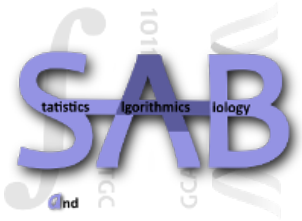


Network inference on *Bacillus subtilis* expression data

Internship Report
4GMM

Léa Grima
Applied Mathematics
National Institute of Applied Sciences (INSA) of Toulouse

Tutors : Nathalie Vialaneix, Elise Maigné, Simon De Givry, Anne Goelzer, Céline Brouard

INSA Tutor : Simona Grusea

October 14, 2022

Acknowledgments

I would first like to thank my tutors Nathalie Vialaneix, Elise Maigné, Simon De Givry, Anne Goelzer and Céline Brouard for their wise advices, for teaching me a lot through their experiences and for their trust during this internship.

I would also like to thank all my teachers from the Applied Mathematics department of the INSA Toulouse whose lessons allowed me to accomplish the tasks that have been given to me.

Finally, I would like to thank all the MIAT unit for their warm welcome during those four months.

Contents

Appendixes table **37**

1 Laboratory

The National Research Institute for Agriculture, Food and Environment (INRAE) is a Public Scientific and Technical Research Establishment (EPST). It is the result of the recent merger of the National Institute for Agricultural Research (INRA) and the National Research Institute of Science and Technology for the Environment and Agriculture (IRSTEA). It is under the joint aegis of the Ministry of Higher Education, Research, and Innovation (MESRI) and the Ministry of Agriculture and Food (MAA). INRAE is also beginning to closely collaborate with the Ministry for the Ecological and Inclusive Transition (MTES) via various partnership agreements. It is divided in 18 regional centres with 14 research divisions and has several missions:

- produce and disseminate knowledge to help solve major societal challenges,
- put this knowledge to work to foster innovation,
- provide expertise and lend support to public policies.

The sustainable development promotion is an overarching goal of the institute and main research fields are agriculture, food and environment. The internship took place in Toulouse Applied Mathematics and Informatics (MIAT) division. More precisely in the Statistics and Algorithms for Biology (SaAB) team, which aims to develop mathematics, statistics and computer science methods in order to solve problems from the field of molecular biology.

2 Context and objective

The objective of the internship was the use of different statistical methods to reconstruct the regulatory network of *Bacillus subtilis* from expression data. A ground truth network (built by biologists) was also provided to allow for the comparison of the method performances.

During the internship, different types of data were analysed: on one hand, the expression data of *B. subtilis* genes, and on the other hand, the regulatory network. In order to understand the data set, definitions and context are given in this section.

2.1 Biological context

As the principal data used during the internship are the expression measures of the genes, this section explains some biological and genetics concepts.

Deoxyribonucleic Acid (DNA)
The deoxyribonucleic acid (DNA) is the molecule that carries genetic information and instructions for development, functioning, growth and reproduction of an organism. It is made of two strands that coil around each other and form a double helix. Each strand is a sequence of nucleotides and they are linked by covalent bonds between phosphate and sugar groups alternating. These nucleotides are made of three elements:

- a phosphate group,
- a sugar group,
- a nitrogen base.

There are four types of nitrogen bases: adenine (A), thymine (T), guanine (G) and cytosine (C). The bases of each strand are linked by pairing rule: adenine bonds with thymine, and cytosine bonds with guanine as shown on Figure 2.

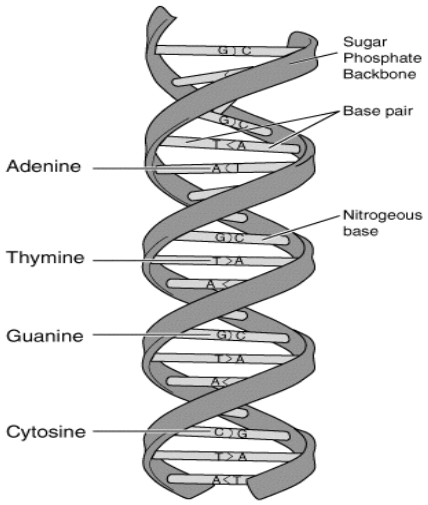


Figure 2: Structure of the DNA. By AutisticPsycho2, 2006, Wikimedia Commons (`https://commons.wikimedia.org/wiki/File:DNA-structure-and-bases.png`)

The strands of DNA encode biological information that allows messenger ribonucleic acid (mRNA) synthesis by using the enzyme RNA polymerase and then translation of these mRNA into proteins.

Ribonucleic acid (RNA)
The Ribonucleic acid (RNA) appears during the process of DNA transcription. It is made of the same biological materials as DNA up to two differences: the thymine is replaced by uracil (U) and the RNA is composed by a single strand. There are various types of RNA:

- the messenger RNA (mRNA) that carries the protein information;
- the transfer RNA (tRNA) that is involved in the translation of the mRNA in an amino acid sequence;
- the ribosomal RNA (rRNA), principal component of ribosomes (*i.e.*, the molecular machine that translates mRNAs into proteins).

As shown on Figure 3, RNA is the result of the DNA transcription. Then, the mRNA is translated by ribosomes and the tRNA is translated into an amino acid sequence which, by folding, will form a functional protein. Thereby, RNA plays an essential role in protein synthesis.

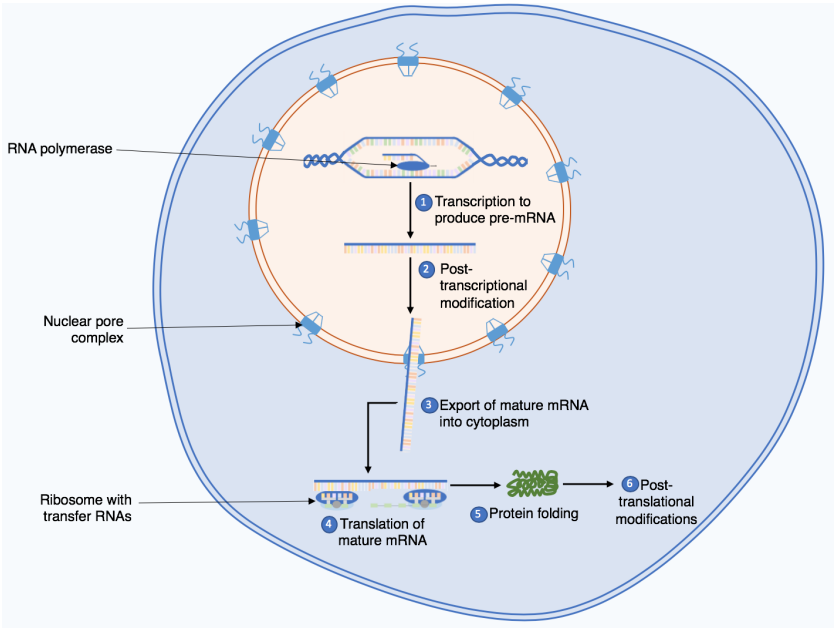


Figure 3: Protein synthesis process from DNA transcription to protein folding. By Kep17, 2020, Wikimedia Commons (`https://commons.wikimedia.org/wiki/File:Summary_of_the_protein_biosynthesis_process.png`)

In this report, the expression is measured by the produced quantity of mRNA by using tiling array. The principle of such an array is explain below, after that the description of the DNA transcription for the specific case of bacteria is given.

Transcription mechanism for bacteria
The DNA transcription mechanism is special for bacteria since bacteria are prokaryotes (organisms having cells without nucleus). Thereby, the first difference with the process

for eukaryotes (organisms having cells with nucleus) is that DNA translation and transcription are made at the same location (the cytoplasm) and simultaneously, whereas for eukaryotes, DNA transcription occurs in the nucleus and translation in the cytoplasm. Moreover, bacteria have only one type of RNA polymerase whereas there are three types for eukaryotes.

Firstly, bacterial RNA polymerase associated with an adding subunit, called sigma factor (σ factor), can recognize binding sequences in DNA. Those sequences are called promoters and the binding of a σ factor initiates the transcription process. The presence of σ factor is another difference with eukaryotes, which do not need them.

Many promoters control a sets of genes that work together: such structure is called an operon, see Figure 4. The genes in an operon are transcribed as a group and have a single promoter.

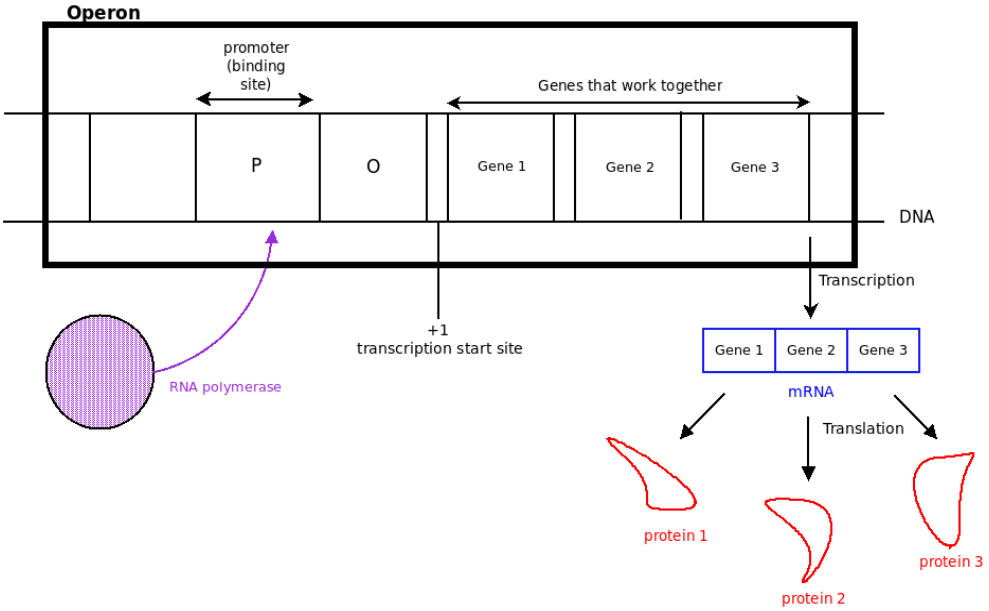


Figure 4: Structure of an operon.

Furthermore, an operon can contain regulatory DNA sequences that represent binding sites for regulatory proteins, called transcription factors (TF). The TFs promote or inhibit the transcription of a given operon and so control which genes are expressed and at which level. Therefore, the regulation for a lot of genes is done by the transcription mechanism.

Besides, each operon can be turned on or turned off depending on the bacteria needs. Some operons are necessary for the bacteria life: for example the operon that is related to the bacteria flagellum is needed for the movement. The proteins that inhibit the transcription are called repressors and they bind on sites called operators. On the contrary, the proteins that increase the transcription of the operon are called activators. These mechanisms are illustrated in Figure 5.

Tiling array

Tiling arrays are micro-array chips used to measure the expression of a large number of genes simultaneously or to genotype multiple regions of a genome. Here, the aim is to obtain a transcriptome mapping, that is to say, to recover to which extent each gene is expressed. The principle is to measure the quantity of mRNA by using the pairing properties of nucleotides. On the tiling array, there are microscopic spots and each spot

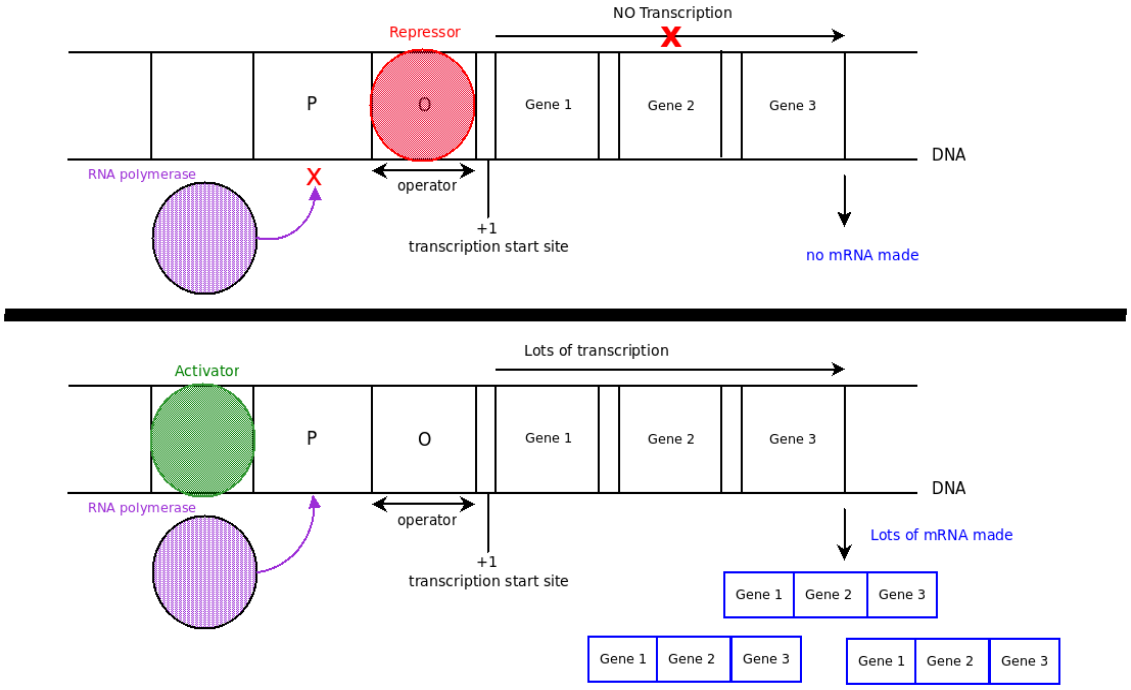


Figure 5: Regulation of an operon by a repressor or an activator.

corresponds to a DNA sequence called a probe. If the target strand is complementary, it will create covalent bonds with the DNA of the micro array spots. This induce a fluorescence/radioactivity that permits the measurement of expression. Thereby, the more fluorescent a spot is, the more expressed the gene corresponding to its probe is.

Now that biological definitions have been given, some notions for graph theory are explained in the next section.

2.2 Graph theory

Some definitions are needed to ease the understanding of the topic. They are presented in this section.

Graph

A graph, or network, is a pair $\mathcal{G} = (V, E)$ where:

- V is a set of vertices/nodes $\{x_i\}$ that can be connected or not;
- $E \subset V \times V$ is a set of paired vertices, called edges.

The edges can be oriented and some nodes can have an edge to themselves (called loop). Graphs are sometimes weighted, and the $(|V| \times |V|)$-matrix of edge weights is often noted W.

Another way to represent a graph is by an adjacency matrix A of size $N \times N$ where N is the number of nodes. A non-zero coefficient of the matrix corresponds to an existing edge in the graph. A is such that:

$$A = \begin{cases} 1 \text{ if } (x_i, x_j) \in E \\ 0 \text{ otherwise} \end{cases} \tag{1}$$

In the weighted case, the adjacency matrix is exactly W. For undirected graphs, A should be symmetric, *i.e.,* $A(x_i, x_j) = A(x_j, x_i)$.

Other definitions are given below to help understand the next sections of the report.

Graph properties

- Connectivity: a graph is said to be connected when any vertex can be reached from any other by a path along the edges. If a graph is not connected, there are several connected components that are maximum connected sub-graphs.
- Density: the density of a directed graph is the number of edges divided by the number of pairs of vertices. For undirected graphs without loops, the number of edges is divided by $\frac{n(n-1)}{2}$ with n the number of vertices.

Vertex properties

- Degree: the degree of a vertex is the number of edges adjacent to it, that is to say the number of edges to which the vertex belongs. Vertices with high degrees are called *hubs*. The degree is a measure of the vertex popularity.
- Betweenness: The betweenness of the vertex is the number of shortest paths between two vertices that pass through it. It is a measure of the vertex importance in the connectivity of the graph.

2.3 Data presentation

2.3.1 Expression Data

The data used in this internship comes from the bacteria *Bacillus subtilis*. This bacteria is a model organism. It is one of the most studied organism because of its properties. It is harmless for Humans but can serve as a model for the study of pathogenic bacteria of the same nature, like *Staphylococcus aureus*. In particular, it secretes an enzyme used in the industry. Also, it presents a large panel of growth phases and different living states presented in Figure 6.

Those properties are relevant for Human, for example the competence is a special property that is not present in all bacteria. It allows the bacteria to incorporate DNA from another species in its own chromosomes without damages. It can be interesting when an animal species produces a particular protein that is useful in the medical field. Instead of using or even killing the individuals, an option can be to first try to find the part of DNA that encodes such protein and then, to give this sequence to the bacteria in the attempt to make it reproduce the given protein.

The *B. subtilis* genome contains 4,200 genes. Here, the data set includes gene expression collected from 269 experiments in the BaSysBio European project (Nicolas et al. (2012)). In this project, biologists working on *B. subtilis* collectively defined a set of 104 environmental conditions to get as many expressed genes as possible. The used technology to measure gene expression was tiling array, already described in section 2.1.

2.3.2 Network and inference

I was also given a graph that was a gene regulatory network (GRN). It aimed to represent a set of genes that can interact with each other. In this case, the vertices are the genes and the edges represent the regulation link between two genes.

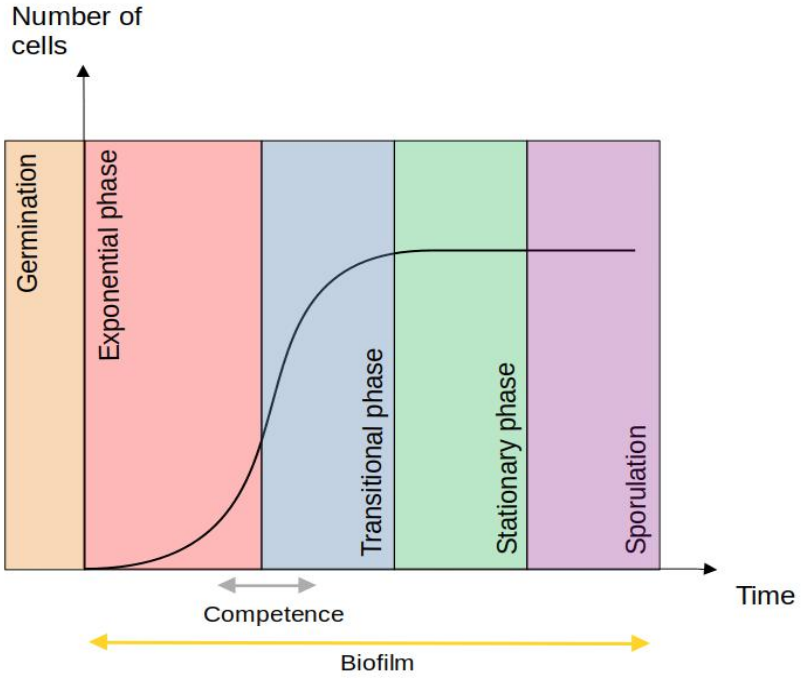


Figure 6: Growth phases of the bacteria *B. subtilis*

The given regulatory network is considered as the gene regulatory network of reference, noted GRN^r. It was built from the DBTBS database (Sierro et al., 2008), which gives experimental information, and from the reading of the literature. Moreover, a marginal part of the network was also obtained by a bioinformatics algorithm called RegPrecise (Novichkov et al., 2010).

GRN^r contains 3,977 vertices (genes) and 10,172 edges. Among these genes, there are 18 σ factors, 179 transcription factors, and 79 other mechanisms for regulation. The GRN^r is not connected and has two connected components. The largest connected component contains 3,968 genes, which represent 99.77% of all vertices.

Moreover, there are genes with high values of degree and betweenness. Most of them are σ factors and once again it confirms their importance in the regulation process. All of them are present in the largest connected component of the GRN^r.

The main purpose of the internship was the evaluation of various statistics methods in the reconstruction of GRN^r from the expression data described before. This part of the work is called inference and the different methods used are presented below, in Section 4.

Since each method provides an inferred network, it allows the measure of their respective performances by comparing the inferred networks and the GRN^r.

3 Exploratory analysis of the data

Firstly, an exploratory analysis was done on the data to have a better understanding of their distribution and to know how to manage them. This exploration was divided into three parts: a descriptive analysis on expression data to start, followed by a principal component analysis (PCA) and finally a visualization and analysis of the regulatory network built by biologists.

3.1 Descriptive analysis

The data set contained the expression data of 3,977 genes collected from tiling arrays in 269 experiments conducted in 104 different conditions. A descriptive analysis was done on these data to detect possible outliers. To do so, the minimum and maximum values for each gene were plotted and none were found extreme so we deduced that there was no outlier. The presence of missing values was also checked and there was none in the studied set.

3.2 Principal component analysis

The second part of the exploration work was a principal component analysis (PCA) with the 269 individuals representing the experiments and the 3,977 variables representing the genes. This was done by using the `PCA` function of the `FactoMineR` package. The aim was to summarize the initial data in a relevant way to be able to represent it in sub-spaces with reduced dimensions. PCA results indicated that the expression data were in accordance with biological knowledge: most of the groups of experiments were well separated on the plot of individuals. In addition, no experiment was detected as an outlier.

3.3 Graph mining

The last part of the exploration was to visualize and analyse the given network. In order to handle the structure of the graph, the package `igraph` was used for both visualization and analysis. The regulatory network is not connected: it is made of 5 components, the biggest one contains 3,968 genes and the other contain isolated genes. Its density is equal to 0.0013, which means that it is sparse. Moreover, there are 136 loops in the GRN^r that shows the presence of genes that regulate themselves (see Figure 7).

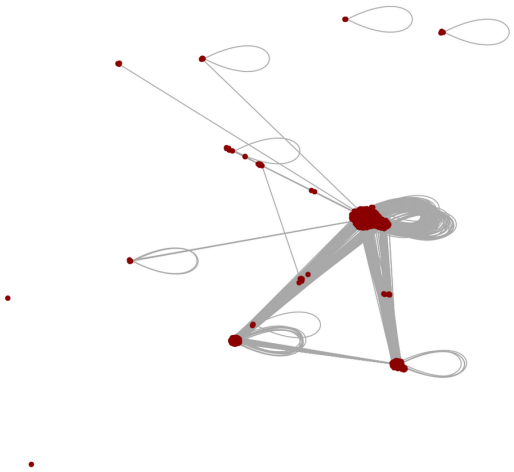
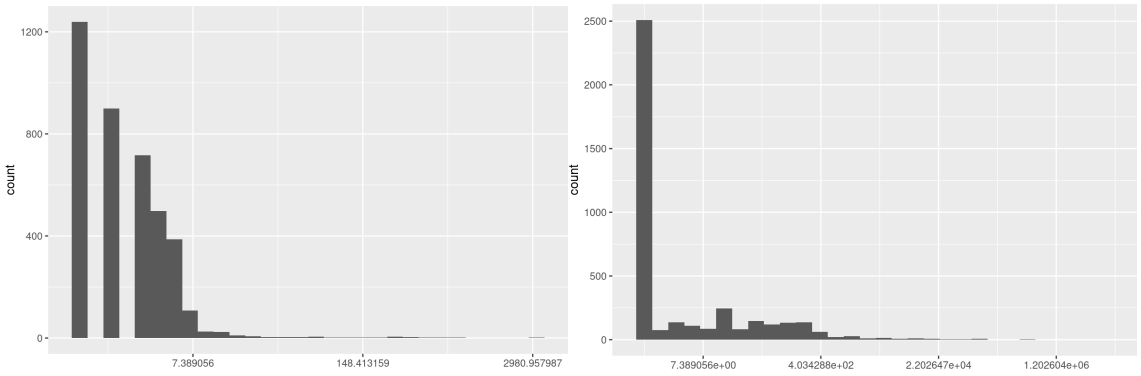


Figure 7: GRN^r representation

After the first visualization of GRN^r in Section 3.3, the aim was to detect important genes in GRN^r. To do so, the degree and betweenness of each nodes were calculated. The distribution was plotted and the values were sorted to detect the genes with the highest degrees and betweennesses.

As shown on Figure 8, GRN^r has skewed degree and betweenness distributions (there is a small number of genes with very high degree and betweenness, probably those implicated in the regulation of many others and, on the contrary, there is a large number of genes with small values of degree and betweenness that are probably those that are regulated but not regulator). This was expected since, in general, GRN have this type of distributions.



(a) Degree (b) Betweenness

Figure 8: Degree (a) and betweenness (b) distribution of GRN^r

Figure 9 shows the relation between degree and betweenness (in log-scales): most of the time, a gene with a high value of its degree also has a high value of its betweenness.

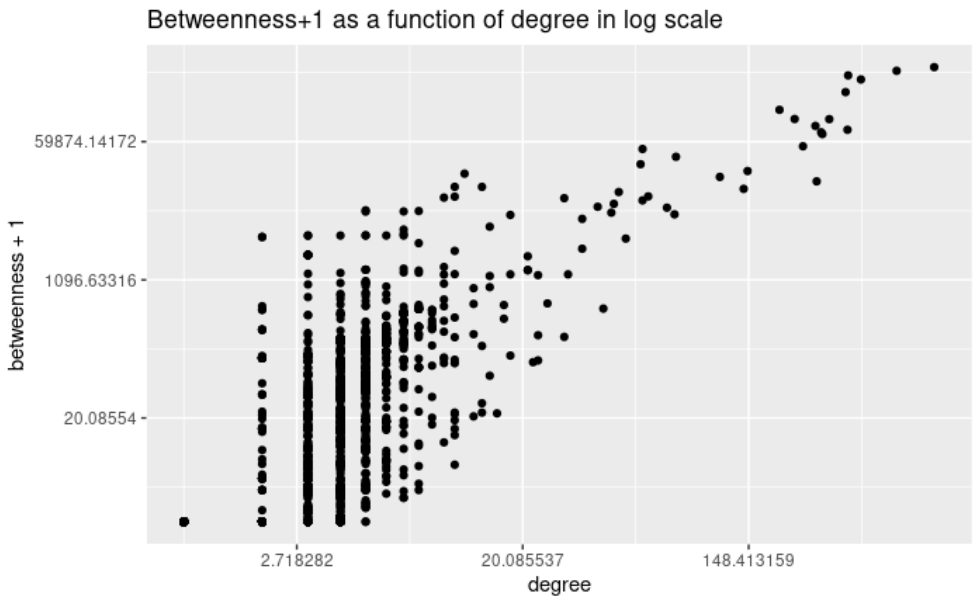


Figure 9: Betweenness+1 as a function of degree for GRN^r in logarithmic scales.

As a proof of their importance in the regulation, these genes were found to often be σ factors. In particular, there was a σ factor named σ_A that was linked with 3,203 genes over the 3,977 present in the network and is thus involved in the regulation of the majority of the studied genes. Moreover, when we compared the degree distributions between σ

factors and the other genes (see boxplot in Appendix 1.1), we saw that the degrees were much larger for σ factors compared to other genes.

4 Methods

In this part, we present the various methods used in order to infer a regulatory network from expression data. We started with the tree-based methods, then we used the "naive approach" based on Pearson correlation, followed by a partial correlation and information theory approach (PCIT).

4.1 Tree based methods

The first used methods are based on regression trees and more precisely on Random Forests. Thereby a definition of Trees in the statistical sense is given below. We also introduce Bootstrap and Bagging that serve in the Random Forest method.

4.1.1 Trees

Global definition
Classification and regression tree is a non parametric method used in classification and regression to predict a response variable Y from p explanatory variables denoted $(X^i)_{i=1,\dots,p}$ observed on n individuals $(X_j)_{j=1,\dots,n}$.

Trees are based on successive divisions of the data according to an explanatory variable at each node. For a node based on X^i, the split is defined by a threshold or a split into two groups of modalities depending on the fact that X^i is numeric or categorical. The split is binary and gives two child nodes.

There are some requirements in this method:

- definition of a criterion to select the best split at each node κ;
- definition of a rule to end the divisions and obtain a *leaf*.

Choose the best split comes down to finding the explanatory variable X^i and the threshold t that minimize the sum of child heterogeneity defined below. The given problem is:

$$\arg\min_{(i,t)} D_{\kappa_L}(i,t) + D_{\kappa_R}(i,t) \tag{2}$$

where the threshold t defines resulting sub-regions, $\kappa_L(i,t)$ and $\kappa_R(i,t)$, called child nodes and such that:

$$\kappa_L(i,t) = \{j : \ X_j^i \leq t\}, \qquad \kappa_R(i,t) = \{j : \ X_j^i > t\}$$

This splitting procedure is done for every node κ until the stopping rule is satisfied.

Since expression data are numerical, we use **regression trees** in our application. Thereby, the selection criterion and stopping rule are given for this case.

Selection criterion for regression trees
The heterogeneity is a non negative function defined for each node κ. It is differently defined in regression and classification. In regression, the heterogeneity criterion is the variance, given in Equation (3). The heterogeneity is equal to 0 when the given node is homogeneous, which means that all individuals have the same value for the response variable Y. On the contrary, the heterogeneity increases when values of Y for individuals of the nodes have a large variance.

$$D_\kappa = \sum_{k\in\kappa} (Y_k - \overline{Y}_\kappa)^2 \quad \text{with} \quad \overline{Y}_\kappa = \frac{1}{|\kappa|}\sum_{k\in\kappa} Y_k \tag{3}$$

The selection criterion corresponds to the maximization of the decrease in heterogeneity as described in Equation (4), where κ_L (left node) and κ_R (right node) are the child nodes of κ induced by a given split on a given variable X^i.

$$\max_{\{\text{Splits of } X^i;\ i=1,\dots,p\}} D_\kappa - (D_{\kappa_L} + D_{\kappa_R}) \tag{4}$$

Solving this problem is equivalent to solving the one presented in Equation (2).

Stopping rule

The stopping rule is satisfied when a given node κ is homogeneous or when it contains less individuals than a set value (generally between 1 and 5) to avoid too fine splitting. The defined final nodes of each tree are called *leaves* and contain the predicted values of Y.

The predictions are done by a majority vote for classification and by computing the mean estimate of all trees for regression:

$$\forall x,\ \hat{f}_B(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}_{z_b}(x)$$

However, trees are unstable and known to produce very different predictions with minimal changes in the training data set, especially when the growth of the trees makes them deep, as it is the case for the stopping rule presented in this section.

Bootstrap and Bagging

The principle of Bootstrap is to randomly draw data sets of size n from the original sample, where n is the number of observations of this sample (here, $n = 269$). In this way, the distribution of the data in bootstrap samples remains the same as in the original data. The drawing is done with replacement so the drawn sets can have duplicated values. This technique allows to obtain new data sets from the original one and to fit a model with each bootstrap sample that gives a predicted value, Y_{new} for any given new observation of $(X^i)_i$, $(X^i_{\text{new}})_i$.

Once the bootstrap samples built and the models that come with them fitted, an aggregation is done. The predictions of each model are combined by a majority vote for classification and by calculating an average for regression. This procedure is called the bootstrap aggregation or bagging.

Moreover, for each bootstrap sample, an Out Of Bag (OOB) sample can be defined, which contains the observations that were not drawn in the bootstrap sample.

The explanatory diagram on Figure (10) summarizes the process.

4.1.2 Original Random Forests

Instead of using individual trees, Random Forest uses a collection of the given model to predict the response Y by aggregating them with bagging. As the bootstrap samples are built on the same original sample, the predictors obtained for each bootstrap samples are not independent.

To improve bagging, a random component is introduced in the "random forest" method. More precisely, at each node, a random choice of m variables over the p available predictors is performed to make the aggregated trees more independent and the best split is chosen among these m variables only. The method is described below in Algorithm 1.

There are important hyper-parameters of the model:

- B, the number of bootstrap samples built from original data that is the number of individual trees that are fitted;

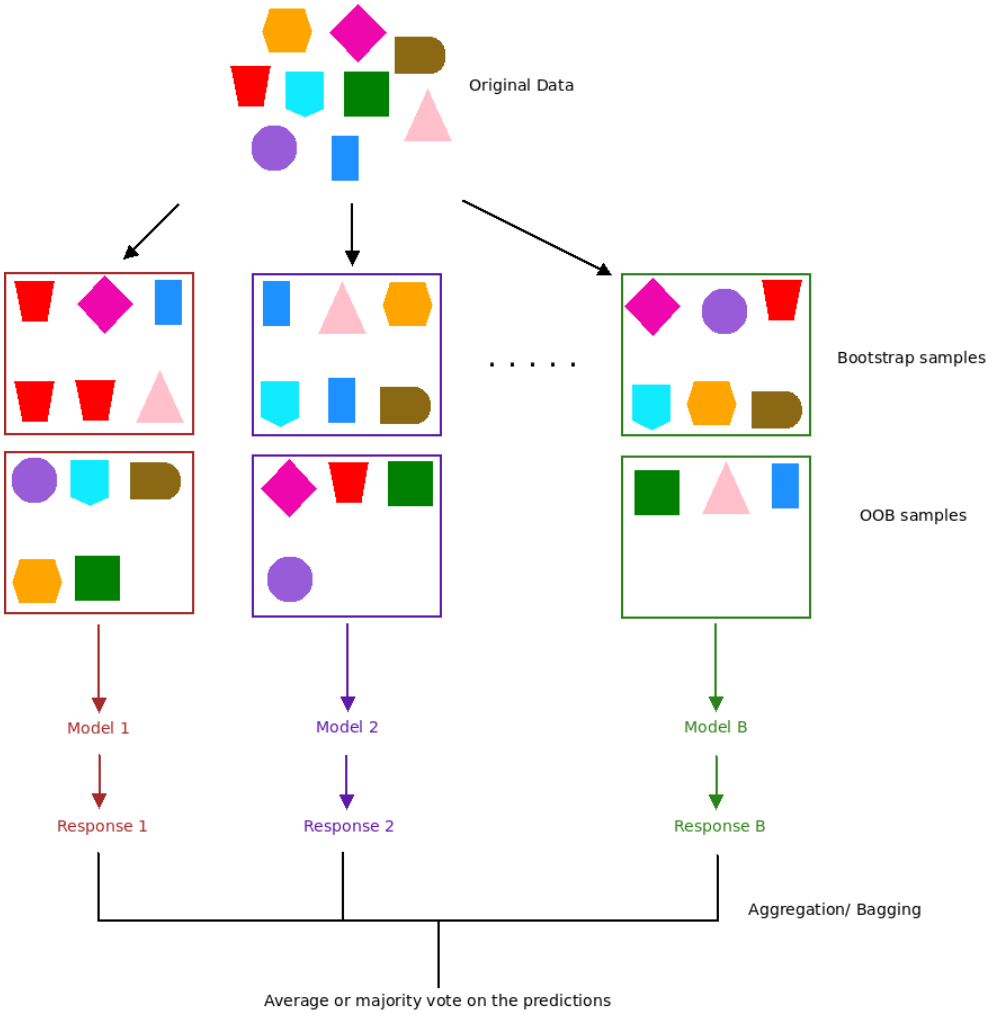


Figure 10: Principle of bootstrap aggregation or Bagging

– m the number of variables drawn randomly before each split. By default, in regression problems it is set to $p/3$.

Unlike trees, Random Forests are not easy to interpret and indices of importance are defined for each explanatory variable to find which variables are the best predictors.

Variable Importance

To evaluate the importance of a variable in the forest, an "Importance" measure called Mean Decrease Accuracy (MDA) has been defined. The aim is to measure the decrease of accuracy induced by a permutation of data by averaging it over all the trees $\hat{f}_{z_b}$ to evaluate the influence of a given variable on the predictions.

More precisely, for a given variable X^i and for the b^{th} tree of the forest, we use the OOB sample, S_b, and define S_b^i, the same sample in which the values of X^i are randomly permuted. Defining R_n as the percentage of well predicted values by a given tree in the OOB sample,

$$R_n(\hat{f}_{z_b}, S) = \frac{1}{\text{Card}(S)} \sum_{j:\,(X_j,Y_j)\in S} (Y_j - \hat{f}_{z_b}(X_j))^2. \tag{5}$$

Algorithm 1 Random Forests (regression case)

Let $Z = (X_1, Y_1), ..., (X_n, Y_n)$ a learning sample
for $b = 1$ to B **do**
  Take a bootstrap sample z_b from Z
  Fit a tree $\hat{f}_{z_b}$ from this sample in which the search for each optimal split is preceded by a random selection of a subset of m predictors.
end for
Calculate the mean estimate such as $\forall x$, $\hat{f}_B(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}_{z_b}(x)$.

the MDA is obtained by the average of decrease in accuracy calculated for all aggregated trees on original and permuted OOB samples:

$$MDA(X^i) = \frac{1}{B}\sum_{b=1}^{B}[R_n(\hat{f}_{z_b}, S_b) - R_n(\hat{f}_{z_b}, S_b^i)]. \tag{6}$$

X^i is going to be all the more important if the index is large. Indeed, large values indicate that the permutation changes the predictions and thus, that the variable X^i has an influence on predicted values.

4.1.3 Extremely Randomized Trees (Extra-Trees)

Many variants of the original random forest presented in Section 4.1.2 exist in the literature. Among them, we also used "Extremely Randomized Trees". There are two main differences:

- the Extra Trees use the whole initial sample multiple times instead of bootstrap samples;
- splits are made randomly by drawing uniformly at random several pairs made up of a random explanatory variable and a random threshold. The pair that maximizes the decrease in heterogeneity is considered as the best one and hence it defines the split.

4.1.4 `GENIE3` package

In this section, we present an adaptation of the random forest method that is used to infer networks. This method is implemented in the package `GENIE3` (Huynh-Thu et al., 2010).

In the described context, the objective is to infer a gene regulatory network: the GRN^r presented in Section 2.3.2. The expression of each gene can be seen as the value to predict and thus, as the response variable of a regression tree. Thereby, for each gene (variable), we fit B regression trees $\hat{f}_{z_b}$ and consequently, a random forest, RF:

$$\forall X^i, \qquad X^i = RF(\{X^k, \forall k \neq i\})$$

All genes different from the one to predict are considered as predictors in the model by default. Moreover, since $k \neq i$, tree-based methods can not recover loop in regulation.

Random forests are fitted for every possible gene to predict, $(X^i)_i$, and edges of the final network correspond to all pairs of predictors and predicted variables having the largest importance overall ("largest" is defined by a user-chosen threshold of the importance values). In addition, the importance in `GENIE3` is not exactly MDA but an adaptation of

this quantity that also accounts for the position of the variable in the different splits of the trees.

Finally, note that the presented method is parameterized by various settings with different roles. Some of the settings are related to the used method and others are hyper-parameters of this method. The most important settings are:

- the method: RF (Random Forest) by default or ET (Extra Trees);
- the number of variables, m, to draw randomly (for Random Forests) or the number of random possible splits to generate (for Extra Trees), before defining a split. By default it is equal to $\sqrt{p}$, with p the number of candidate regulators;
- the number of trees, B, by default it is equal to 1,000.

Other options allow to pass a regulator list and a target list. The first option is going to reduce the space of explanatory variables. The second one permits to reduce the number of variables to predict, therefore the number of random forests that have to be computed. In addition, the method is implemented such that it allows for parallel computation.

Finally, the function `GENIE3` returns a weighted adjacency matrix W, where $W_{k,i}$ is the importance of the link between gene k (regulatory gene) and gene i (target gene). A threshold can thus be chosen to keep the highest weights. We chose this value so as to obtain a number of inferred edges close to the number of edges in GRN^r.

Different tests were done to obtain the best possible inferred network with the tree-based methods. The values for m and B remained the default values for all of our tests. Variations of the settings were:

- both RF and ET were run on scaled and non-scaled data;
- both RF and ET were run with two different random seeds to assess reproducibility;
- RF was run with a restricted regulator list as input, which contained the list of σ factors;
- RF was run with another restricted regulator list, which contained the list of genes identified as regulators in our data.

4.2 Methods based on correlations

The second type of method that we used is based on correlations between variables.

4.2.1 Network inference based on correlation

The first method used was what we call the "naive approach" (also known as "relevance network" in the litterature (Butte and Kohane, 2000);(Butte and Kohane, 1999)). It consists in calculating the Pearson correlation between every pair of genes. Correlations are then thresholded by a user-defined value (keeping only correlations whose absolute value are above a given threshold). The kept pairs of genes defines the inferred edges. This method is close to the one implemented in the R package `WGCNA` (Langfelder and Horvath, 2008), which is frequently used by biologists for network inference.

Nevertheless, this approach is not completely satisfying since it does not detect the difference between "indirect" and "direct" correlations. As we see on Figure 11, when X^i and $X^{i'}$ are directly regulated by a same gene X^k, the Pearson correlation between X^i and $X^{i'}$ is expected to be large, on the same order of magnitude that the correlation between

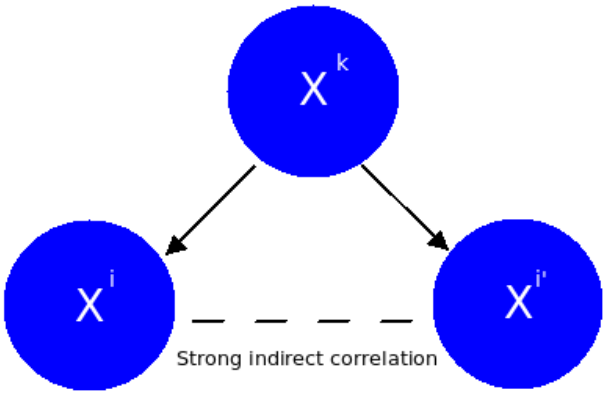


Figure 11: Example illustrating the limit of the use of the Pearson correlation.

X^i and X^k. The method is thus not able to distinguish between the direct link between X^k and X^i and the indirect link between X^i and $X^{i'}$. In a Gaussian framework, partial correlations are used instead of Pearson correlations to solve this issue.

4.2.2 Gaussian Graphical Model

Gaussian Graphical Models (GGM) assume that the vector $X = (X^i)_{j=1,\dots,n}$ containing gene expression is a Gaussian vector. This vector represents a centered multivariate normal distribution in $\mathbb{R}^p$, more formally: $X \sim \mathcal{N}(0_p, \Sigma)$, where 0_p is the null vector of $\mathbb{R}^p$ and Σ the positive definite covariance matrix of X.

The aim of GGM is to capture relevant partial correlations between two variables X^i and $X^{i'}$ given all the other variables of the data set. The partial correlation matrix, denoted by $C \in \mathcal{M}^{p\times p}$, is defined by:

$$\forall i \neq i' : \ i, i' \ \in \{1, \dots, p\}, \qquad C_{i,i'} = \mathrm{Cor}(X^i, \ X^{i'} | \{X^k\}_{k\neq i,i'})$$

Under given assumptions, one can prove (Edwards, 1995) that it exists a relation between $S = \Sigma^{-1}$ and C such as:

$$C_{i,i'} = -\frac{S_{i,i'}}{\sqrt{S_{i,i} S_{i',i'}}} \tag{7}$$

Thereby, the computation of the inverse of the covariance matrix, S, is needed to calculate the partial correlation. However, the (population) covariance matrix, Σ is unknown and, in high dimension ($p \gg n$), its empirical estimate, $\hat{\Sigma}$, obtained from the matrix of real observations $(X_j)_{j=1,\dots,n}$ is non-invertible (or ill-conditioned when p is smaller but close to n).

The problem of the estimation of the partial correlations is also equivalent to linear models. Indeed, given the linear model,

$$X^i = \sum_{k\neq i} \beta_{i,k} X^k + \epsilon \tag{8}$$

we can demonstrate that

$$\beta_{i,i'} = 0 \iff C_{i,i'} = 0 \iff S_{i,i'} = 0.$$

However, under this alternative framework, the problem remains the same: in the high dimension setting, the linear model of (8) is ill-posed and can not be estimated (*i.e.*, we can not find unique estimates of $\beta_{i,k}$, $\hat{\beta}_{i,k}$, from the observation $(X^i_j)_j$).

Two types of methods exist to overcome the high dimension issue: regularization and penalization methods.

Regularization consists in making the estimator $\hat{\Sigma}$ well-conditionned by adding a constant $\lambda > 0$ at each eigenvalues and then by taking the inverse:

$$\hat{S} = (\hat{\Sigma} + \lambda I)^{-1}$$

This solution is equivalent to solving a modified least square estimation of the linear model (8):

$$\hat{\beta} = \arg\min_{\beta_{i.}} \sum_{j=1}^{n} \left[Y_j^i - (X_j^{-i})^\top \beta_{i.} \right]^2 + \lambda \|\beta_{i.}\|_2^2$$

where:

- $(X_j^{-i}) \in \mathbb{R}^{p-1}$ is the vector X_j without its i^{th} component,
- $\beta_{i.} = (\beta_{i,k})_{k \neq i} \in \mathbb{R}^{p-1}$.

Usually, regularization methods return estimates that are all non-zero. A post-processing step is usually necessary to keep only the largest estimated partial correlations. This method is implemented in the package `GeneNet` (Schäfer and Strimmer, 2005).

In penalization methods, the estimator of Σ is not modified a l_1 penalization term is used (instead of l_2) to modify the least square estimation problem:

$$\hat{\beta} = \arg\min_{\beta_{i.}} \sum_{j=1}^{n} \left[Y_j^i - (X_j^{-i})^\top \beta_{i.} \right]^2 + \lambda \|\beta\|_1$$

where $\lambda > 0$ is the regularization hyperparameters.

When using l_1 penalization, some $\hat{\beta}_{i,k}$ are forced to be directly estimated as zero.

Two approaches exist to solve the previous problem: one is a global optimization framework, implemented in the package `glasso` (Friedman et al., 2008), and the other, by (Meinshausen and Bühlman, 2006), uses independent Lasso regressions. Both are implemented in the package `huge`.

However, these methods are time and memory needing. Hence, we chose to perform the inference with another method inspired by GGM and described in the next section.

4.2.3 Partial correlation and information theory (PCIT)

This second method is used for the inference and is based on partial correlations and on information theory. The aim is to identify relevant associations between genes that can correspond to edges in the inferred network. The main difference between PCIT and GGM is that only partial correlations of triplets of genes are computed, which is computationally easier. Nevertheless, the normality assumption is still relevant for PCIT.

The PCIT algorithm proceeds in two different steps to determine if there is a connection between two genes X^i and $X^{i'}$. First, the partial correlation coefficients are computed for all trio of genes formed by X^i, $X^{i'}$ and X^k. The expression of such coefficient is given in Equation(9). In fact, three Pearson correlation coefficients and three partial correlation coefficients are calculated each time we put a new X^k. Finally a local tolerance level ϵ_k is calculated with those partial correlation coefficients and the Pearson correlation coefficients, the formula is given in Equation(10). A condition is defined to establish a connection or not between X^i and $X^{i'}$. More formally:

Step 1
First-order partial correlation coefficients are computed for every trio of genes X^i, $X^{i'}$ and X^k. The given formula for partial correlation coefficient between X^i and $X^{i'}$ given X^k, denoted $r_{ii',k}$, is:

$$r_{ii',k} = \frac{r_{ii'} - r_{ik}r_{i'k}}{\sqrt{(1 - r_{ik}^2)(1 - r_{i'k}^2)}} \tag{9}$$

Where $r_{ii'}$, r_{ik} and $r_{i'k}$ represent the Pearson correlation coefficients between respectively X^i and $X^{i'}$, X^i and X^k, $X^{i'}$ and X^k. The given formula is:

$$r_{ii'} = \frac{\mathrm{Cov}(X^i, X^{i'})}{\sigma_i \sigma_{i'}},$$

with σ_i and $\sigma_{i'}$ the standard deviation of variables X^i and $X^{i'}$. r_{ik} and $r_{i'k}$ are calculated in the same way. This value gives the strength of the relationship between X^i and $X^{i'}$ that is uncorrelated with X^k. Similarly, $r_{ik,i'}$ and $r_{i'k,i}$ are obtained.

Step 2
A local threshold is calculated for every trio of genes in order to capture relevant associations. It is the ratio of partial to direct (Pearson) correlations. It is noted ϵ_k and is given by the next equation:

$$\epsilon_k = \frac{1}{3}\left(\frac{r_{ii',k}}{r_{ii'}} + \frac{r_{ik,i'}}{r_{ik}} + \frac{r_{i'k,i}}{r_{i'k}}\right). \tag{10}$$

Then PCIT deduces that there is an edge between X^i and $X^{i'}$ if and only if:

$$\forall\, k \neq i, i', \qquad |r_{ii'}| > |\epsilon_k\, r_{ik}| \text{ or } |r_{ii'}| > |\epsilon_k\, r_{i'k}| \tag{11}$$

The two described steps are repeated for every pair of genes in order to capture the meaningful gene to gene associations defining inferred edges.

4.2.4 `PCIT` package

The previous algorithm is implemented in the package `PCIT` (Reverter and K. F. Chan, 2008).

The used function takes the correlation matrix of the expression data as input and applies the PCIT algorithm on it. The output is a list with linear indices of the correlation matrix that indicates the pair of genes that verify the condition described in Equation(11). Every pair of genes found by PCIT defines an edge in the inferred network and so an inferred graph is obtained.

There are some options for parallelization but we did not use them because the running time on our data is short. As for the tree-based approach, three different simulations were done to obtain the best possible inferred network:

- The first simulation used default settings in `PCIT` function;
- The second simulation used the option "max" for the setting *tol* of the `PCIT` function, corresponding to take the maximum of the three components of the right hand term of (10);
- The last one was a variant of the first one where we used biological knowledge and restricted the final graph to the edges adjacent to σ factors.

4.3 Bayesian network approach

In this section, the third and last approach studied during the internship is presented: the Bayesian Networks. Unfortunately, due to a lack of time, I was only able to learn the general concepts for this method but not to use it on the datasets.

4.3.1 Global description

A Bayesian network (BN) is a probabilistic graphical model that represents the conditional dependencies of a set of random variables (vertices) with a directed acyclic graph (DAG). It can be discrete or continuous according to the nature of the variables. There are two ways of seeing a BN, the DAG gives a visual representation while the probabilities linked to dependencies between variables allow the interpretation of connections.

Therefore, to construct a BN we need to define two different objects:

- a DAG, which is a graph $\mathcal{G} = (V, E)$ also called the structure of the model;
- the probability tables for each variable conditionally to its causes also called the parameters of the model.

Both objects can be defined by experts or computed from the given data but in general, the structure come from expert knowledge and parameters are computed from observed data. In network inference, however, the DAG is what is learned from observed data, it represents the conditional independence between variables.

The aim of such representation is to describe the joint probability of the given set of vertices,

$$\mathcal{P}(V) = \Pi_{X^i \in V} \mathbb{P}(X^i |\ pa(X^i)),$$

with $pa(X^i)$ the parents of node X^i in the DAG. Hence, given the DAG and the distribution of every node conditional to its parent, the overall distribution of the nodes is also known.

In our work, we had to consider discrete BN, where the variables $(X^i)_i$ are assumed to take discrete values. Since, in our case, $(X^i)_i$ correspond to gene expression, a discretization step is required to transform the continuous values into discrete ones. This step is presented in Section 4.3.3.

4.3.2 Inference methods

As the objective of our work is to infer networks, methods used for structure learning in BN are briefly described in this section.

There are three types of algorithms to search the structure of a DAG from the data:

- Constraint-based algorithms as inductive causation (IC) (Verma and Pearl, 1991);
- Score-based algorithms (Koller and Friedman, 2009);
- Hybrids algorithms combining the previous two methods as Sparse Candidate (SC) (Friedman et al., 1999) and Max-Min-Hill-Climbing (MMHC) (Tsamardinos et al., 2006).

The one chosen here is the method based on the optimization of a scoring function (Trösser et al., 2021).

The principle is to attribute a score to each DAG explored by the method and to chose the one which maximizes the scoring function. In general, the DAG is initially empty and as iterations progress, edges are added to maximize the chosen score.

There are two main scores used for networks: the BIC (Bayesian Information Criterion) and the BDeu (Bayesian Dirichlet equivalent uniform), both based on the model likelihood. For both scores, two DAGs that represent the same conditional independence set have the same score.

4.3.3 Application

As explained before, we had to discretize the gene expressions before being able to use the BN model implemented by the SAaB team. It was decided to discretize them all in maximum 3 classes depending, for each gene, on how many modes were present in the expression distribution.

4.4 Evaluation of the methods

In this section, the evaluation criteria used to compare the methods are presented. Indeed, the various methods give various inferred network denoted by GRN^i and we have to compare them to GRN^r.

4.4.1 Global comparison

To compare the real and inferred networks in a global way, various elements are analysed for each inferred network:

- the connectivity and the density (Section 2.2);
- the number of edges;
- the number of genes in the largest connected component;
- the number of edges common between GRN^r and GRN^i, to know how many edges are recovered by the tested inference methods.

By using edges common to GRN^r and GRN^i we can compute the precision and recall associated with each method. On the one hand, the "precision" measures the quality of inferred edges by comparing edges common to GRN^r and GRN^i and edges of GRN^i (inferred edges).

$$P = \frac{\text{number of common edges}}{\text{number of inferred edges}}$$

On the other hand, the "recall" measures the capacity of the inference method to recover the true edges by comparing edges common to GRN^r and GRN^i and edges of GRN^r (real edges).

$$R = \frac{\text{number of common edges}}{\text{number of edges in GRN}^r}$$

4.4.2 Comparison of vertex properties

Then we look at the vertex properties in more details. The aim of studying the vertices properties in each network is to see if the inference methods allow to recover the *hubs* or the other important vertices of GRN^r. To do so, the degree and the betweenness of each vertex are computed for each inferred network and their distribution is compared to the one obtained for GRN^r.

Degree and betweenness distributions
First, we compare the distributions: in general, GRNs have skewed degree and betweenness distributions. The comparison of these distributions allows us to intuit the resemblances or divergences in the structure. We also plotted the betweenness as a function of the degree to assess if there is a relation between these two quantities.

Wilcoxon-Mann-Whitney test
Another way to see if the vertex properties are preserved is to perform a Wilcoxon-Mann-Whitney test to compare the rankings of the distribution for a given vertex characteristics.

Ranks are computed both for degrees and betweennesses and the Wilcoxon-Mann-Whitney test is used to compare ranks obtained in GRN^r and ranks obtained in GRN^i: tested H_0 hypothesis is thus "Ranks are identical in GRN^r and GRN^i".

Precision and Recall for σ factors After having calculated the global "precision" and "recall", we calculated P and R for each σ factors by using three different methods. Let us denote a given σ factor by σ_f:

– **First method (M1):** P is defined as the number of common edges afferent to σ_f in GRN^r and GRN^i divided by the number of edges afferent to σ_f in GRN^i. R is defined as the number of common edges afferent to σ_f in both networks divided by the number of edges afferent to σ_f in GRN^r;

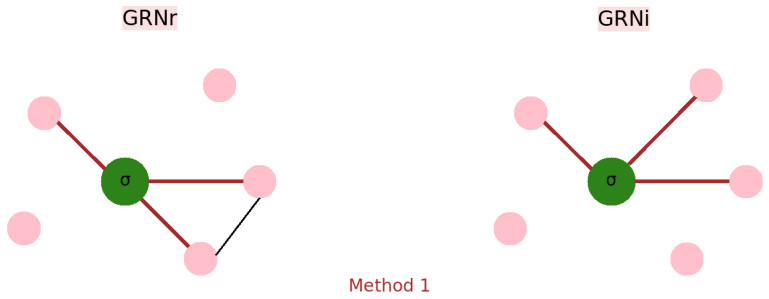


Figure 12: Example illustrating how **M1** (afferent edges) works.

– **Second method (M2):** Sub-networks containing vertices afferent to σ_f in GRN^r are extracted in GRN^r and GRN^i. P is defined as the number of edges common to the two given sub-networks divided by the number of edges in the sub-network of GRN^i. R is defined as the number of edges common to the two given sub-networks divided by the number of edges in the sub-network of GRN^r;

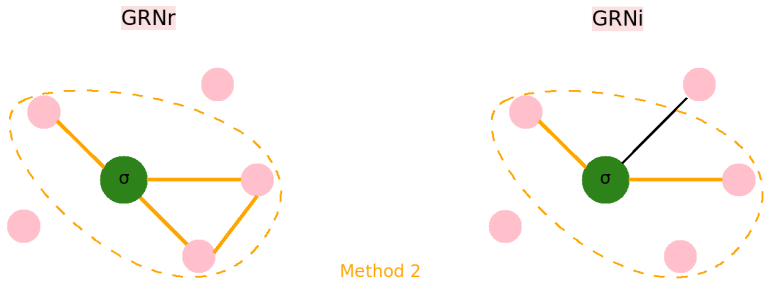


Figure 13: Example illustrating how **M2** (afferent nodes from GRN^r) works.

– **Third method (M3):** Sub-networks containing vertices afferent to σ_f in GRN^r and GRN^i. P is defined as the number of edges common to both sub-networks divided by the number of edges in the sub-network extracted from GRN^i. R is defined as the number of edges common to both sub-networks divided by the number of edges in the sub-network extracted from GRN^r.

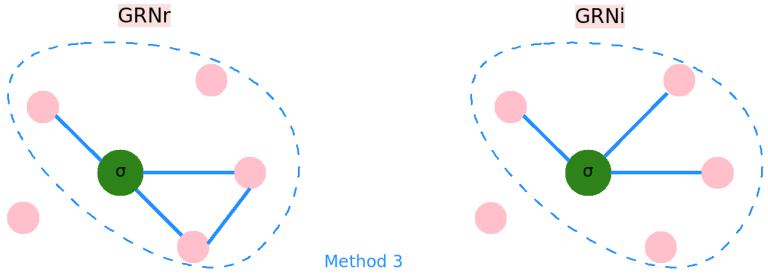


Figure 14: Example illustrating how **M3** (union of afferent nodes) works.

The values of R are similar in the second and third methods since only edges between nodes connected to σ_f differ. To represent the results, we used plot that give the precision as a function of the recall.

4.4.3 Clustering similarities

The comparison of GRN^r and inferred networks was also done by comparing the results of a node clustering algorithm.

In the graph context, a clustering consists in grouping a set of vertices in such a way that vertices of a same group are more similar with each other than with vertices of another group. Each obtained group of vertices is called a "cluster", a "community" or a "module".

To perform the clustering, several approaches exist:

– the optimization of a quality criterion;

– the spectral clustering;

– model-based clusterings.

The method used in our study is the first one with a criterion named the *modularity* (Newman and Girvan, 2004). It is a metric that measures the quality of a given node clustering. For a number K of clusters, the principle is to find the partition of vertices $(C_1, \ldots, C_K)$ which maximizes the following criterion:

$$Q(C_1, \ldots, C_K) = \frac{1}{2m} \sum_{k=1}^{K} \sum_{x_i, x_j \in C_k} (A_{ij} - P_{ij}),$$

with

– A_{ij} the coefficients of the adjacency matrix associated of the GRN;

– P_{ij} the coefficient of the adjacency matrix of a graph with same degree distribution, corresponding to a "null" model;

– $2m$ the number of edges of the GRN.

More precisely, we have:

$$P_{ij} = \frac{d_i d_j}{2m},$$

where d_i and d_j represent the degrees of node x_i and x_j. Hence, Q increases when (x_i, x_j) are in the same cluster and $A_{ij} \gg P_{ij}$. A high value of the modularity indicates that the connections are dense within a cluster and that the connections are sparser between clusters and the value of P is designed so that edges afferent to nodes with very high degree account less to inscrease modularity than edges afferent to nodes with small degree.

As the optimization of the modularity is NP-hard to solve, most methods using the modularity produces an approximated solution. The chosen approach is a multi-level greedy approach named the *Louvain algorithm* (Blondel et al., 2008).

Clusters of GRN^r and GRN^i are then compared by computing the Normalized Mutual Information (NMI) and the Adjusted Rand Index (ARI),two measures of similarity between clusterings. The more these indices are close to one, the more the compared clusters are similar.

In addition, cluster qualities are obtained by computing precision and recall for each cluster in GNR^i. Doing so means that the subgraph of GNR^r made from the vertices in the studied GNR^i cluster is extracted and used as a reference.

Clusters are finally explored in deeper details from a more biological point of view: in particular, the distribution of all regulatory gene combinations is computed for each cluster. The aim is to assess if the genes in a given cluster mostly share common regulators (which would give a biological meaning to the clusters).

5 Results and comparison of methods

The results for each method are presented in this section. Then a comparison is done by using different metrics and indices presented in Section 4.4. Analysis are slightly different from an inferred graph to another but globally we proceeded in the same way.

5.1 Clustering on GRN^r

In this section, we present the results of the clustering performed on GNR^r.

The Louvain algorithm, as implemented in the package `igraph`, was performed on the largest connected component of GRN^r (containing 3,968 genes). 21 different clusters were obtained (for a modularity equal to 0.51) and Table 1 gives the distribution of the number of nodes in every cluster. In cluster 14 to 21, there were only 47 genes in total, which represented only 1.18% of the genes in the largest connected component. The value of the modularity (0.51) is one that is usually considered as good.

Cluster	Number of genes	Cluster	Number of genes
1	219	12	168
2	1,286	13	57
3	184	14	5
4	284	15	5
5	332	16	15
6	586	17	7
7	307	18	5
8	327	19	2
9	106	20	4
10	36	21	4
11	29		

Table 1: Distribution of the number of genes by cluster in GRN^r.

To detect if the clusters were associated to particular biological mechanisms or to special groups of genes, we looked at the distribution of the combinations of regulator in every cluster. It appeared that in clusters 1 to 13 the majority of the combinations of regulator contained σ_A. For example, in the biggest cluster (number 2), more than 60% of the genes were regulated only by this σ factor as we see in Figure 15. Nevertheless, there was a lot of combinations of regulator that were only found once and concerned less than one percent of the genes in the cluster.

5.2 Inferred networks with tree-based methods

First type of method used to infer GRN^r were the tree-based ones.

In the present section, we restrict ourselves to the best network obtained from `GENIE3`, which turned out to be the one based on random forest (and not on extra trees), with unscaled expressions and where the predictors were restricted to be chosen within 18 σ factors only. This network will be named GRN$^{\mathrm{RF}}$ in the following and the threshold applied on importance was 0.09.

GRN$^{\mathrm{RF}}$ had 10,799 edges (density $\sim 0,1\%$, identical to the one of GRN^r, by design). It is not connected and its largest connected component contained 3,918 vertices. Over the 10,799 inferred edges, 2,133 were common to GRN^r. This number represented 20.97% of the edges in GRN^r (Precision, P) and 19.75% of the edges of GRN$^{\mathrm{RF}}$ (Recall, R).

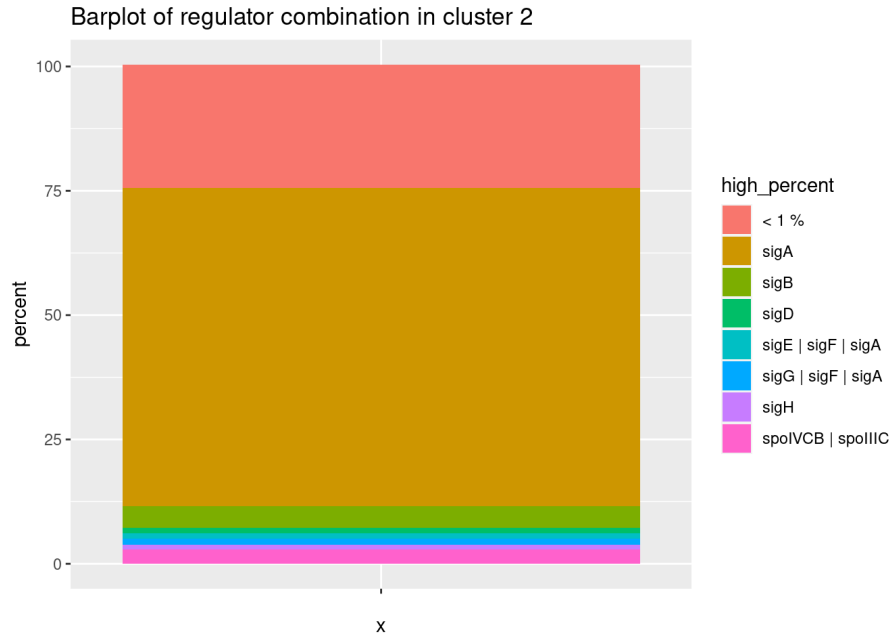
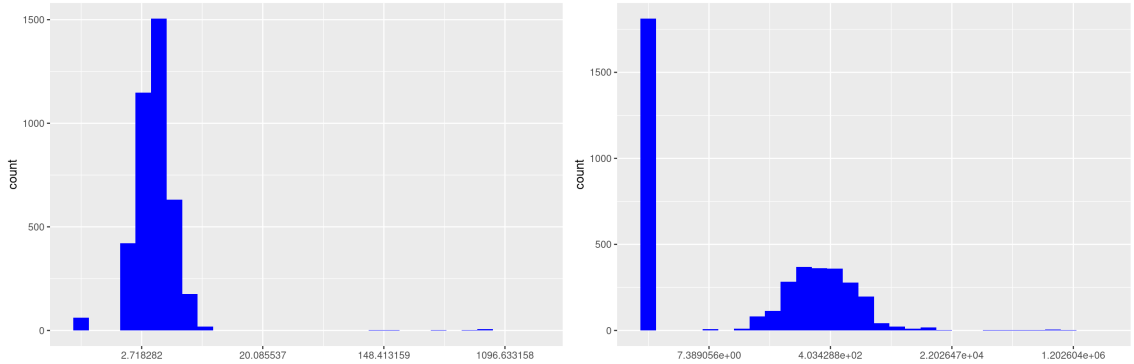


Figure 15: Distribution of the combinations of regulators in cluster 2 of GRN^r.

$$P = 0.2097 \qquad \text{and} \qquad R = 0.1975$$

These values are usually considered as good for the network inference problem (especially at genome scale).

The distributions of degree and betweenness in GRN^{RF} (Figure 16) were very similar to the ones of GRN^r (Figure 8), even if the highest values were very different (largest degree for GRN^{RF}: 1,706 – largest degree for GRN^r: 3,203).



(a) Degree (b) Betweenness

Figure 16: Degree (a) and betweenness (b) distribution of GRN^{RF}

However, the results of rank tests on degree and betweenness were not conclusive since the p-values were too high: 0.72 for the degree test and 0.34 for the betweenness test. Thereby, we could not say anything about the similarities in ranks between GRN^r and GRN^{RF}.

In addition, Appendix 2.1 gives the precision as a function of the recall for σ factors obtained with the three methods described in Section 4.4.2. We observed that, depending on the method used to calculate P and R, the values were different. With the first method

the tendency was that σ factors had precision below 0.5, except for σ_A that have a precision neat to one and a recall near to zero. Nevertheless, there were σ factors for which we had value such as:

$$P \in [0.2; 0.5] \text{ and } R \in [0.35; 0.6].$$

These values can be considered as good for inference problem in the given context. The σ factors concerned by these values were σ_B, σ_E, σ_F, σ_G and σ_K (`spoIIIC` and `spoIVCB`). These σ factors are known to be implied in the sporulation phase excepts for σ_B implied in stress management.

6 clusters were found in GRN^{RF} and the distribution of the number of vertices in every cluster is given in Table 2. Unlike GRN^r, there were no clusters containing very few genes. The modularity associated to this clustering was equal to (-0.01) which can be considered as bad. Moreover, the small values of precision and recall showed that the quality and the quantity of recovered edges were bad.

Cluster	Number of genes	Precision	Recall
1	524	0	0
2	1,087	0.07	0.03
3	614	0.01	0.01
4	754	0.05	0.03
5	860	0.03	0.03
6	103	0	0

Table 2: Distribution of the number of genes by cluster in GRN^{RF} and precision/recall values by cluster.

As in GRN^r, we looked at the combination of regulators present in each cluster to detect if there were any groups representing a biological mechanism. It appeared again that σ_A was omnipresent in every cluster. However, when we looked in details the cluster 5 (Figure 17), there are a lot of genes regulated by the σ factors implied in the sporulation phase with σ_A. In fact, σ_A alone regulated 40% of the 754 genes in cluster 4 and appeared in all combination representing more than 1% of the genes classified in the cluster 4 (Appendix 2.2).

We computed the Normalized Mutual Information and the Adjusted Rand Index to compare inferred and real clusters we obtained small values,

$$\text{NMI} = 0.017 \text{ and } \text{ARI} = 0.008.$$

It signified that there were no similarities between the clusters made in GRN^{RF} and in GRN^r.

Finally, we visualized the distribution of GRN^r clusters in GRN^{RF} clusters (Figure 18) to see if there was a matching between them.

None of the inferred clusters corresponded to only one cluster of GRN^r.

5.2.1 Other networks inferred by `GENIE3`

The results obtained for the other networks inferred by GENIE3 are recapitulated in Appendix 2.3. There were similar whether the method used was Rf or ET and whether data were scaled or no.

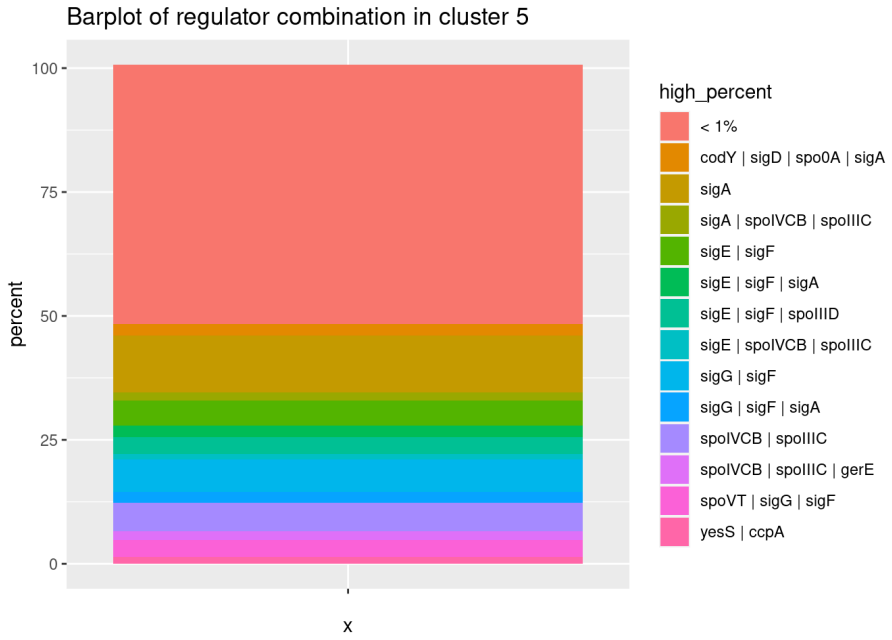


Figure 17: Distribution of the combinations of regulators in cluster 5 of GRN^{RF}.

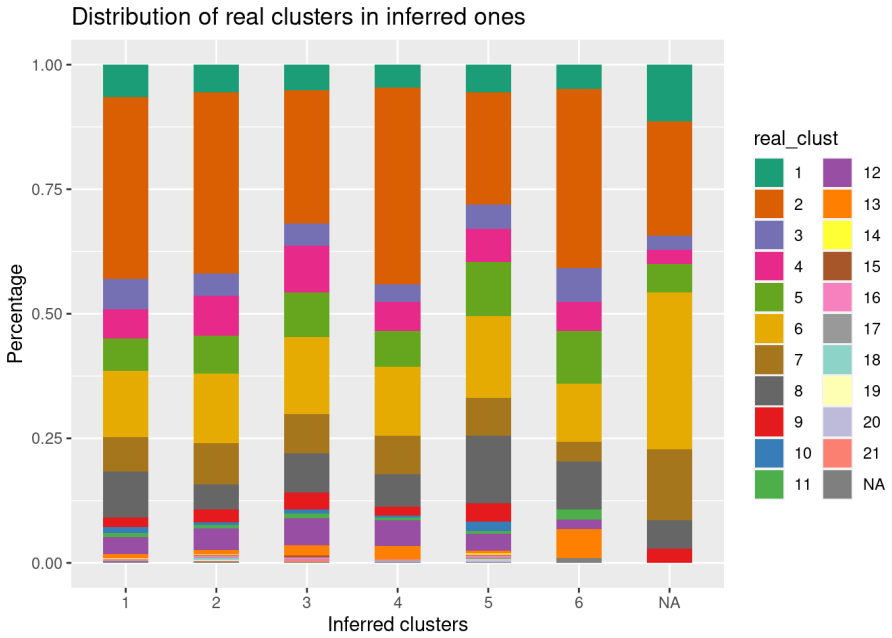


Figure 18: Distribution of GRN^{r} clusters in GRN^{RF} clusters.

5.3 Inference by methods based on correlations

Second type of methods used to infer GRN^{r} were the ones using correlations between variables. We started with the inference based on Pearson correlations and continued with the inference based on PCIT algorithm.

5.3.1 Inferred network using Pearson correlations

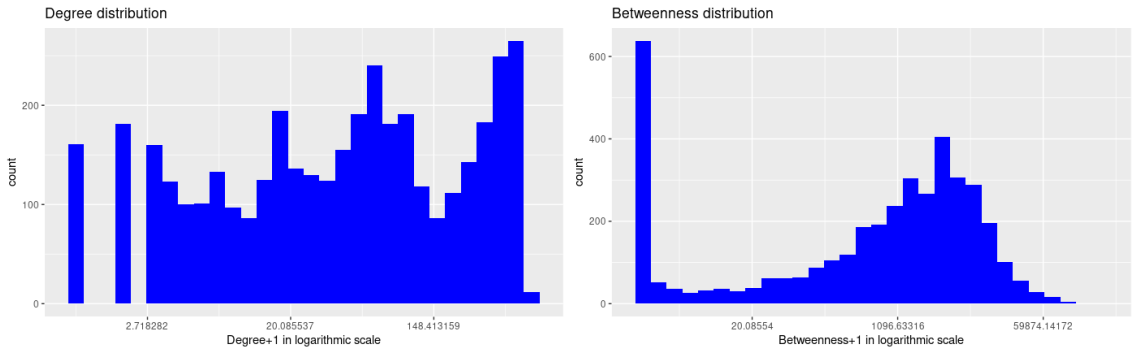
In this section, we study the "relevance network" constructed by thresholding the Pearson correlations from the correlation matrix. The threshold was set at 0.75 to obtain enough genes in the largest connected component and compare it with GRN^r. This network will be named GRN^{cor} in the following.

GRN^{cor} had 222,051 edges (20 times more than in GRN^r) and a density equal to 2.8% (higher than in GRN^r as a consequence). Over more than 200,000 inferred edges, 1,208 were commons to GRN^r. By using this number, we were able to calculate the global precision, P and recall, R:

$$P = 0.0054 \qquad \text{and} \qquad R = 0.1206$$

The value of precision allowed to deduce that this method inferred too much edges because it was under 1%. Nevertheless, 12% of the edges in GRN^r are recovered. Moreover, GRN^{cor} is not connected and its largest connected component contains 3,553 nodes.

This time, the distributions of degree and betweenness (Figure 19) were totally different from the ones of GRN^r (Figure 8). In fact, there were a lot of genes with high degree and high betweenness because of the high number of edges. This came from the chosen threshold that allowed to keep a lot of correlation even the ones that were indirect. Again, the highest values were very different (largest degree for GRN^{cor}: 581 -largest degree for GRN^r: 3,203).



(a) Degree (b) Betweenness

Figure 19: Degree (a) and betweenness (b) distribution of GRN^{cor}.

However, a rank test was done for both vertices properties. As for the previous analysis, the p-values were too high: 0.58 for the degree test and 0.81 for the betweenness test. Nothing could be said on the similarities of ranks between GRN^r and GRN^{cor}.

Then, we observed the precision as a function of the recall for σ factors (Appendix 3.1) for the three methods described above in Section 4.4.2. The results were not the same depending on the method used to calculate P and R. The values were scattered for the first method but for the second and the third we observed a great decrease of the precision for almost all σ factors. Another remark could be done, the σ factors with largest recall were σ_E, σ_F, σ_G and σ_K (`spoIIIC` and `spoIVCB`). It signifies that the recovering of their edges was not that bad.

26 clusters were found in GRN^{cor} and the distribution of vertices in each cluster is given in Table 3. Like in GRN^r there were clusters with very few genes: 15 clusters over the 26 found had less than 10 genes. By summing the number of genes for those 15 clusters we obtained only 44 genes (1.11% of studied genes). Nevertheless, the modularity associated with this clustering was equal to 0.59 which is considered as good.

Cluster	Number of genes	Cluster	Number of genes
1	823	14	18
2	563	15	2
3	1,107	16	2
4	125	17	2
5	95	18	88
6	232	19	3
7	403	20	2
8	6	21	3
9	36	22	2
10	18	23	2
11	4	24	3
12	2	25	3
13	5	26	3

Table 3: Distribution of the number of genes by cluster in GRN^{cor}.

We also calculated the precision and recall for each clusters in Table 4. From cluster 13 to 26 the precision and recall were equal to zero. Moreover, cluster 12 had excellent values but it contained only 2 genes so it was no relevant for the global inference. The biggest clusters had small values of precision and recall so we deduced that the quality and the quantity of inferred edges were bad.

Cluster	Precision	Recall
1	0.01	0.04
2	0.01	0.49
3	0	0.02
4	0.01	0.06
5	0.03	0.08
6	0.01	0.02
7	0.01	0.09
8	0.01	0
9	0.01	0.07
10	0.04	0.08
11	0.17	0.33
12	1	1
13	0	0
⋮	⋮	⋮

Table 4: Precision and Recall by cluster in GRN^{cor}.

To know if the clusters represent any biological mechanism, the combination of regulators were extracted. We observed only clusters with more than 40 genes to detect such mechanism.

The cluster 2 (Figure 20) seemed to represent a part of the sporulation phase since the majority of genes were regulated by combination of σ_E, σ_F, σ_G and σ_K (`spoIIIC` and `spoIVCB`).

Another cluster that was remarkable was the cluster 4 (Appendix 3.2) in which a lot of genes were regulated by σ_B alone or in combination. This cluster could represent the phases where the bacteria was under stress.

The clustering of GRN^{r} and GRN^{cor} were compared by computing both indices presented in Section 4.4.3. Both values were small so we deduced that there were no similarities between both clusterings.

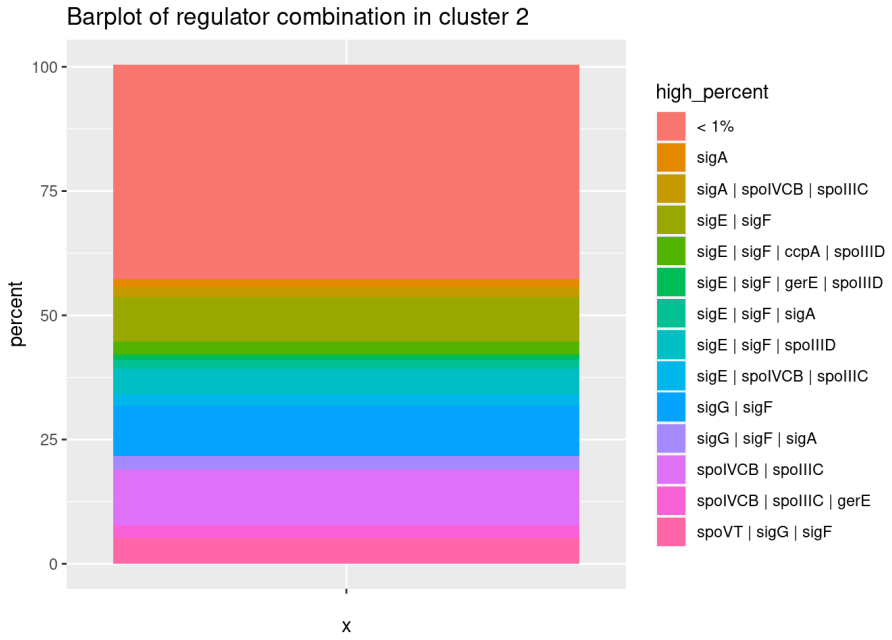


Figure 20: Distribution of the combinations of regulators in cluster 2 of GRN^{cor}.

$$\text{NMI} = 0.049 \text{ and } \text{ARI} = 0.024$$

Finally, in Figure 21 we visualized the distribution of GRN^{r} clusters in GRN^{cor} ones (only the ones with more than ten genes) to see if there was a matching between clusters.

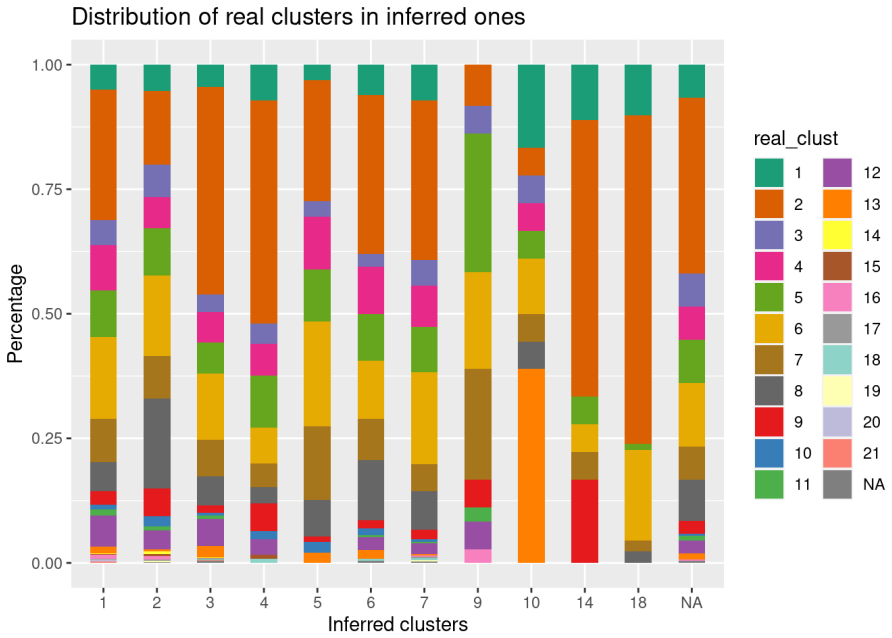


Figure 21: Distribution of GRN^{r} clusters in GRN^{cor} clusters.

As for the previous inferred network, none of the inferred clusters corresponded to only one cluster of GRN^{r}.

5.3.2 Networks inferred by PCIT

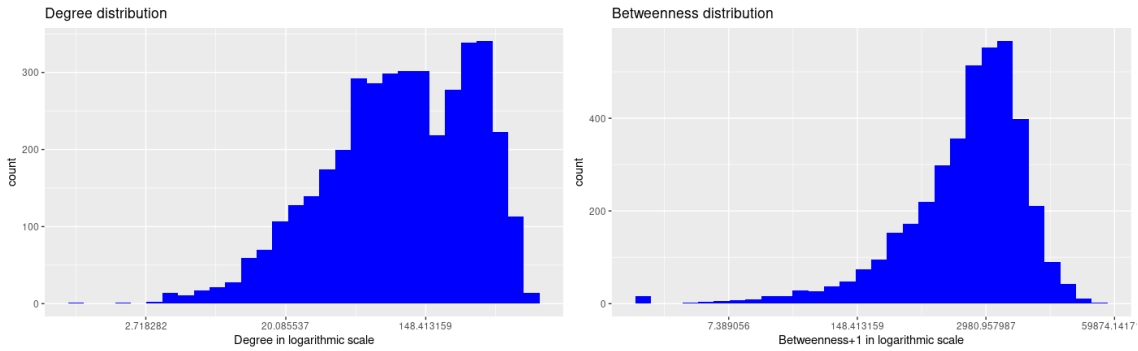
In the present section, we restrict our analysis to the best network inferred with PCIT. It is the one inferred with the option "max" for the setting *tol*. This network will be named GRN^{pcit} in the following.

GRN^{pcit} had 320,214 edges (30 times more than in GRN^{r}) and a density equal to 4.1% (higher than in GRN^{r}). The larger density was a consequence of the high number of edges recovered by the method. Among the inferred edges, only 1,036 were commons to GRN^{r} and it gave a global precision P and a global recall R,

$$P = 0.0032 \qquad \text{and} \qquad R = 0.1035.$$

As for the previous inference method, this one inferred too much edges and the precision tended to zero. However, 10% of the edges in GRN^{r} were recovered. Unlike the others inferred network, GRN^{pcit} is connected and its largest connected component contained 3,977 vertices (all the studied genes).

By comparing the distributions of degree and betweenness with those of GRN^{r} we observed again a big difference. Unlike GRN^{r}, for GRN^{pcit} (Figure 22) there were a lot of genes with large values of degree and betweenness and very few with small values. One more time, the highest values are very different (largest degree for GRN^{pcit}: 674 -largest degree for GRN^{r}: 3,203).



(a) Degree (b) Betweenness

Figure 22: Degree (a) and betweenness (b) distribution of GRN^{pcit}.

A rank test was also done to determine if there were similarities between the degree and betweenness ranks in GRN^{pcit} and GRN^{r}. It appeared that we could not conclude since the p-values were too high. For the degree we had 0.87 and for the betweenness we had 0.80 which was far above the usual level of 0.05.

Precision and recall were also computed for σ factors by the three described methods (Section 4.4.2). The results are presented in Appendix 4.1 and as for GRN^{cor} we had small values of precision for method 2 and 3. The same σ factors appeared to have the good values of precision and recall, the ones implied in the sporulation phase.

7 clusters were found in GRN^{pcit} and the distribution of the vertices in each cluster and the precision/recall by cluster are given in Table 5.

As we had a very large number of inferred edges, the values of precision tended to zero. Nevertheless, for clusters 5 and 6 we had good values of recall so 30% of the recovered edges were effectively in GRN^{r}.

To verify if the clusters represented any known biological mechanism, we looked at the combination of regulators present in each one of them. We remarked the omnipres-

Cluster	Number of genes	Precision	Recall
1	538	0.01	0.04
2	701	0	0.03
3	1,161	0	0.04
4	737	0	0.03
5	560	0.01	0.31
6	113	0.02	0.27
7	167	0	0.02

Table 5: Distribution of the number of genes by cluster in GRN^{pcit} and precision/recall values by cluster.

ence of σ_A in the two first clusters which contains more than 1,200 genes combined (see Appendix 4.2).

In cluster 5 (Figure **??**), we observed 50% of genes regulated by at least one of the next σ factors: σ_E, σ_F, σ_G and σ_K (`spoIIIC` and `spoIVCB`). Those σ factors being involved in the sporulation, we could suppose that a part of the mechanism was recovered.

Finally, we computed the NMI and adjusted Rand index to detect similarities between clustering of GRN^r and GRN^{pcit}. As both values were close to zero, we deduced that there were no similarities between clusterings:

$$\text{NMI} = 0.028 \text{ and } \text{ARI} = 0.005.$$

Finally, we visualized the distribution of GRN^r clusters in GRN^{pcit} clusters in Figure 23. Again, none of the inferred clusters corresponded to only one cluster of GRN^r.

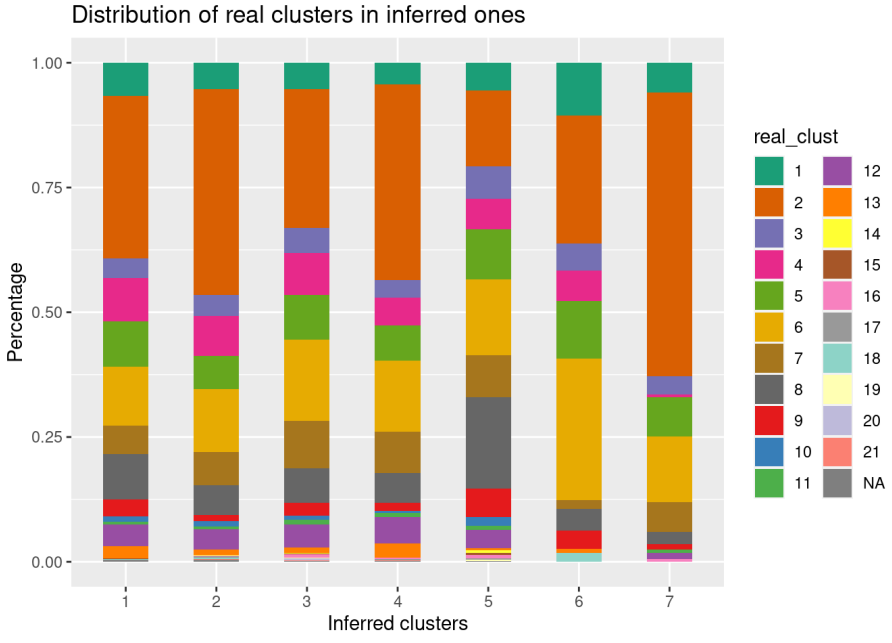


Figure 23: Distribution of GRN^r clusters in GRN^{pcit} clusters.

6 Conclusion

This internship provided networks inferred by various statistical methods. These networks have been compared to GRN^r, there is a summary of the results in Table 6:

	Method	Scaling	Settings	Nb common edges	Nb cluster	Precision (global)	Recall (global)	NMI	ARI
GENIE 3	RF	No	By default	259	40	0.02	0.03	0.06	0.02
	RF	Yes	By default	252	38	0.02	0.02	0.06	0.03
	RF	No	Reg_list	1.517	14	0.13	0.15	0.03	0.01
	RF	No	Change seed	263	43	0.02	0.03	0.07	0.02
	RF	No	σ_list	2.133	6	0.21	0.20	0.02	0.01
	ET	No	By default	256	34	0.02	0.03	0.07	0.07
	ET	Yes	By default	266	33	0.02	0.03	0.06	0.06
PCIT		No	By default	2.656	4	0	0.26	0.02	0.01
		No	Max	1.036	7	0	0.10	0.03	0.01
Pearson		No	Thresh. 0.75	1.208	26	0.01	0.12	0.04	0.02

Table 6: Table summarizing all inferences done during the internship

By looking at the table above, we notice that the inference of regulatory networks is not an easy task. Indeed, with the relevance network and `PCIT` there are too much edges and the methods infer a lot of wrong edges. At this stage, the best result is obtained with `GENIE3` but by passing the σ factor list to help the method. It means that only with the expression data, the methods tested give bad results.

Future work

Other methods can be tested to recover GRN^r as the Bayesian networks presented in Section 4.3 for example. One of the biggest advantage of this method is that the inferred network is oriented. It signifies that, in theory, the real regulation links can be recovered by using this approach.

Moreover, to determine if there exists a matching between real and inferred clusters statistical tests can be done. For example, a chi-square test will be useful to know if the proportion observed on Figure 23 for example can be interpreted or no.

Personal conclusion

From a more personal point of view, this internship allowed me to associate two fields of study that are very interesting to me: mathematics and biology. It also allowed me to discover the world of research and the work in a laboratory. Moreover, I was able to improve my oral communication skills since I did a presentations of my work every week for my tutors. I also did two presentations, at the beginning and at the end of the internship, to present my work to a larger audience.

From a more scientific point of view, I learned a lot about graph theory and how to manage the data associated with it. I also improved my computer skills especially in `R` language by programming with different packages. I learned how to read a scientific paper associated with a package and to understand how it works.

Bibliography

Blondel, V., Guillaume, J.-L., Lambiotte, R., and Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, P10008:1742–5468.

Butte, A. J. and Kohane, I. (1999). Unsupervised knowledge discovery in medical databases using relevance networks. *Proceedings of the AMIA Symposium*, pages 711–715.

Butte, A. J. and Kohane, I. (2000). Mutual information relevance networks: functional genomic clustering using pairwise entropy measurements. *Proceedings of the Pacific Symposium on Biocomputing*, pages 418–429.

Edwards, D. (1995). *Introduction to Graphical Modelling*.

Friedman, J., Hastie, T., and Tibshirani, R. (2008). Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441.

Friedman, N., Pe'er, D., and Nachman, I. (1999). Learning bayesian network structure from massive datasets: the ≪sparse candidate≪ algorithm. *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 206–215.

Huynh-Thu, V. A., Irrthum, A., Wehenkel, L., and Geurts, P. (2010). Inferring regulatory networks from expression data using tree-based methods. *PLoS ONE*.

Koller, D. and Friedman, N. (2009). *Probabilistic graphical models: principles and techniques*. MIT press.

Langfelder, P. and Horvath, S. (2008). Wgcna: an r package for weighted correlation network analysis. *BMC Bioinformatics*, 9(559).

Meinshausen, N. and Bühlman, P. (2006). High-dimensional graphs and variable selection with the lasso. *The Annals of Statistics*, 34(3):1436–1462.

Newman, M. and Girvan, M. (2004). Finding and evaluating community structure in networks. *Physical Review E*, 69:026113.

Nicolas, P., Mäder, U., Dervyn, E., Rochat, T., Leduc, A., Pigeonneau, N., Bidnenko, E., Marchadier, E., Hoebeke, M., Aymerich, S., et al. (2012). Condition-dependent transcriptome reveals high-level regulatory architecture in *Bacillus subtilis*. *Science*, 335(6072):1103–1106.

Novichkov, P. S., Laikova, O. N., Novichkova, E. S., Gelfand, M. S., Arkin, A. P., Dubchak, I., and Rodionov, D. A. (2010). RegPrecise: a database of curated genomic inferences of transcriptional regulatory interactions in prokaryotes. *Nucleic acids research*, 38(suppl_1):D111–D118.

Reverter, A. and K. F. Chan, E. (2008). Combining partial correlation and an information theory approach to the reversed engineering of gene co-expression networks. *Bioinformatics*, 24(21):2491–2497.

Schäfer, J. and Strimmer, K. (2005). An empirical bayes approach to inferring large-scale gene association networks. *Bioinformatics*, 21(6):754–764.

Sierro, N., Makita, Y., de Hoon, M., and Nakai, K. (2008). DBTBS: a database of transcriptional regulation in *Bacillus subtilis* containing upstream intergenic conservation information. *Nucleic acids research*, 36(suppl_1):D93–D96.

Trösser, F., de Givry, S., and Katsirelos, G. (2021). Improved acyclicity reasoning for bayesian network structure learning with nstraint programming. In *Proc. of IJCAI-21*, Montreal, Canada.

Tsamardinos, I., Brown, L. E., and Aliferis, C. E. (2006). The max-min hill-climbing bayesian network structure learning algorithm. *Machine Learning*, 65:31–78.

Verma, T. and Pearl, J. (1991). Equivalence and synthesis of causal models. *Uncertainty in Artificial intelligence*, 6:255–268.

Appendixes

1 Appendix A: Analysis of GRN^r

1.1 Boxplot of degree and betweenness according to the nature of genes

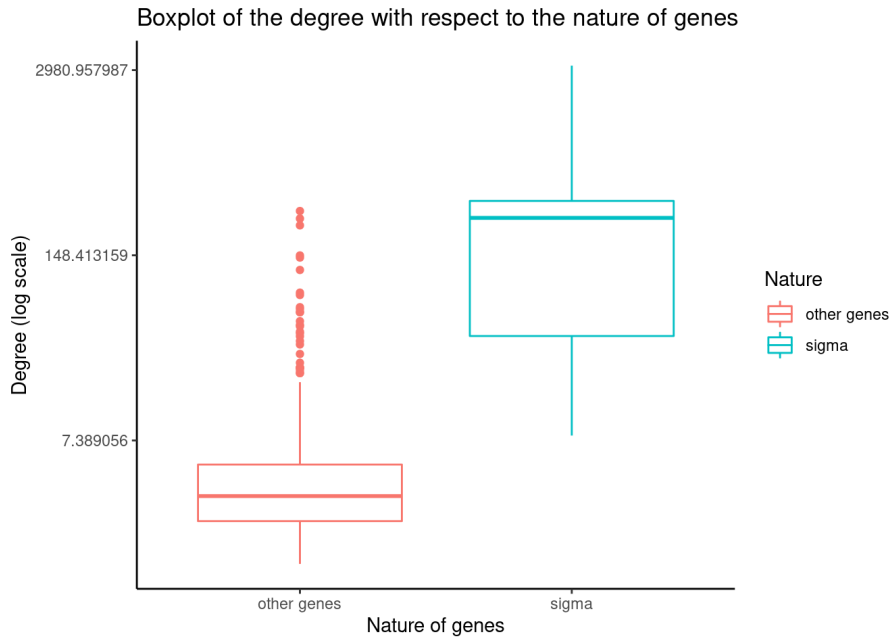


Figure 24: Boxplot of the degree in GRN^r.

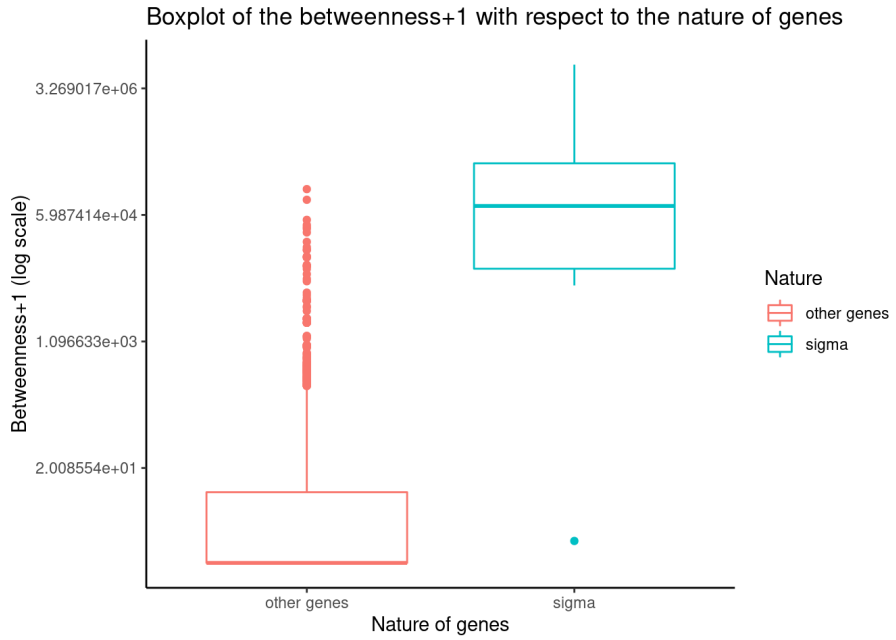


Figure 25: Boxplot of the betweenness in GRN^r.

2 Appendix B: Results from GENIE3 inference

2.1 Precision and Recall by σ factors

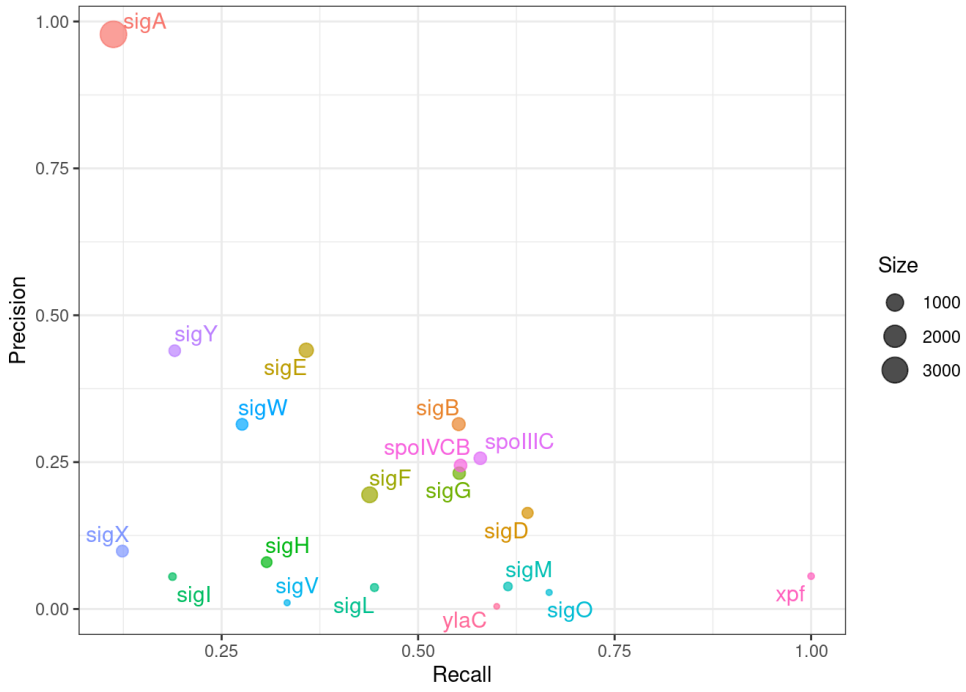


Figure 26: Precision as a function of recall for σ factors in GRN^{RF} (method 1).

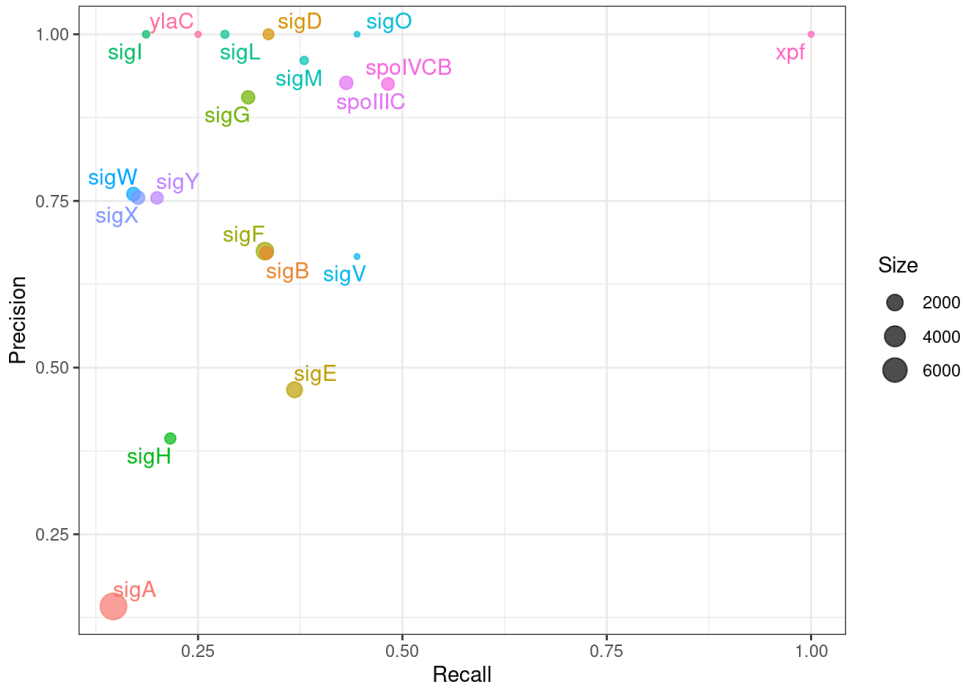


Figure 27: Precision as a function of recall for σ factors in GRN^{RF} (method 2).

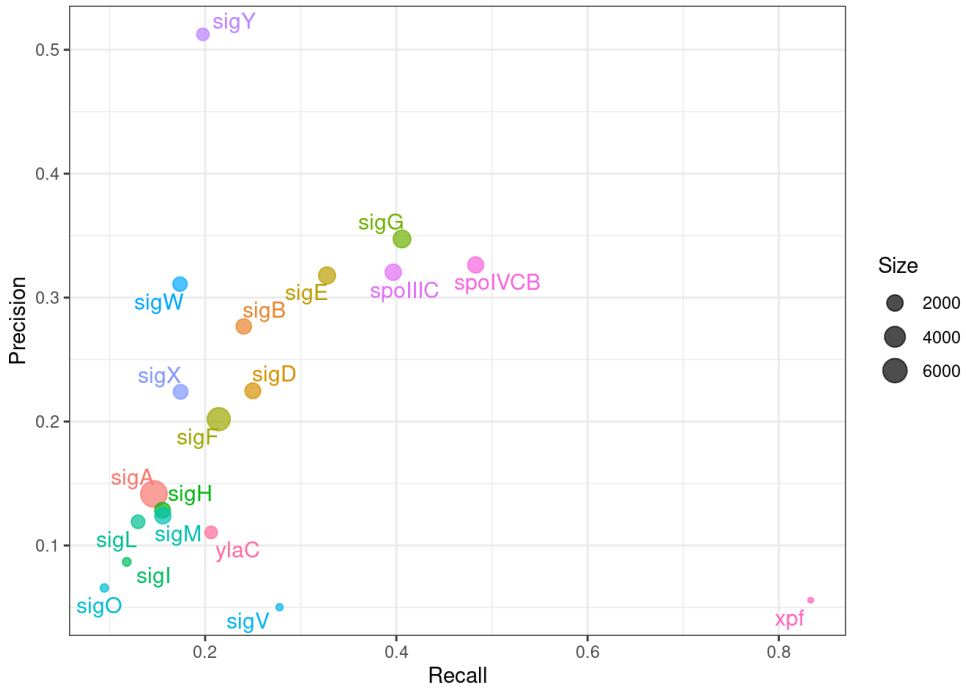


Figure 28: Precision as a function of recall for σ factors in GRN^{RF} (method 3).

2.2 Combination of regulators in cluster 4 of GRN^{RF}

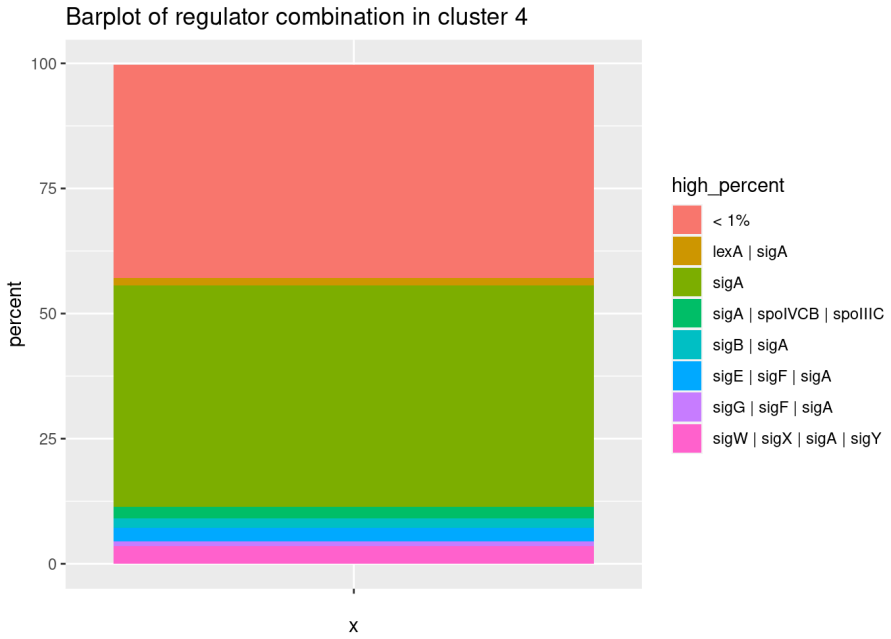


Figure 29: Distribution of the combinations of regulators in cluster 4 of GRN^{RF}.

2.3 Other results obtained with GENIE3

Method	Scaling	Settings	Nb_common_edges	Nb_cluster	Precision	Recall
RF	No	By default	259	40	0.02	0.03
RF	Yes	By default	252	38	0.02	0.02
RF	No	Reg_list	1.517	14	0.13	0.15
RF	No	Change seed	263	43	0.02	0.03
ET	No	By default	256	34	0.02	0.03
ET	Yes	By default	266	33	0.02	0.03

Table 7: Results of the different inferences done with GENIE3.

3 Appendix C: Results from "naive" inference

3.1 Precision and Recall by σ factors

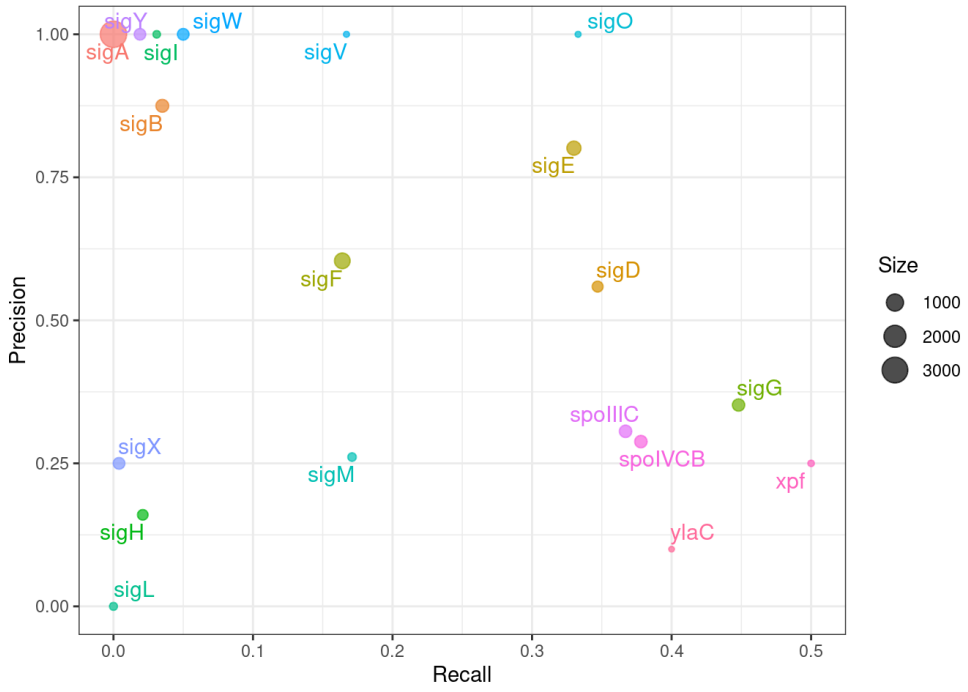


Figure 30: Precision as a function of recall for σ factors in GRN$^{\text{cor}}$ (method 1).

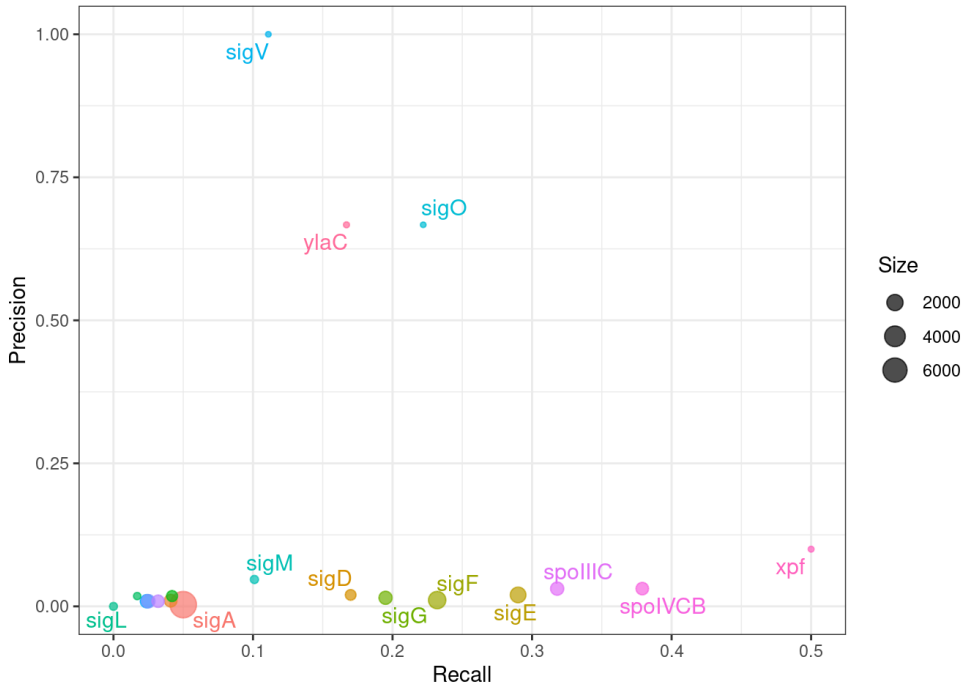


Figure 31: Precision as a function of recall for σ factors in GRN$^{\text{cor}}$ (method 2).

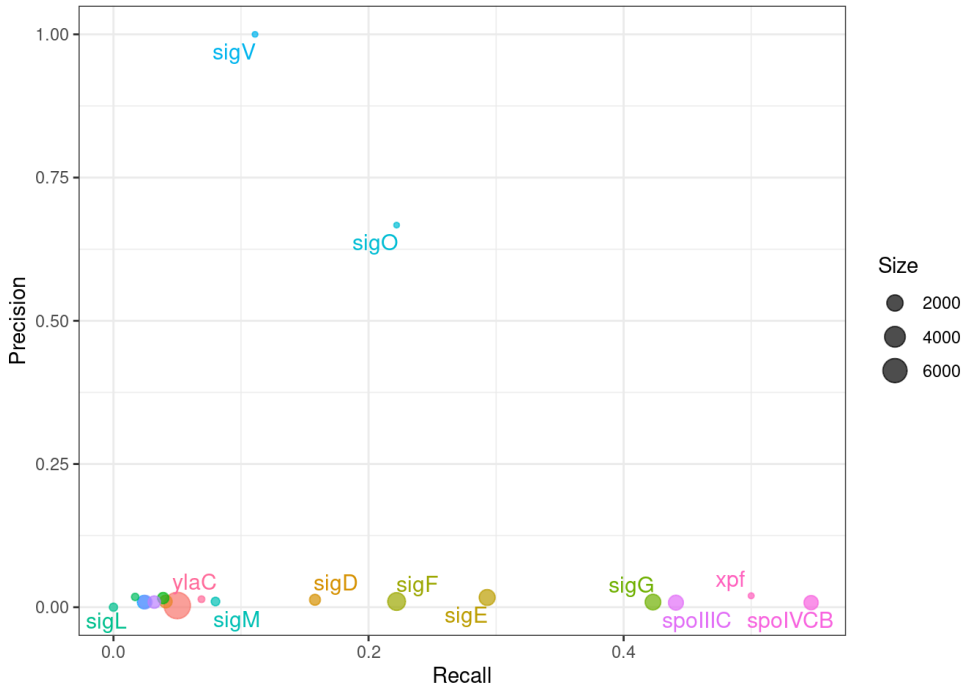


Figure 32: Precision as a function of recall for σ factors in GRN$^{\text{cor}}$ (method 3).

3.2 Combination of regulators in cluster 4 of GRN$^{\text{cor}}$

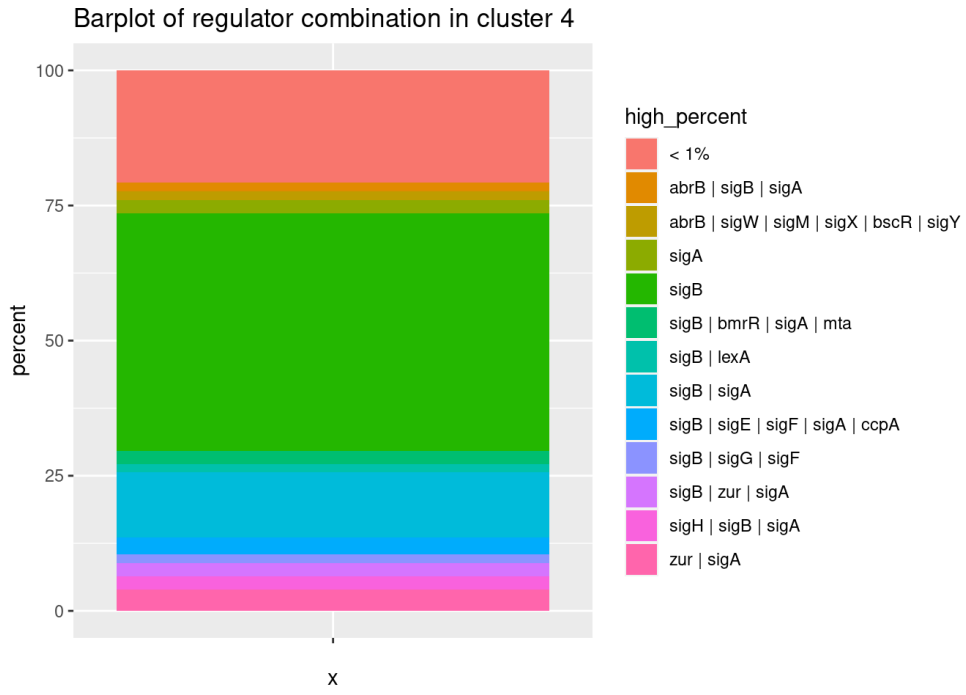


Figure 33: Distribution of the combinations of regulators in cluster 4 of GRN$^{\text{cor}}$.

4 Appendix D: Results from PCIT inference

4.1 Precision and Recall by σ factors

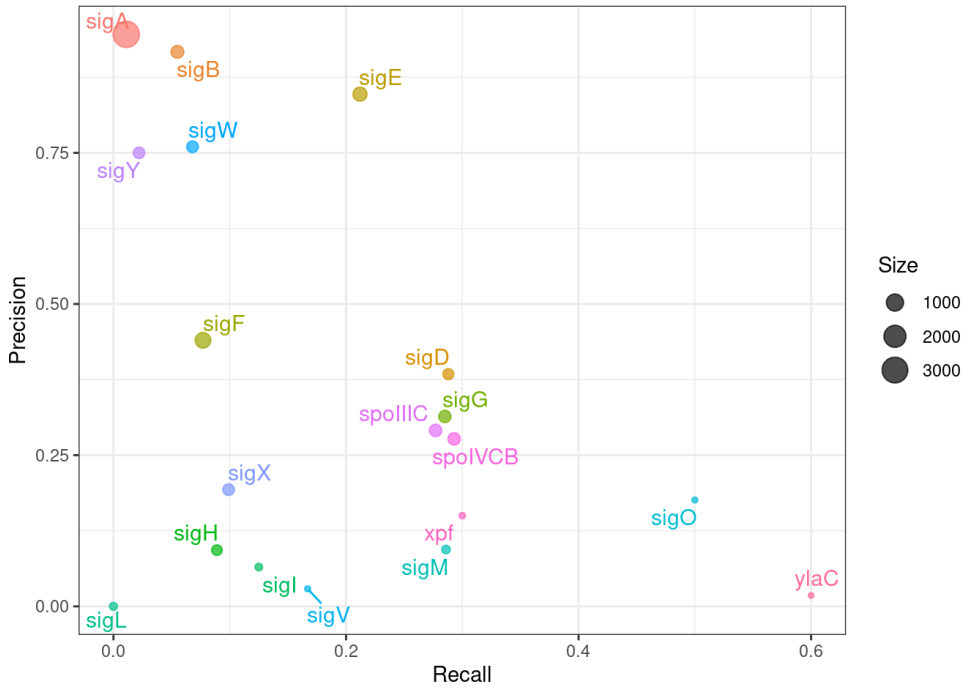


Figure 34: Precision as a function of recall for σ factors in GRN$^{\text{pcit}}$ (method 1).

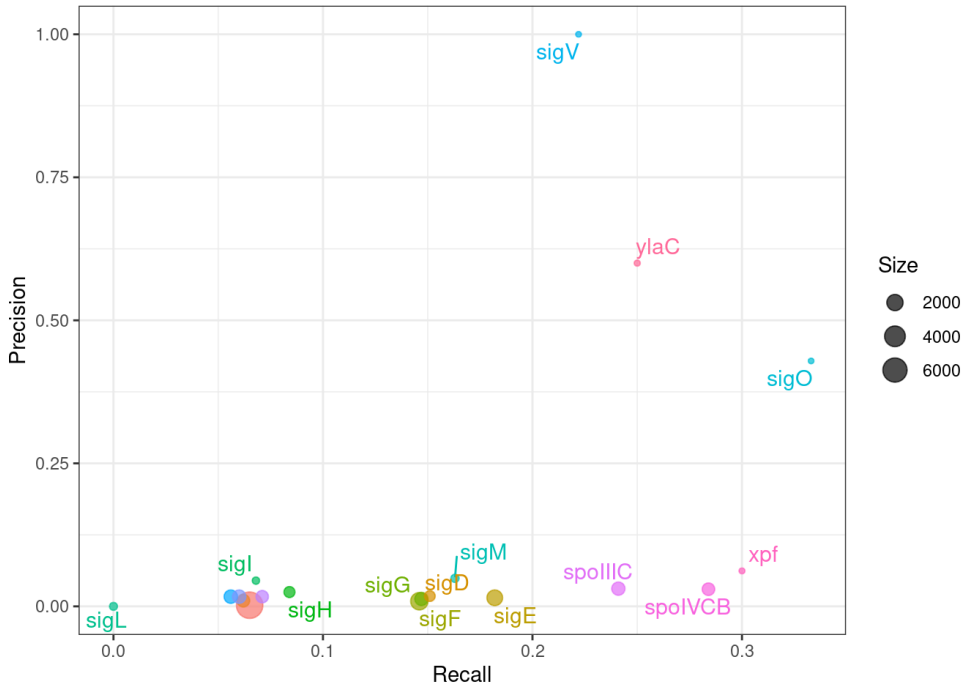


Figure 35: Precision as a function of recall for σ factors in GRN$^{\text{pcit}}$ (method 2).

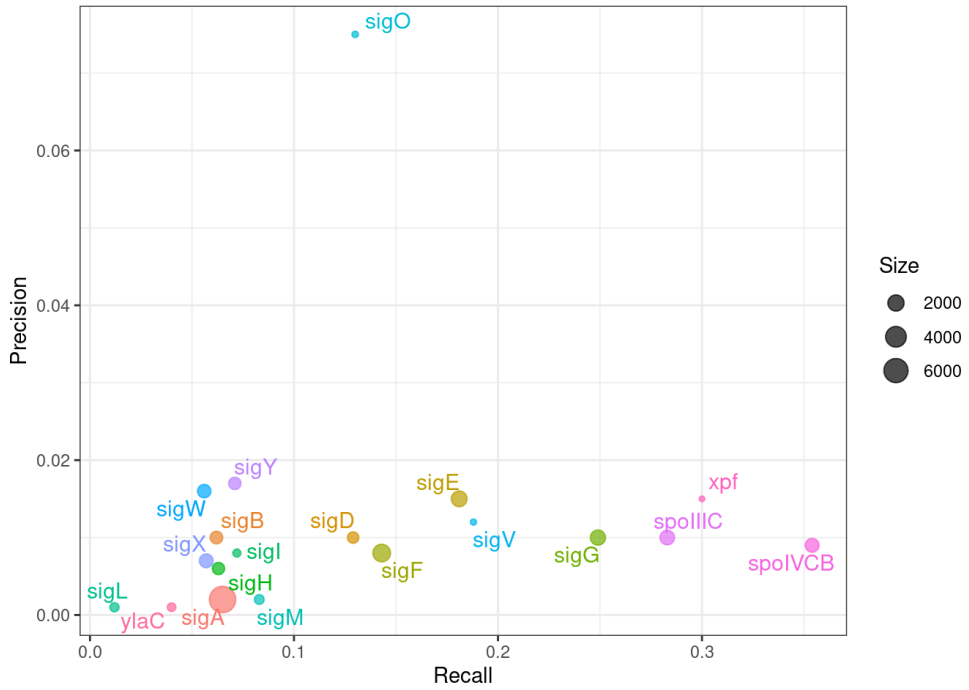


Figure 36: Precision as a function of recall for σ factors in GRN$^{\text{pcit}}$ (method 3).

4.2 Combination of regulators in first clusters

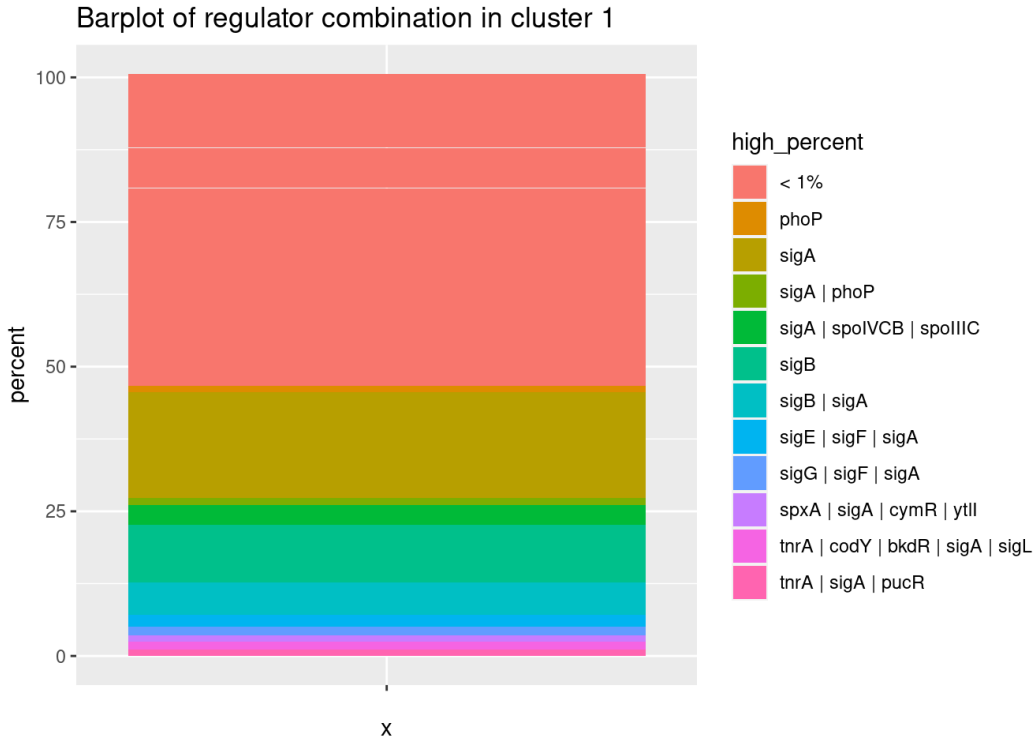


Figure 37: Combination of regulators in cluster 1 of GRN$^{\text{pcit}}$.

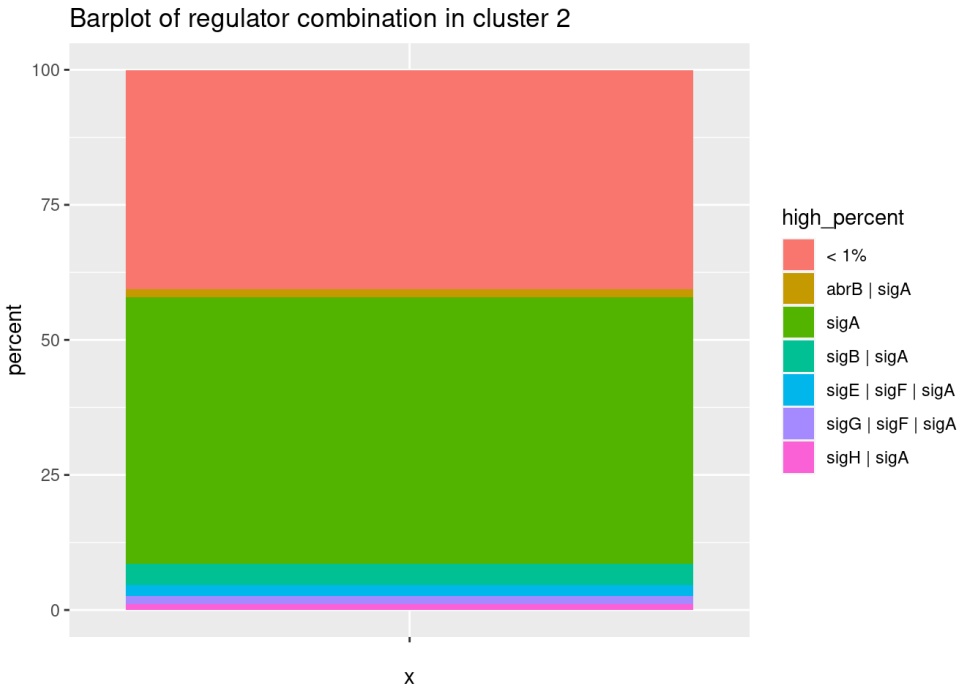


Figure 38: Combination of regulators in cluster 2 of GRN^{pcit}.