSD method on the data set with methylation values for people with and without T1D, resulted in four networks. Each of the networks was a text file where a line represented an edge. The two first columns were two CpG sites (source and target node), the third column specified the edge weight. The fourth column stated if the edge type was C, S or D. One of the four networks included all the edge types, and the three others were networks for each edge type alone. Figure 4.1 shows a visual representation of the combined CSD network. As shown in Figure 4.2, the biggest component consisted of 212 nodes (7.3%), the second largest consisted of 45 nodes (1.6%) and the rest of the components consisted of less than 30 nodes (1.0%). Thus, the vast majority of the nodes were not part of a large component.

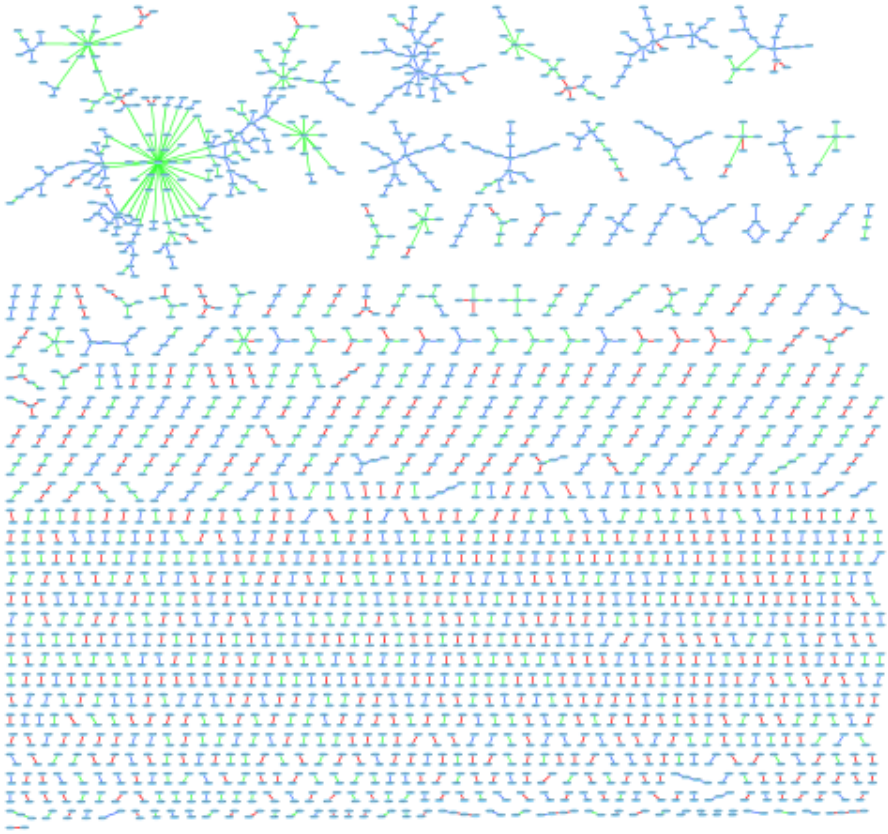

Figure 4.1: The figure shows a cytoscape created visual representation of the whole CSD network. C connections are represented by the colour blue, S connections with red and D connections with green.

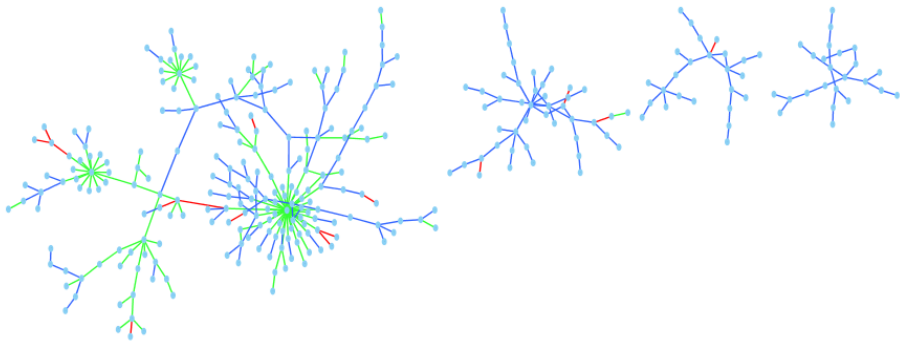

Figure 4.2: The figure shows a cytoscape created visual representation of the four biggest components of the network. C connections are represented by the colour blue, S connections with red and D connections with green.

4.2 Analysis with networkx

The CSD network was analysed using the library networkx in Python, and the code is found in Listing 3 in Appendix A. Results from the analysis are represented schematically in Table 4.1. The number of the different connection types, were relatively evenly distributed. Nevertheless, the D connection type was overrepresented at the center of the biggest component, as seen in Figure 4.2. In the periphery of the largest component and in the second largest component, it was most of the C connection type.

Network interactions	**CSD**	**C**	**S**	**D**
Number of nodes	2898	978	1120	979
Number of edges	1824	639	597	605
Average degree	1.2588	1.3067	1.0661	1.2360
The network is connected	False	False	False	False
The network has bridges	True	True	True	True
The density of the network	0.00043452	0.0013375	0.00095270	0.0012638
The average clustering of the network	0.0014280	0.0016725	0.0	0.0

Table 4.1: The table shows properties of four different networks. The first is the CSD network, and the rest are networks consisting of only C, S or D connections.

4.3 Degree distribution

The results from the analysis represented in Table 4.1 states that the average degree of the CSD network is 1.26, but this in not necessarily close to the degree of a typical node, and it does not say anything about if there are some nodes with very high degree. To examine that, the degree distribution is needed. The degree distribution tells how many nodes that has each degree, as seen in the diagram in Figure 4.3.

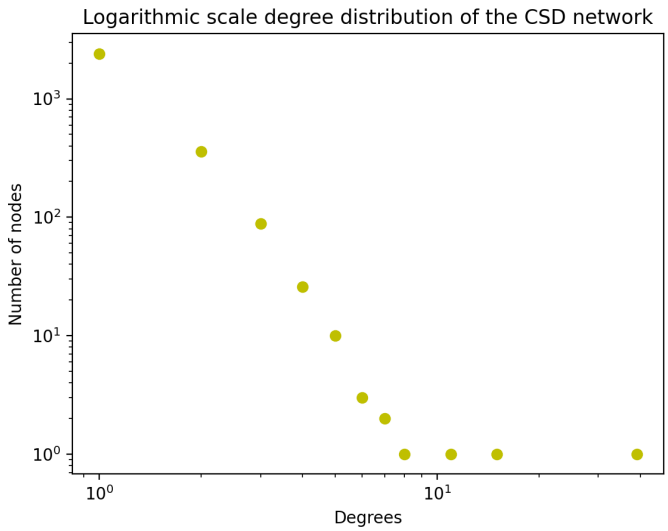

Figure 4.3: The diagram shows the degree distribution in the CSD network on logarithmic scales.

The degree distribution followed a power law, as a logarithmic plot gives a straight line. Networks that have degree distributions that follows a power law, are scale free [18]. This means that there were some nodes with degree far away from the average degree. A random generated network with the same number of nodes and edges, would have nodes with a more evenly distributed degree. Scale free networks, on the other hand, have more hubs. How much this hubs are methylated, may be critical to whether a person develops T1D.

4.4 Candindates for important CpG sites

The degree distribution indicated that some of the nodes had a much higher degree than the others, which is called hubs. By using the code in Listing 3 in Appendix A, the degrees higher than five that existed in the network, were identified. The code also gave a dictionary with node as key and degree as value. In that way the 9 nodes with more than 5 neighbours were identified, and is listed in Table 4.2.

$cg09736162$ was the node with the highest degree, and is highlighted in Figure 4.4. The three nodes with the highest degree were marked candidates for important CpG sites, as hubs often plays central functional roles. In addition, the three nodes also had a relatively high closeness centrality. Nodes that forms communities within a component in a network may have similar biological functions, or interact in the same processes. To investigate this, the neighbours of the three hubs were found using the same code in Appendix A. $cg10031456$ and $cg09088193$ were neighbours in the second biggest component, and both found on the top nine list of nodes with highest degree. Therefore, the nodes had relatively high clustering coefficients C_{i}. The firs of the two nodes had the fourth highest degree of all the nodes in the network. All its neighbours were therefor identified.

CpG site	$\mathbf{k_i}$	$\mathbf{CC_i}$	$\mathbf{C_i}$	**Nodes in component**	**Edge type**
cg09736162	39	0.016262	0.0026991	212	D
cg23173455	15	0.010012	0	212	D
cg04542415	11	0.0094805	0	212	D
cg10031456	8	0.0054332	0.035714	45	C
cg02946754	7	0.0080250	0	212	D
cg07588113	7	0.0021485	0	20	D
cg00067471	6	0.0020711	0	7	D
cg09088193	6	0.0052209	0.066667	45	C
cg04348872	6	0.0021143	0	8	D

Table 4.2: The table lists the nodes (CpG sites) with the highest degrees k_i. The node with the highest degree is found in the biggest component of 212 nodes. The CpG sites cg10031456 and cg09088193 are neighbours, and are found in the second biggest component of 45 nodes. The table also include closeness centrality CC_i, clustering C_i, the number of nodes in the component where the node is found and which edge type that it is most of linked to the node.

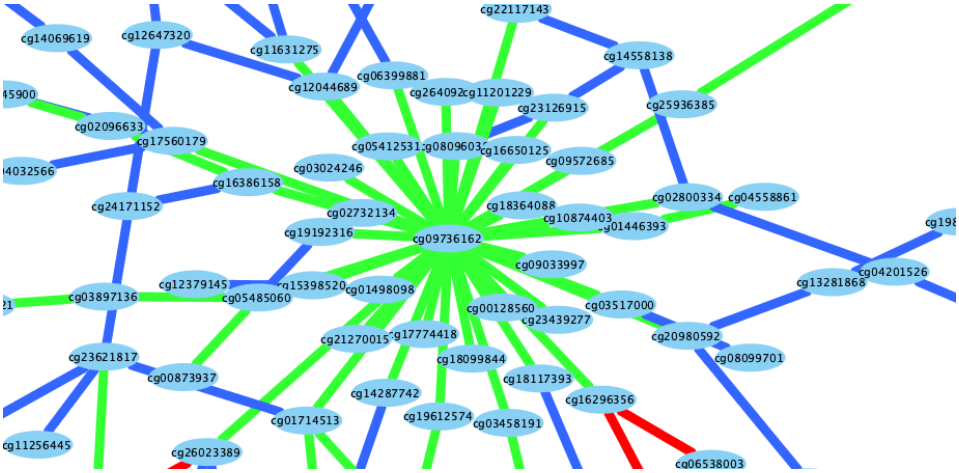

Figure 4.4: The figure shows a cytoscape created visual representation of some central nodes in the biggest components of the network. C connections are represented by the colour blue, S connections with red and D connections with green.

4.5 Biological function

To access biological functions using the Gene Ontology Consortium tools [4] and the NCBI Gene tools [3], the CpG sites had to be converted to associated gene names. The gene names of all the CpG sites were found using a data table published by Illumina Inc. [28]. This was done in Python with the code in Listing 4 in Appendix A. $25,450$ out of the $27,006$ CpG sites (94.2%) were found in the data table. The gene names found of the four hubs with highest degree and their neighbours are listed in Appendix B. The gene names of the hubs with highest degree ($cg09736162$) and third highest degree ($cg04542415$) was identified, and their functions and processes were found using the NCBI Gene tools, and given in Table 4.3 [3].

A Gene Ontology (GO) enrichment analysis with the ontology "molecular function", powered by PANTHER, was done on four gene sets [4]. The gene sets consisted of the four hubs with highest degree and their neighbours. The gene set with fewest neighbours gave no hits, but the three others are presented in Table 4.4.

CpG site	Functions and processes of the gene assosiated with CpG site
cg09736162	**CELSR3:** G protein-coupled receptor activity Calcium ion binding Protein binding G protein-coupled receptor signaling pathway Wnt signaling pathway, planar cell polarity pathway NAS axonal fasciculation cell-cell adhesion cilium assembly dopaminergic neuron axon guidance homophilic cell adhesion via plasma membrane adhesion molecules neuron migration planar cell polarity pathway involved in axon guidance regulation of protein localization regulation of protein phosphorylation serotonergic neuron axon guidance
cg04542415	**ZFP64:** DNA binding metal ion binding protein binding regulation of gene expression

Table 4.3: The table lists two hubs and their biological functions found using the NCBI Gene tools [3].

CpG site	Functions and processes of gene assosiated with CpG site
cg09736162 and its 39 neighbours	Binding (35.3%) Catalytic activity (35.3%) Molecular transducer activity (17.6%) Molecular function regulator (5.9%) Transcription regulator activity (5.9%)
cg23173455 and its 15 neighbours	Binding (40.0%) Catalytic activity (26.7%) Molecular function regulator (26.7%) Transporter activity (6.7%)
cg04542415 and its 11 neighbours	Binding (42.9%) Catalytic activity (28.6%) Structural molecule activity (14.3%) Transcription regulator activity (14.3%)

Table 4.4: The table lists results from GO enrichment analysis [4].

Most of the genes expected to be disease associated, had important biological functions and were protein coding. A search with NCBI Gene tools, was therefore done for all these genes. Relevant gene functions are listed in Table 4.5. An opposite search was also done, to see if some T1D associated genes were found in the network.

CpG site	Position in network	Processes involved in
cg11201229	D connection with the hub with highest degree and in the biggest component	Insulin receptor signaling pathway
cg10874403	D connection with the hub with highest degree and in the biggest component	Regulation of insulin-like growth factor receptor signaling pathway

Table 4.5: The table shows which central nodes in the network were found to be relevant to T1D using the NCBI Gene tools [3].

Gene and processes involved in	CpG site	Position in network
GRB10 Insulin receptor binding Insulin receptor signaling pathway Negative regulation of insulin receptor signaling pathway	cg06790324 cg25915982	Not central, one C edge Not central, one C edge
IGF2 Insulin receptor binding Insulin-like growth factor receptor binding Insulin receptor signaling pathway Insulin receptor signaling pathway via phosphatidylinositol 3-kinase Positive regulation of insulin receptor signaling pathway	cg16817891	Not central, one D edge
FOXO3 Insulin receptor signaling pathway	cg14310890	Not central, one C edge

Table 4.6: The table contains some genes that may be relevant to T1D and where the associated nodes are in the network. The gene function and processes were found using the NCBI Gene tools [3].

Chapter 5

Discussion

From the data set generated by Rakyan et al., only $\hat{N} = 32$ data points were used in the CSD method [2]. To determine the standard error of the correlation within a condition, $\hat{N} = 49$ is recommended as minimum [6]. This may have made the network and analysis from it less reliable.

The CSD network created from the data set was sparse and scale free. This is typical for a real and biological network [18], and therefore supported the assumption that the CSD network was good at presenting biological networks. In such a gene network, it is assumed that hubs play central roles. Communities with many internal links are assumed to influence each other, therefore they may have a similar function or take part in the same process.

5.1 The biological function of hubs in the network

The CSD network was designed in a way so that a correlation between the methylation levels in a CpG site pair that was either conserved or different between healthy and people with T1D provided an edge. If genes that were central to the network, encode something associated with T1D, it may strengthen the assumption that the network represents patterns

that can explain the difference between people with and without T1D. Centrality in the network can for example be in the form of closeness centrality or hubs.

Two T1D relevant genes were found to have a D connection with the hub with highest degree as seen in Table 4.5. The genes are involved in the insulin receptor signaling pathway and regulation of insulin-like growth factor receptor signaling pathway. In T1D, it is not the insulin receptors that do not work, but that they have absolute insulin deficiency [11]. The genes are therefore not highly relevant, but it is conceivable that the methylation levels of the genes may interact with genes that are more directly linked to T1D. However, it was not found relevant gene functions in any of the other nodes linked to this hub. Maybe this would have be found with further GO enrichment analysis. Another possibility might be that the other genes are important for T1D because they interact with genes that are associated with T1D. Insulin receptor associated genes were also found at not central places in the network, as seen in Table 4.6. This may be weaknesses with the network, but another explanation may be that some or all insulin receptor associated genes are not part of patterns that display T1D.

5.2 The types of connections in the network

The three different types of edges represents the different types of of differential co-expression [6]. In this case the values are the degree of methylation of a CpG site. Specific co-expression (S) is when a gene pair is correlated under only one condition. Differentiated co-expression (D) is when a gene pair is correlated in both conditions, but with the opposite sign. A gene pair that is significantly co-expressed in both condition and do not show differential co-expression between conditions, shows conserved co-expression (C). The network contained all the edge types, and the same edge types tend to accumulate in the larger components as seen in Figure 4.1.

The CSD network had a lot of C edges in the periphery of the biggest component, and in the second biggest component. This means that the genes which the edges are connecting are co-expressed both for people with and without T1D. It is therefore not likely to be a T1D associated pattern. Nevertheless, the genes seem to have important roles that is

conserved across conditions.

A GO enrichment analysis was done on $cg10031456$ and its neighbours which were connected homogeneously by C edges. To do that the CpG sites were first converted to gene names. Only four of the nine CpG sites were converted, and in this analysis it is not known if the genes are protein coding. The GO enrichment analysis on molecular function did not give any hits. Nevertheless, it seems that the genes play an important role when it comes to interaction with other genes.

The D edges were found central in the network, in the way that the three biggest hubs, that were also found in the biggest component, all were homogeneously linked with D edges. The edges represents opposite correlations for people with and without T1D. Hubs like these with a lot of D edges might be a pattern that indicates T1D.

The greatest hub, $cg09736162$, had 39 neighbours connected with D edges as seen in Table 4.4. It also had the highest closeness centrality among the hubs. Although the network was sparse, the clustering coefficient was a little high compared with other nodes that were not connected to other hubs. This clustering is high for a node with only D edges. All triangles that a node with only D edges is a part of have at least two D edges, and the two negatives become positive, therefore it becomes mathematically impossible to have a triangle of negative correlations. The hub also was located in the biggest component. These properties makes the D edges between the hub and its neighbours a possible T1D associated pattern.

The second greatest hub, $cg23173455$, had 15 neighbours that were all connected with D edges. Its clustering coefficient was 0, but this is expected for a node with D edges only. Its closeness centrality was lower than for the greatest hub. This makes the greatest hub with its connections a stronger candidate for being part of a T1D associated pattern. Nevertheless, it is not refuted that the hub $cg23173455$ may be a part of such a pattern as well. The same goes for the third greatest hub, $cg04542415$ with its 11 D edges.

S edges also represents a difference in correlation of methylation levels between gene pairs over conditions. Therefore, these edges can also be examined to find possible disease pathways. On the network in Figure 4.1, there were few accumulations of S edges, and

many of these edges were in small components. It may therefore be more likely that the D edges represents interesting T1D associated pathways.

Chapter 6

Conclusion

A differential co-expression network analysis was executed using the software programs developed by Voigt et al. The data set that was used was the DNA methylation profiles generated by Rakyan et al [2]. The resulting network was analysed, and it was assumed that the network was a good representation of a real biological network, as it was sparse and scale free [18]. A network with some hubs gave rise to candidates for T1D relevant CpG sites, based on node properties like degree, closeness centrality, clustering coefficient and which types of edges they were connected with [18; 19].

Gene Ontology enrichment analysis and gene searches was done to investigate the candidates for important nodes and their neighbours. Few T1D relevant properties were found in the searchers.

The second and third greatest hubs, $cg23173455$ and $cg23173455$ respectively, were less central in the network, but their connections to their neighbours may be investigated closer, as they both were D edge homogeneous hubs. Because C edges represents conserved co-expression across conditions, they are not expected to be part of T1D associated pathways or patterns. There were few accumulations of S edges in the network, and many of these edges were in small components.

A possible T1D associated network pattern may rather be the greatest hub, $cg09736162$,

and its 39 neighbours connected with D edges. This was the most interesting pattern. This hub had the highest closeness centrality among the hubs. The clustering coefficient of the hub was a bit high compared with other nodes that were not connected to other hubs. $cg09736162$ was also a part of the biggest component.

Further analysis should be done on the network. It is suggested to be done in a more systematic manner, to get results with more statistical power and to get more results faster. Approaches using machine learning, could be a way to obtain this, and then $LASSO$ could be a useful method [29].

Bibliography

[1] Dabelea D, Mayer-Davis EJ, Saydah S, Imperatore G, Linder B, Divers J, et al. Prevalence of type 1 and type 2 diabetes among children and adolescents from 2001 to 2009. JAMA - Journal of the American Medical Association. 2014;.

[2] Rakyan VK, Beyan H, Down TA, Hawa MI, Maslau S, Aden D, et al. Identification of type 1 Diabetes-associated DNA methylation variable positions that precede disease diagnosis. PLoS Genetics. 2011;7(9):1–9. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3183089/pdf/pgen.1002300.pdf.

[3] NCBI. Gene;. Available from: https://www.ncbi.nlm.nih.gov/gene/.

[4] Gene Ontology Consortium. Gene Ontology;. Available from: http://geneontology.org/.

[5] Kitano H. Computational systems biology; 2002.

[6] Voigt A, Nowick K, Almaas E. A composite network of conserved and tissue specific gene interactions reveals possible genetic interactions in glioma. vol. 13; 2017. Available from: https://doi.org/10.1371/journal.pcbi.1005739.

[7] Voigt A, Nowick K, Almaas E. CSD;. Available from: https://github.com/andre-voigt/CSD.

[8] Atkinson MA, Eisenbarth GS, Michels AW. Type 1 diabetes; 2014. Available from: https://doi.org/10.1016/S0140-6736(13)60591-7.

[9] Usher-Smith JA, Thompson MJ, Sharp SJ, Walter FM. Factors associated with the presence of diabetic ketoacidosis at diagnosis of diabetes in children and young adults: A systematic review; 2011.

[10] Rorsman P, Renström E. Insulin granule dynamics in pancreatic beta cells; 2003.

[11] Kerner W, Brückel J. Definition, classification and diagnosis of diabetes mellitus; 2014.

[12] Daneman D. Type 1 diabetes. In: Lancet; 2006. .

[13] Ravasi T, Suzuki H, Cannistraci CV, Katayama S, Bajic VB, Tan K, et al. An Atlas of Combinatorial Transcriptional Regulation in Mouse and Man. Cell. 2010;140(5):744–752.

[14] Zhu H, Wang G, Qian J. Transcription factors as readers and effectors of DNA methylation. Nature Reviews Genetics. 2016;17(9):551–565. Available from: http://dx.doi.org/10.1038/nrg.2016.83.

[15] Bestor TH. DNA methylation: evolution of a bacterial immune function into a regulator of gene expression and genome structure in higher eukaryotes.; 1990.

[16] Bird A. Epigentic Memory. Genes and Development. 2002;16:16–21. Available from: http://genesdev.cshlp.org/content/16/1/6.full?sid=457710c3-37d5-4723-9641-8e625d965d11.

[17] Rakyan VK, Down TA, Balding DJ, Beck S. Epigenome-wide association studies for common human diseases; 2011.

[18] Sukma M. Network Science. Cambridge University Press; 2015. Available from: http://networksciencebook.com/.

[19] Freeman LC. Centrality in social networks conceptual clarification. Social Networks. 1978;.

[20] Zhang B, Horvath S. A general framework for weighted gene co-expression network analysis. Statistical Applications in Genetics and Molecular Biology. 2005;Available from: https://pubmed.ncbi.nlm.nih.gov/16646834/.

[21] Yu H, Liu BH, Ye ZQ, Li C, Li YX, Li YY. Link-based quantitative methods to identify differentially coexpressed genes and gene Pairs. BMC Bioinformatics. 2011;12(1):315. Available from: http://www.biomedcentral.com/1471-2105/12/315https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3199761/pdf/1471-2105-12-315.pdf/?tool=EBI.

[22] Langfelder P, Horvath S. WGCNA: An R package for weighted correlation network analysis. BMC Bioinformatics. 2008;9. Available from: https://bmcbioinformatics.biomedcentral.com/track/pdf/10.1186/1471-2105-9-559.

[23] Voigt A, Almaas E. Assessment of weighted topological overlap (wTO) to improve fidelity of gene co-expression networks. BMC Bioinformatics. 2019;20(1):1–11. Available from: https://bmcbioinformatics.biomedcentral.com/track/pdf/10.1186/s12859-019-2596-9.

[24] Nowick K, Gernat T, Almaas E, Stubbs L. Differences in human and chimpanzee gene expression patterns define an evolving network of transcription factors in brain. Proceedings of the National Academy of Sciences of the United States of America. 2009;106(52):22358–22363. Available from: https://www.pnas.org/content/pnas/106/52/22358.full.pdf.

[25] Choi JK, Yu U, Yoo OJ, Kim S. Differential coexpression analysis using microarray data and its application to human cancer. Bioinformatics. 2005;21(24):4348–4355. Available from: https://doi.org/10.1093/bioinformatics/bti722.

[26] Yu H, Liu BH, Ye ZQ, Li C, Li YX, Li YY. Link-based quantitative methods to identify differentially coexpressed genes and gene Pairs. BMC Bioinformatics. 2011;12.

Available from: https://bmcbioinformatics.biomedcentral.com/track/pdf/10.1186/1471-2105-12-315.

[27] Bellingeri M, Bevacqua D, Scotognella F, Alfieri R, Cassi D. A comparative analysis of link removal strategies in real complex weighted networks. Scientific Reports. 2020;10(1):1–15. Available from: https://www.nature.com/articles/s41598-020-60298-7.pdf.

[28] NCBI. Illumina HumanMethylation450 BeadChip; 2011. Available from: https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GPL13534.

[29] Tibshirani R. Regression Shrinkage and Selection Via the Lasso. Journal of the Royal Statistical Society: Series B (Methodological). 1996;Available from: https://doi.org/10.1111/j.2517-6161.1996.tb02080.x.

Appendix A

```
//Parameters depending on input file

const char* expDataFile = "T1D.txt"; //Name of expression
↪ data file
const char* outFile = "RhoAndVarOrig.txt"; //Name of output
↪ data file
const int sampleSize = 32; //Number of data points per gene
↪ (normally number of individuals from which data is
↪ collected, corresponds to columns in the expression
↪ data text file)
const int numberOfGenes = 27006;// Number of distinct genes
↪ for which there is expression data (corresponds to rows
↪ in expression data)
const int subSampleSize = 10;// of subsamples for
↪ determination of variance in co-expression. 10 is a
↪ good minimum - can be increased if sampleSize is very
↪ large.
```

Listing 1: Parameters depending on input file.

Python codes

```
#This code splits the dataset into one text-file with T1D
↪ and one with their twin.
import re
```

```
count = len(open("twin_study.txt").readlines(  ))# 27580
print(count)
#Opens file and makes the variable lists:
lists=[]
with open("twin_study.txt", "r") as f:
    tittel= f.readline() #Line 1 is the title
    descriptionsList= f.readline().split("\t") #Line two is
    ↪  the descriptions
    for lis in range(count-2): #Line 3 to 27580
        midlertidigList = f.readline().split("\t")
        #lists.append(midlertidigList)
        #import re
        matches1 = 0
        pattern1 = re.compile(r"\d.\d+E.\d+")
        match1 = pattern1.finditer(str(midlertidigList))
        #print(len(match1))
        lengde=0
        for m in match1:
            lengde= lengde+1
            #print(lengde)
        if lengde<1:
            lists.append(midlertidigList)
            #print(m)
#Now, lists are all lines from line 3 in the dataset (exept
↪  the lines including -3.40E+38), starting with the cg
↪  number that has index 0.
print(len(lists))

#Making a list with descriptions of those with T1D:
T1DDescriptions = []
```

```
#import re  # regex
for object_in_list in descriptionsList:
    text_to_search = object_in_list
    # print(text_to_search)
    pattern =
    ↪  re.compile(r'"\d+_CD1?4_T1Dpair\d+_T1D:AVG_Beta"\n?')
    ↪  # defines the pattern that will be searched for.
    matches = pattern.finditer(text_to_search)  # indicates
    ↪  where the pattern will be searched for in.
    for match in matches:
        #print(match[0])  # [0] indicates which property
        T1DDescriptions.append(match[0])

#Making a list with descriptions of those that have a twin
↪  with T1D:
T1DTwinDescriptions = []
#import re  # regex
for object_in_list in descriptionsList:
    text_to_search = object_in_list
    # print(text_to_search)# ok
    pattern =
    ↪  re.compile(r'"\d+_CD1?4_T1Dpair\d+_unaffected:AVG_Beta"\n?')
    ↪  # definerer mønsteret som skal søkes etter
    matches = pattern.finditer(text_to_search)  # hvilken
    ↪  tekst man skal mete etter mønsteret i.
    for match in matches:
        #print(match[0])  # [0] peker på hvilken
        ↪  egenskap/gruppe vi vil se på.
        T1DTwinDescriptions.append(match[0])
```

```
#Find the indeces that T1D descriptions have in
↪ descriptionsList:
T1DIndexList=[0] #Starts with zero to get the nucleotide
↪ positions first in each row.
for i in range(len(descriptionsList)):
    for j in range(len(T1DDescriptions)):
        if descriptionsList[i]==T1DDescriptions[j]:
            T1DIndexList.append(i)

#Find the indeces that T1DTwin descriptions have in
↪ descriptionsList:
T1DTwinIndexList=[0]
for i in range(len(descriptionsList)):
    for j in range(len(T1DTwinDescriptions)):
        if descriptionsList[i]==T1DTwinDescriptions[j]:
            T1DTwinIndexList.append(i)

#Find the indeces that cg descriptions have in
↪ descriptionList:
cgIndexList=[0]


#Function that allow you to choose indeces:
#source: https://www.oreilly.com/library/view/
# python-cookbook/0596001673/ch04s07.html
def select(lst, indices):
    return (lst[i] for i in indices)

#Make string of T1D descriptions (T1DDescriptionsStr):
```

```
#T1DDescriptionsTuple = select(descriptionsList,
↪  T1DIndexList)
T1DDescriptionsStr=""
#for i in T1DDescriptionsTuple:
#    T1DDescriptionsStr+=(i + "\t")
#if T1DDescriptionsStr.endswith("\t"):
#    T1DDescriptionsStr=T1DDescriptionsStr[:-2]
T1DDescriptionsStr+="\n"

#Make string of T1DTwin descriptions
↪  (T1DTwinDescriptionsStr):
#T1DTwinDescriptionsTuple = select(descriptionsList,
↪  T1DTwinIndexList)
T1DTwinDescriptionsStr=""
#for i in T1DTwinDescriptionsTuple:
#    T1DTwinDescriptionsStr+=(i + "\t")
#if T1DTwinDescriptionsStr.endswith("\t"):
#    T1DTwinDescriptionsStr=T1DTwinDescriptionsStr[:-2]
T1DTwinDescriptionsStr+="\n"

cgDescriptionsStr=""
#cgDescriptionsStr+="\n"

#write T1D.txt
with open("T1D.txt", "w") as T1D_f:
    #T1D_f.write("Nucleotide position of CpG site -
    ↪  T1D"+"\n")
    T1D_f.write(T1DDescriptionsStr)
    for i in range(len(lists)):
        L=lists[i]
```

```
        T1DValuesTuple = select(L, T1DIndexList)
        T1DValuesStr=""
        for i in T1DValuesTuple:
            T1DValuesStr+=i + "\t"
        if T1DValuesStr.endswith("\t"):
            T1DValuesStr=T1DValuesStr[:-2]
        T1DValuesStr+="\n"

        T1DValuesStr=T1DValuesStr.replace('"', '')
        T1DValuesStr=T1DValuesStr.replace('\t\t', '\t0\t')
        T1DValuesStr=T1DValuesStr.replace('\t\t', '\t0\t')
        T1D_f.write(T1DValuesStr)


#write T1DTwin.txt
with open("T1DTwin.txt", "w") as T1DTwin_f:
    #T1DTwin_f.write("Nucleotide position of CpG site -
    ↪  T1D"+"\n")
    T1DTwin_f.write(T1DTwinDescriptionsStr)
    for i in range(len(lists)):
        L=lists[i]
        T1DTwinValuesTuple = select(L, T1DTwinIndexList)
        T1DTwinValuesStr=""
        for i in T1DTwinValuesTuple:
            T1DTwinValuesStr+=i + "\t"
        if T1DTwinValuesStr.endswith("\t"):
            T1DTwinValuesStr=T1DTwinValuesStr[:-2]
        T1DTwinValuesStr+="\n"

        T1DTwinValuesStr=T1DTwinValuesStr.replace('"', '')
```

```
        T1DTwinValuesStr=T1DTwinValuesStr.replace('\t\t',
        ↪  '\t0\t')
        T1DTwinValuesStr=T1DTwinValuesStr.replace('\t\t',
        ↪  '\t0\t')
        T1DTwin_f.write(T1DTwinValuesStr)


#write cg.txt
with open("cg.txt", "w") as cg_f:
    #cg_f.write("Nucleotide position of CpG site -
    ↪  T1D"+"\n")
    cg_f.write(cgDescriptionsStr)
    for i in range(len(lists)):
        L=lists[i]
        cgValuesTuple = select(L, cgIndexList)
        cgValuesStr=""
        for i in cgValuesTuple:
            cgValuesStr+=i + "\t"
        if cgValuesStr.endswith("\t"):
            cgValuesStr=cgValuesStr[:-2]
        cgValuesStr+="\n"

        cgValuesStr=cgValuesStr.replace('"', '')
        cgValuesStr=cgValuesStr.replace('\t\t', '\t0\t')
        cgValuesStr=cgValuesStr.replace('\t\t', '\t0\t')
        cg_f.write(cgValuesStr)
```

Listing 2: linesToLists.py: This code splits the data set into one text-file with T1D and one with their twin.

```
#This code analyses the networks with networkx
from networkx.utils import open_file, make_str
import networkx as nx
```

```
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
import numpy as np

# Type in the network that is to be analysed.
 ↪  'CSDSelection.txt', 'CNetwork.txt', 'SNetwork.txt' or
 ↪  'DNetwork.txt'.
K = nx.read_edgelist('CSDSelection.txt', nodetype=str,
 ↪  data=(('weight',float),('theType', str),))
print(nx.info(K))
#nx.draw(K)
#plt.show()
#print(nx.get_edge_attributes(K,'theType')) #or 'weight'
print('The network is connected:', nx.is_connected(K))
print('The network has bridges:', nx.has_bridges(K))
# print('The average shortest path length:',
 ↪  nx.average_shortest_path_length(K)) #Do this on the
 ↪  biggest component.
print('The density of the network:', nx.density(K))
#print('The clustering of the network:', nx.clustering(K))
 ↪  #gives the clustering value for each node.
print('The average clustering of the network:',
 ↪  nx.average_clustering(K))

#Comment out the 9 following lines for plain C, S or
 ↪  DNetwork.

sources=['cg09736162', 'cg23173455', 'cg04542415',
 ↪  'cg10031456', 'cg02946754', 'cg07588113', 'cg00067471',
 ↪  'cg09088193', 'cg04348872']
```

```
for i in range(len(sources)):
    print(sources[i], ':')
    print('CC: ', nx.closeness_centrality(K, sources[i]))
    print('Clustering: ', nx.clustering(K, sources[i]))
print('Neighbours of cg09736162:',list(nx.all_neighbors(K,
↪  'cg09736162')))
print('Neighbours of cg23173455:',list(nx.all_neighbors(K,
↪  'cg23173455')))
print('Neighbours of cg04542415:',list(nx.all_neighbors(K,
↪  'cg04542415')))
print('Neighbours of cg10031456:',list(nx.all_neighbors(K,
↪  'cg10031456')))

degrees_dictionary1 = nx.degree(K)

targets= list(degrees_dictionary1)
print(targets)

print(degrees_dictionary1) #To find which nodes that has
↪  each degree.
all_degrees1 = []
for key in degrees_dictionary1:
    all_degrees1.append(key[1])
unique_degrees1 = list(set(all_degrees1))  # all the unique
↪  degrees.
unique_degrees1.sort(reverse=True)
print(unique_degrees1)

#DEGREE DISTRIBUTION:
```

```
#Source for the degree distribution function: The channel
↪  "Social networks" on Youtube.

#Fitting function
#Source for the regression: The channel "Phys Whiz" on
↪  YouTube.
def func(x,a,b):
    return a*np.exp(b*x)

def plot_deg_dist(K):

    degrees_dictionary= nx.degree(K)
    all_degrees=[]
    for key in degrees_dictionary:
        all_degrees.append(key[1])

    unique_degrees = list(set(all_degrees)) #all the unique
    ↪  degrees.

    count_of_degrees= []

    for i in unique_degrees:
        x= all_degrees.count(i)
        count_of_degrees.append(x)

    plt.plot(unique_degrees, count_of_degrees, 'yo-')
    plt.xlabel('Degrees')
    plt.ylabel('Number of nodes')
    plt.title('Degree distribution of the CSD network')
    plt.show()
```

```
plt.loglog(unique_degrees, count_of_degrees, 'yo')
plt.xlabel('Degrees')
plt.ylabel('Number of nodes')
plt.title('Logarithmic scale degree distribution of the
↪  CSD network')
plt.show()

xData=np.array(unique_degrees)
yData=np.array(count_of_degrees)
plt.plot(xData, yData, 'yo', label='Experimental data')
popt, pcov=curve_fit(func, xData, yData)

residuals = yData - func(xData, *popt)
ss_res = np.sum(residuals ** 2)
ss_tot = np.sum((yData - np.mean(yData)) ** 2)
r_squared = 1 - (ss_res / ss_tot)

xFit=np.arange(1.0, 40.0, 0.01)
plt.plot(xFit, func(xFit, *popt), 'r', label='y= %5.3f
↪  * x^ %5.3f' % tuple(popt))
plt.plot(xFit, func(xFit, *popt), 'r', label='R^2=
↪  %5.5f' % r_squared)
plt.xlabel('Degrees')
plt.ylabel('Number of nodes')
plt.title('Degree distribution of the CSD network with
↪  regression line')
plt.legend()
plt.show()
```

```
    plt.loglog(unique_degrees, count_of_degrees, 'yo',
    ↪ label='Experimental data')
    plt.xlabel('Degrees')
    plt.ylabel('Number of nodes')
    plt.title('Logarithmic scale degree distribution of the
    ↪ CSD network')
    plt.show()

plot_deg_dist(K)
```

Listing 3: data$_s ets.py$: $This code analyses the networks with networkx.$

```
#This code creates a dictionary with CpG sites as keys and
↪ gene names as values.
# The code also reads in files with CpG sites and writes
↪ files with associated gene names.

count = len(open("cgToGenename.txt").readlines(  ))#
print(count)
#Opens file and makes the variable lists:
lists=[]
with open("cgToGeneName.txt", "r") as f:

    #descriptionsList= f.readline().split("\t") #Line two
    ↪ is the descriptions
    for lis in range(count): #Line 1 to the last,
        midlertidigList = f.readline().split("\t")
        lists.append(midlertidigList)

#Now, lists are all lines from line 3 in the dataset,
↪ starting with the cg number that has index 0.
print(len(lists))
```

```
#Function that allow you to choose indeces:
#source: https://www.oreilly.com/library/view/
# python-cookbook/0596001673/ch04s07.html
def select(lst, indices):
    return (lst[i] for i in indices)

#Make dictionary:
cgDict= {}
for i in range(len(lists)):
    L=lists[i]
    valuesTuple1= select(L, [0])
    valuesTuple2= select(L, [21])
    for i in valuesTuple1:
        for j in valuesTuple2:
            cgDict[i]=j
#print(cgDict['cg10031456'])

#open cg.txt:
count2 = len(open("cg.txt").readlines())#
print(count2)
#Opens file and makes the variable lists:
lists2=[]
with open("cg.txt", "r") as cg_f:
    for lis in range(count2): #Line 1 to the last, but not
    ↪  the very last that is empty

        midlertidigList2 = cg_f.readline()
        midlertidigList2=midlertidigList2[:-1]
```

```
        #print(midlertidigList2)
        lists2.append(midlertidigList2)


#open Neighbours_cg09736162.txt:
count62 =
↪  len(open("Neighbours_cg09736162.txt").readlines())#
#Opens file and makes the variable lists:
lists62=[]
with open("Neighbours_cg09736162.txt", "r") as
↪  Neighbours_cg09736162_f:
    for lis in range(count62): #Line 1 to the last, but not
    ↪  the very last that is empty
        midlertidigList62 =
        ↪  Neighbours_cg09736162_f.readline()
        midlertidigList62=midlertidigList62[:-1]
        lists62.append(midlertidigList62)


#open Neighbours_cg23173455.txt:
count55 =
↪  len(open("Neighbours_cg23173455.txt").readlines())#
#Opens file and makes the variable lists:
lists55=[]
with open("Neighbours_cg23173455.txt", "r") as
↪  Neighbours_cg23173455_f:
    for lis in range(count55): #Line 1 to the last, but not
    ↪  the very last that is empty
        midlertidigList55 =
        ↪  Neighbours_cg23173455_f.readline()
```

```
        midlertidigList55=midlertidigList55[:-1]
        lists55.append(midlertidigList55)


#open Neighbours_cg04542415.txt:
count15 =
↪  len(open("Neighbours_cg04542415.txt").readlines())#
#Opens file and makes the variable lists:
lists15=[]
with open("Neighbours_cg04542415.txt", "r") as
↪  Neighbours_cg04542415_f:
    for lis in range(count15): #Line 1 to the last, but not
    ↪  the very last that is empty
        midlertidigList15 =
        ↪  Neighbours_cg04542415_f.readline()
        midlertidigList15=midlertidigList15[:-1]
        lists15.append(midlertidigList15)

#cg10031456
#open Neighbours_cg10031456.txt:
count56 =
↪  len(open("Neighbours_cg10031456.txt").readlines())#
#Opens file and makes the variable lists:
lists56=[]
with open("Neighbours_cg10031456.txt", "r") as
↪  Neighbours_cg10031456_f:
    for lis in range(count56): #Line 1 to the last, but not
    ↪  the very last that is empty
        midlertidigList56 =
        ↪  Neighbours_cg10031456_f.readline()
```

```
        midlertidigList56=midlertidigList56[:-1]
        lists56.append(midlertidigList56)


#Now, lists2 are all lines from line 1 in the dataset,
 ↪  sincluding only the cg number
print("Length og list2: ", len(lists2))
print("list2[4]= ", lists2[4])
example=lists2[0]
print("example:" ,cgDict[example])


#write geneNamesNeighbours_cg09736162.txt
with open("geneNamesNeighbours_cg09736162.txt", "w") as
 ↪  geneNamesNeighbours_cg09736162_f:
    geneNamesNeighbours_cg09736162=""
    cgNotInDict62=0
    for i in range(len(lists62)):
        holder62 = lists62[i]
        if holder62 in cgDict.keys():
            geneNameNeighbours_cg09736162 =
             ↪  cgDict[holder62]
            geneNameNeighbours_cg09736162 += "\n"
            geneNamesNeighbours_cg09736162 +=
             ↪  geneNameNeighbours_cg09736162
        else:
            cgNotInDict62+=1
    geneNamesNeighbours_cg09736162_f.write\
        (geneNamesNeighbours_cg09736162)
    print(cgNotInDict62)
```

```
#write geneNamesNeighbours_cg23173455.txt
with open("geneNamesNeighbours_cg23173455.txt", "w") as
↪   geneNamesNeighbours_cg23173455_f:
    geneNamesNeighbours_cg23173455=""
    cgNotInDict55=0
    for i in range(len(lists55)):
        holder55 = lists55[i]
        if holder55 in cgDict.keys():
            geneNameNeighbours_cg23173455 =
            ↪  cgDict[holder55]
            geneNameNeighbours_cg23173455 += "\n"
            geneNamesNeighbours_cg23173455 +=
            ↪  geneNameNeighbours_cg23173455
        else:
            cgNotInDict55+=1
    geneNamesNeighbours_cg23173455_f.write\
        (geneNamesNeighbours_cg23173455)
    print(cgNotInDict55)

#write geneNamesNeighbours_cg04542415.txt
with open("geneNamesNeighbours_cg04542415.txt", "w") as
↪   geneNamesNeighbours_cg04542415_f:
    geneNamesNeighbours_cg04542415=""
    cgNotInDict15=0
    for i in range(len(lists15)):
        holder15 = lists15[i]
        if holder15 in cgDict.keys():
```

```
            geneNameNeighbours_cg04542415 =
             ↪  cgDict[holder15]
            geneNameNeighbours_cg04542415 += "\n"
            geneNamesNeighbours_cg04542415 +=
             ↪  geneNameNeighbours_cg04542415
        else:
            cgNotInDict15+=1
    geneNamesNeighbours_cg04542415_f.write\
        (geneNamesNeighbours_cg04542415)
    print(cgNotInDict15)

#10031456
#write geneNamesNeighbours_cg10031456.txt
with open("geneNamesNeighbours_cg10031456.txt", "w") as
 ↪  geneNamesNeighbours_cg10031456_f:
    geneNamesNeighbours_cg10031456=""
    cgNotInDict56=0
    for i in range(len(lists56)):
        holder56 = lists56[i]
        if holder56 in cgDict.keys():
            geneNameNeighbours_cg10031456 =
             ↪  cgDict[holder56]
            geneNameNeighbours_cg10031456 += "\n"
            geneNamesNeighbours_cg10031456 +=
             ↪  geneNameNeighbours_cg10031456
        else:
            cgNotInDict56+=1
    geneNamesNeighbours_cg10031456_f.write\
        (geneNamesNeighbours_cg10031456)
    print(cgNotInDict56)
```

```
#write geneNames.txt (ALL)
with open("geneNames.txt", "w") as geneNames_f:
    geneNames=""
    cgNotInDict=0
    for i in range(len(lists2)):
        holder = lists2[i] #ok:cg27665659

        if holder in cgDict.keys():
            geneName = cgDict[holder]
            geneName += "\n"
            geneNames += geneName
        else:
            cgNotInDict+=1
            geneName = "no name"
            geneName+="\n"
            geneNames += geneName



        #geneName=cgDict[holder]
        #print("gennavn", geneName)


    geneNames_f.write(geneNames)
    print('CpG sites that are not found in dict.: ',
    ↪  cgNotInDict)
```

Listing 4: cgDict.py: This code creates a dictionary with CpG sites as keys and gene names as values. The code also reads in files with CpG sites and writes files with associated gene names.

Appendix B

A selection of gene names

cg09736162 and neighbours	cg23173455 and neighbours	cg04542415 and neighbours	cg10031456 and neighbours
CELSR3	INPP4A	ZFP64	LRRC39
TIGD7	PZP	SPIC	KRT39
FAM107A	KBTBD10	MPDZ	PDC
SACS	G6PC2	MYOM1	MIR548F1
DNMBP	FAM71F1	CPA5	C4orf17
ANXA13	C16orf78	C10orf129	
JRKL	ATP10A	SFRS2IP	
CTNNAL1	IL1F7	SFRS12	
TMPRSS11B	ELAC1	SH3TC2	
SFRS18	CXorf23	CDH12	
CCR9	EFNA3	PYDC2	
MKS1	JMY	PTPN3	
POU1F1	IL1F5		
PIK3C2A	GNRH1		
SDAD1	KCTD1		
HIATL1			
BLZF1			
NME7			
LPAL2			
RAD51L3			
IL1RL1			
TAS2R5			
LOC285780			
LY86			
PGBD4			
C15orf24			
NPL			
CCDC14			
SIRT1			
MAPK8			
MEP1A			
LARP7			
MPDZ			
XRCC2			
PLCE1			
FAIM2			
GCM1			
SLC2A9			

The table lists the gene names of the four hubs with highest degree and their neighbours. The conversion from CpG-sites to gene names was done with a data table published by Illumina Inc. in the code in Listing 4 in Appendix A [28].