Master thesis : Gene regulatory network inference from observational and interventional expression data

Auteur : Smetz, Colin
Promoteur(s) : Geurts, Pierre
Faculté : Faculté des Sciences appliquées
Diplôme : Master en ingénieur civil en informatique, à finalité spécialisée en "intelligent systems"
Année académique : 2016-2017
URI/URL : http://hdl.handle.net/2268.2/2590

University of Liège
Faculty of applied sciences

Gene regulatory network inference from observational and interventional expression data

Graduation Studies conducted for obtaining the Master's degree in Computer science and Engineering by Colin Smetz

Advisor : PhD Pierre Geurts

Academic year 2016 - 2017

Abstract

Gene regulatory network inference from observational and interventional expression data

Author : Colin Smetz Master's degree in Computer science and Engineering
Academic year : 2016 - 2017 **Promoter** : PhD Pierre Geurts

The problem of reverse-engineering biological networks has attracted a lot of attention in the last decades. Studying the interactions occurring inside a living organism is of great importance to understand the behavior of biological systems. The development of computer science and the abundance of new genetic data raised the question of predicting gene regulatory networks. These networks describe how some genes regulate the expression of some other genes.

Many methods have already been developed to infer these networks from gene expression data. Among them, GENIE3, a method based on Random Forests, was proposed and achieved state-of-the-art performance. However, one drawback of GENIE3 is its inability to use the specificities of some types of gene expression measurements, potentially missing useful information. In particular, datasets often include knockouts, which are measurements done after the deletion of a gene.

This thesis proposes new variants for GENIE3, based on the idea of *enriched random forests*, in order to integrate knockout specific information as weights guiding GENIE3 to a better prediction. First, the methods are tested on ideal cases where a knockout of every gene is available. Better predictions are indeed achieved and several ways of achieving the best results are highlighted. Realistic cases are then tested. Less convincing results are then obtained, although interesting phenomena are discovered.

The second part of the thesis studies the possibility of predicting the effect of knockouts. Differences and similarities with the GRN prediction problem are analyzed and a method of evaluation, although imperfect, is proposed. Several methods are then evaluated, showing relatively encouraging results. Some initiated reflections call for future developments.

The possibility of using the proposed weighted GENIE3 methods in other situations is also briefly explained. Important improvements are indeed achieved on several datasets without the use of knockouts.

Acknowledgements

I would like to thank my advisor Pierre Geurts for his guidance during the choice of my master thesis' subject, his continuous help and suggestions during the progression of this work, and his very helpful comments for the writing.

I also want to thank Vân Anh Huynh-Thu for her time and help throughout the year and her very helpful proofreading too.

I also thank my father and my brother for their reading and comments.

Finally, I am grateful to the other researchers and master students doing machine learning whose presentations and discussions were a great source of motivation.

Contents

Chapter 1

Introduction

The problem of reverse-engineering biological networks has attracted a lot of attention in the last decades. Indeed, studying the interactions occuring inside a living organism is of great importance when it comes to understanding the behavior of biological systems and their consequences. The development of computer science and the disponibility of new data has made possible the development of new methods capable of predicting these networks from the data. In particular, the simplification of gene sequencing techniques and gene expression measurements resulted in abundant genetic data, which raised the question of finding genetic networks.

Genes are used by an organism to produce specific proteins. The more a gene is used, the more it is *expressed*. Some genes have the particularity of being regulators of other genes. These regulators will produce transcription factors, which are proteins capable of inhibiting or increasing the expression of other specific genes. All these interactions between genes form a **gene regulatory network**.

In the recent years, many methods have been developed in order to infer gene regulatory networks (GRN) from gene expression data. Some of them use statistics or information theory. Another group of them uses machine learning techniques. This is the case of GENIE3, a method using Random Forests, who is currently one of the top performing methods.

However, one of the drawbacks of this method is that it considers every kind of gene expression measurement in the same way. Indeed, there are different types of data. Observational data is obtained by measuring gene expressions under natural conditions. Interventional data is obtained by doing the measure after an artificial intervention. In particular, *knockouts* consist of the deletion of some gene, whose expression is forced to zero. Knockout data is much more informative but currently GENIE3 is not fully using this information as it ignores its particularities.

The purpose of this master thesis is thus to develop new methods in order to integrate knockout data in a more efficient way in GENIE3, with the objective of obtaining better predictions of the network. This will be the first part of this thesis, where we will propose several approaches. These approaches are all based on the idea of a weighted version of GENIE3, where weights are values derived from knockouts so as to influence GENIE3 in a positive way. Results will be given on ideal and realistic network predictions. We will see that important improvements can be made, although the problem becomes quite hard when we deal with realistic networks.

The second part of the thesis will focus on a similar but quite different problem which is the computational prediction of the effect of a knockout. In this part, we will not care about the entire network, but will instead focus on finding genes which are impacted, directly or not, by the deletion of another gene. As it will be explained, this problem is more difficult than the other one and poses interesting challenges.

In Chapter 2, we will start by defining in details the initial problem of gene regulatory network inference and the specific questions that this master thesis will address. In Chapter 3, we will define the different methods and evaluation techniques that will be used throughout the work. The next two chapters will focus on the problem of GRN inference by the integration of knockout specific information in GENIE3, first in an ideal case (Chapter 4) and then in a more realistic case (Chapter 5). We will then address the second problem of knockouts prediction in Chapter 6. Finally, Chapter 7 will show the interest of our new methods in other situations where no knockouts are available, before we conclude.

Chapter 2

Problem definition

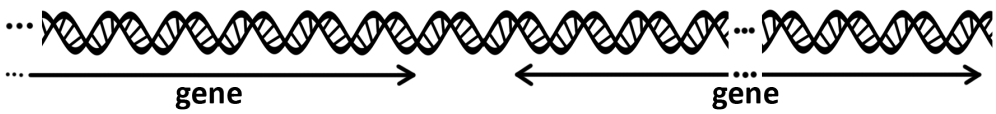
In this chapter, we define step by step the concepts that are necessary to understand the subject of this master thesis. We start by explaining what gene regulatory networks are and the problem of their inference. Then, we will look at the available types of data used by inference techniques. We therefore present an overview of such techniques. Finally, we precisely define the objectives pursued by this master thesis in the next chapters.

2.1 Gene regulatory networks

2.1.1 Genes

Since they are central in the problem, it is important to clearly define what genes are, their roles and their specificities.

A **gene** corresponds to a region in the DNA of a living organism. It is the molecular unit of heredity. That means that it is responsible for passing phenotypic (observable) traits to the offspring. The role of genes is mostly to code for protein sequences, i.e. to tell other molecules which proteins to produce. These proteins play a central role in the functioning of living organisms.



DNA is thus composed of many different genes whose number and functions vary between species. For example, yeast contains about 6,000 genes while the human genome has more than 20,000 genes.

2.1.2 Gene expression and regulation

Now, when a gene is actually used to synthesize some gene product (usually proteins), we say that it is *expressed*. The expression of some gene is not constant and can vary a lot under different circumstances.

Several techniques exists for measuring gene expressions of an organism at a given moment. The most common one is the microarray technology, which has the

advantage of being quite simple and cheap. RNA-seq technologies are gaining more and more success, but despite their increased precision, they stay relatively difficult to use. All these methods have the ability to provide a numerical value for each gene representing the expression intensity at the time of measurement.

Variations of gene expressions depend on environmental factors, but also on the expression of other genes. Indeed, some genes will act as regulators of other genes, i.e. they will influence the expression of other genes. A gene i might either be an activator of some gene j, or a repressor. Usually, only a small subset of all genes are regulators. These genes will also be called **transcription factors**, although this term refers to the proteins produced by these genes.

This leads us to the notion of *gene regulatory networks* (GRN) which will be at the core of our problem. Genes can be seen as nodes of a network, with regulatory links between them, representing the whole internal regulation structure of gene expressions.

2.1.3 Network inference problem

An important problem which gave rise to quite a lot of research in recent years is the problem of gene regulatory networks inference. The idea is to develop methods which would be able to retrieve the correct regulatory network, for one species, using essentially gene expression data. We will give an intuitive definition for now and we will formalize the problem at the end of this chapter.

Basically, the data at hand consists of many measurements, each of them containing an expression value for each gene. From these samples, we want to find the true causal links in the network. This network might be seen as a directed or undirected graph, whether we are interest in the direction of regulatory links or not. The graph can also be signed or unsigned whether we want to know the excitatory/inhibitory nature of links or not. We insist on the fact that we want direct links only. If A regulates B, which in turn regulates C, we do not want to find a link $A \rightarrow C$ although A regulates C indirectly through B.

One difficulty of the problem resides in the very small number of samples compared to the high number of features (the genes).

Usually, methods do not directly provide a network but instead provide a score associated to each possible gene-to-gene interaction. When a score is high, it means that the method is more certain about it being a true interaction. By sorting these scores, we can thus obtain a ranking of all interactions, from the most certain to the least certain one. The problem of finding a correct threshold for separating real interactions from the others is yet another problem that will not be developed in this thesis.

In addition to the pure engineering and mathematical interest behind this problem, the importance in the field of biology is also obvious. Gene regulatory networks being central in all organisms, solving this problem would help understand how these organisms work, how they evolve or how they would react to environmental or genetic perturbations.

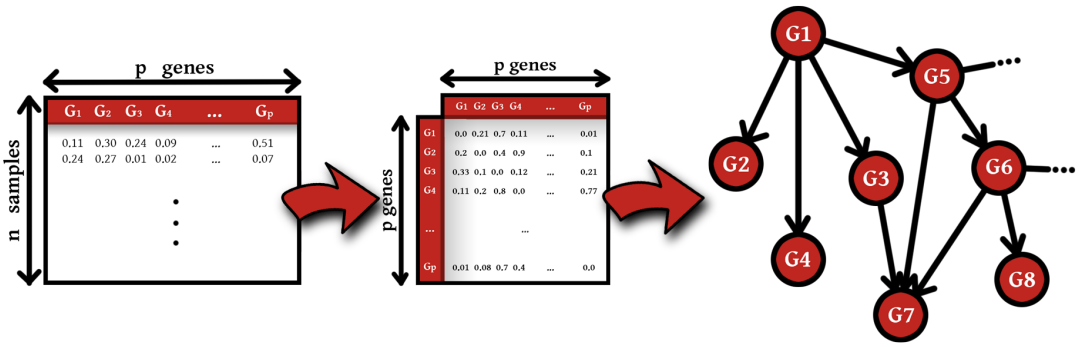


Figure 2.1: Schematic representation of the different steps in GRN inference

2.2 Types of data

In this section, we will take a look at the different types of data that we might encounter when working on gene network inference. We think it might be important to give enough details about this since this master thesis will be focused on one particular type of sample.

2.2.1 Wild type

What is commonly referred as a wild type sample is a measure of gene expressions under natural conditions, usually at steady-state level, thus representing the usual gene expression values. It is the most basic type of data found in datasets, but it is still important.

2.2.2 Time series

Time series samples are measures of gene expressions taken at regular time intervals, typically after the application of a perturbation on the system. These samples thus show the evolution of gene expressions as a function of time. As gene regulation is basically a process that is not instantaneous, more information can usually be extracted from such data.

2.2.3 Knockdown and knockout

Knockdown and knockout samples are a type of *interventional* data (by opposition to *observational* data). Interventional data refers to samples obtained after performing a manipulation on the organism.

A knockout corresponds to the deletion of some gene, i.e. forcing the expression of this gene to be zero. A knockdown is the same principle, except that the expression of the gene is only artificially reduced to some value, higher than zero. The measure is taken after stabilization of the organism following the intervention. In some cases, several genes are knocked out at once. In all cases, the knocked out genes are known.

It is important to realize that this type of data brings more *causal* information than a simple observational sample. As the sample is actually the result of some

operation, we can have more confidence on the direction of regulatory links. The extreme amplitude of the modification also highlights with more evidence the regulation processes. Taking advantage of that will be at the center of our objectives.

2.3 Overview of inference methods

In this section, we will give a brief overview of existing techniques for gene network inference. Some of them will be described in more details in the next chapter, but we believe it is interesting to have a global picture of the research before going further. The following list is not exhaustive at all. [12] provides a much more complete and detailed review.

2.3.1 Unsupervised methods

In unsupervised methods, we have no a priori knowledge of any of the regulatory links. This is the most common type of method in gene network inference as finding even a few true or false interactions experimentally can be quite costly.

Correlation-based methods Methods based on correlation assume that two genes having correlated expression levels have a higher chance of sharing a regulatory link. The most common measure is the Pearson coefficient, but more adapted rank-based method are frequently used, such as Kendall's τ coefficient or the *Mutual rank* method. Several variants taking advantage of GRN properties exist. Another possible measure of correlation between expression vectors is the mutual information. *Relevance networks* [3] use the mutual information I between gene expression vectors to deduce an interaction score. Again, several variants have been developed. *Context likelihood of relatedness* [4] extend this method by analyzing the distribution of MI values. *ARACNE* [13] applies the Data Processing Inequality in order to filter out indirect interactions. *MRNET* [16] combines MI and a feature selection algorithm (minimum-redundancy-maximum-relevance).

Model-based methods Model-based methods are based on physical models imitating the biological systems, typically using ordinary or stochastic differential equations, and reconstructing GRNs by learning the parameters of these models [5] [2]. This is probably the most complex type of methods.

Probabilistic models These methods make use of well known techniques in Probability and Statistics. *Bayesian Networks* model the network as a DAG (directed acyclic graph) and use bayesian network learning techniques to maximize $P(G|X)$, the probability of the graph given input data X. Some of these methods also handle dynamic interactions [24]. *Graphical gaussian models* [11] look at the conditional dependencies among genes by computing the covariance matrix from available data and by inferring an undirected graph from it. The *Graphical LASSO algorithm* [15] uses a similar idea but in the same time uses a penalized linear regression approach to control the sparsity of the network.

(Linear) regression methods This category includes methods using regression models, namely linear ones. One example of them is the TIGRESS method [7], which formulates the GRN inference problem as a sparse linear regression problem, and learns the regulators of each target gene by combining feature selection with LARS with stability selection.

GENIE3 This method [10] uses a gene by gene approach, trying to infer regulators of genes by using Random Forests and Extra-Trees for regression and feature selection.

Z-SCORE The Z-SCORE method [21] focuses specifically on knockout data. The score of some interaction $i \to j$ is chosen to be the normalized absolute deviation of gene j after a knockout of gene i. If no knockout exist for some gene i, we simply take the sample where it is minimal.

Other Many other methods exist like *SIGMOID* which uses linear combination with soft thresholding, *Mass-distance* [23] which is based on a probabilistic similarity measure between expression vectors, or *EUCLID* which simply takes the expression vectors of two genes and uses their euclidean distance as an interaction score.

2.3.2 Supervised methods

Supervised methods take advantage of some interactions that have been experimentally verified. This can be either a positive (there is a regulatory link between genes) or a negative (there is certainly no link) information, but the first one is more frequent.

Supervised methods train a machine learning classifier based on the known interactions. As the vast majority of interactions are not regulatory links, we can take some of them randomly as negative samples. Samples are usually built from the gene expression vectors of both genes of an interaction. All interactions are then predicted from this classifier.

In practice, this approach can be applied to lots of machine learning techniques, like support vector machines (for example the SIRENE method [17]) or Random Forests [22]. These methods often outperform unsupervised methods, but they depend on the existence of partial a priori knowledge of the network.

Remark : semi-supervised techniques, combining both ideas also exist, but are not yet very common. We will not talk about them.

2.4 Objectives

We now have all the keys to explain the objectives that will be pursued by this master thesis. The focus in the next chapters will be on knockout data and how we can take advantage of them. Two objectives will be pursued. Before we explain them, it is necessary to formalize the situation.

2.4.1 Notations

Our goal is to make predictions about gene regulations. Genes form a regulatory network G. We will use the notation G_{ij} to denote whether interaction $i \rightarrow j$ between genes i and j is a real interaction ($G_{ij} = 1$) or not ($G_{ij} = 0$).

We have as input data gene expression measurements, denoted by the matrix X of size $N \times p$ (N samples and p genes). The value of X_k^j corresponds to the expression value of gene j in sample number k.

This data can be divided into two parts : X_{obs} of size $N_{obs} \times p$ corresponds to the observational data (wild-type, time series, etc.) while X_{KO} of size $N_{KO} \times p$ corresponds to the knockout data. We have $N = N_{obs} + N_{KO}$.

We also introduce another notation to identify the knockout of one gene i in particular. This notation will be valid only if a gene has been knocked out only once and if no simultaneous knockout exist. We will thus use the notation $X_{KO(i)}^j$ to denote the value of gene j after a knockout of gene i. This must not be mistaken with $X_{KO,i}^j$ which represents the value of gene j in the ith sample of knockout data (which is not necessary the knockout of gene i).

2.4.2 First objective

A first objective will be the improvement of gene network inference by integrating knockouts specific information X_{KO} to other methods. We will focus on GENIE3. Indeed, GENIE3 is currently one of the state-of-the-art methods but it does not allow to fully exploit the information contained in knockout data (it treats this data as any other observational data).

Our goal will thus be to improve GENIE3 in order to exploit knockout data in a more optimal way. In practice, this will be done by combining GENIE3 with the Z-SCORES method who has the opposite problem of fully exploiting knockouts while having poor performance on observational data.

This problem will be treated in details in chapters 4 and 5

2.4.3 Second objective

A second objective will be the prediction of the effect of new knockouts. As only a few knockout samples are usually available, we might be interested in knowing what the effect of a new knockout would be, i.e. which genes would be affected. This might be interesting given the fact that knockouts are more difficult to obtain. Also, if a knockout intervention is not done with the purpose of getting data for GRN inference, but for another reason, it could be useful to have an idea of the consequences of this intervention beforehand.

We will thus assume that we have some observation data X_{obs} and a restricted number N_{KO} of knockouts X_{KO} with $N_{KO} < p$. There is then a number $p - N_{KO}$ of unknown knockouts. For some gene i which has not been knocked out, what we would like to predict is the value of $X_{KO(i)}^j$ for $j \in 1, ..., p$, or, at least, which of these will get exceptional values (i.e. which genes are affected). This will be formalized in more details in chapter 6.

This problem is somewhat different from the previous one. The main difference is that in regulatory network inference, only direct links are sought, while any kind

of sufficiently important effect is desired here.

Chapter 3

Methods

Now that the problem is well defined, we will explain in more details some of the methods that will be used throughout the rest of this work. We will start by two inference methods mentioned before (Z-scores and GENIE3) that were the starting points of our research. We will then develop new methods that will be used later to achieve our objectives, although we will stay at a quite abstract level for now. Finally, we will talk about some common evaluation techniques used to assess our performances.

3.1 Z-scores

Z-scores is a very easy method specifically designed for knockout data. It is based on the idea that if the expression of a gene j deviates greatly from its usual value after a knockout of gene i, then there is probably a regulatory link between i and j.

Formally, to find the Z-score z_{ij}, we look at the expression $X^j_{KO(i)}$ of gene j in the sample corresponding to a knockout of gene i. Then, we compute the absolute deviation from some "mean value" μ_j, scaled according to the standard deviation σ_j :

$$z_{ij} = \left| \frac{X^j_{KO(i)} - \mu_j}{\sigma_j} \right|$$

The values μ_j and σ_j are simply the mean and standard deviation computed from all expression values of gene j in the dataset. But we can imagine a few variants. Instead of using the whole dataset, these values (μ and σ) can be computed using only wild-type samples or only knock-out samples. Although the differences of performance are usually not really significant, we will sometimes use some of these variants later. From one single knockout sample, we can thus compute p Z-scores (where p is the number of genes), one for each interaction from the knocked out gene to every gene.

No knockout case

As mentioned before, knockouts are not always available, but there is a simple way to adapt the method in that case. If there is no knockout sample for a gene i, just

take the sample where its expression value is the lowest (since it would be zero in a knockout) and compute Z-scores from it.

Of course, this method is much less powerful in such a case. Real knockouts provide a direction information : since the knocked out gene has been artificially put to zero, we know that other perturbations are a consequence of that. Here, we have no such information. Moreover, deviations will be often smaller and thus less significant.

Comments

On knockout data, as it will be shown later, this method yields quite good results. However, it has some inevitable drawbacks that we would like to highlight here. First, Z-scores are sensitive to indirect effects. Indeed, if $A \rightarrow B$ and $B \rightarrow C$ and if their effect are both important (and have the same sign), then the effect from A to C is expected to be high too. Some variants propose to try to remove them at the end, with more or less success.

Another drawback is that Z-scores do not take into account the direction of the deviation (increase or decrease from the mean), assuming that the distribution is symmetric, since we take the absolute value.

Z-scores are also dependent, as mentioned before, on the data from which the mean and the standard deviation are computed.

Finally, this method is not directly adaptable to a simultaneous knockout of several genes.

3.2 GENIE3

GENIE3 [10] is a Random Forest-based method that was a top-performer in the 2009's DREAM4 challenge and in the 2010's DREAM5 challenge. Its principle could be summarized like this :

1. Use a gene by gene approach : find regulators of one gene, and repeat for each other gene independently.

2. To find regulators of one gene, predict its expression values from the expression of all other genes, using Random Forests. Then, use the feature importances returned by the Random Forest as scores for the interactions.

3. Gather all scores together in the end and rank them.

We will now detail these steps, starting with a brief explanation of the Random Forests machine learning technique.

3.2.1 Regression trees

A regression tree is a supervised machine learning technique. Its objective is to learn from a training set a model predicting the (continuous) value of some output variables from the value of other variables, by subdividing the samples recursively, thus forming a tree.

Figure 3.1 illustrates how it works. At each node of the tree, a subset of all samples is available. The method will then try to separate these samples into two groups in order to reduce at most the "impurity" of the output variable inside each subgroup. The two subgroups form the two children of the node, from which the same procedure will be applied.

This separation is usually done using a greedy heuristic. The split is based on the value of one of the input variables, let us denote it x. Some value a is chosen, then samples are divided between the ones where $x < a$ and the ones where $x \geq a$. The choice of the variable x and the value a is done such that the impurity reduction of the output variable is maximal.

Many types of impurity criteria exist. They are different whether we are dealing with a classification or a regression problem. In our case of regression trees, the most common criterion is simply the variance. We can thus define the impurity reduction like this :

Variance (impurity measure). *Given a variable y with sample values $y_i, i = 1, ..., N$, we define its variance $V\{y\}$ as :*

$$V\{y\} = \frac{1}{N} \sum_{i=1}^{N} (y_i - c)^2$$

, where

$$c = \frac{1}{N} \sum_{i=1}^{N} y_i$$

Impurity reduction. *Given a series of sample values $y_i, i = 1, ..., N$ at some node s, which are partitioned in two distinct subgroups $y'_j, j = 1, ..., N_l$ and $y''_k, k = 1, ..., N_r$ at child nodes, with $N_l + N_r = N$, we define the impurity reduction as :*

$$(\Delta I)_s = NV\{y\} - N_l V\{y'\} - N_r V\{y''\}$$

The tree can be developed until each leaf contains only one sample. However, to avoid overfitting the data and obtain a better generalization accuracy, the tree might be stopped earlier.

Once the tree has been built, predicting the output of a new sample can be done by following the separation criterion at each node, stopping when it reaches a leaf node. The output value is then the mean value of the output in all samples of the leaf node.

Regression trees have several interesting advantages compared to other machine learning techniques. Since the tree structure can be retrieved directly, the result is easily interpretable. Doing a prediction has a complexity of O(log N) for N samples which is quite fast. But the characteristic that interests us the most, which is linked to the interpretability, is the possibility to assess the importance of each variable for predicting the output.

Indeed, regression trees can provide a numerical value for each input feature which is representative of its importance. A low value would mean that the feature could be removed without deteriorating much the prediction accuracy, while a high value would mean the opposite. A simple way to compute this is by looking at the impurity reductions.

Feature importance. *Let S be the set of all nodes which split the data based on the value of some feature f. The importance of that feature f is then defined as*

$$\sum_{s \in S} (\Delta I)_s$$

Basically, the importance of the feature is equal to the total impurity reduction induced by this feature. This can then be used to do feature selection, i.e. selecting the most useful features when there are too many of them.

Although regression trees have many advantages, they suffer very often from overfitting and unstability (a small variation in the data yields a totally different tree), causing bad generalization accuracy. These problems can be solved by using ensemble techniques, which leads us to the concept of Random Forests.

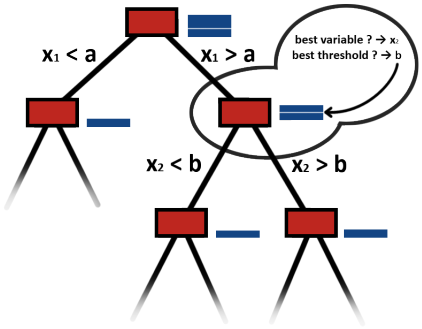


Figure 3.1: Development of a regression tree

3.2.2 Random Forests

Random Forests are one example of an ensemble technique. Ensemble techniques propose a solution to overfitting problems by building an ensemble of many imperfect models which achieve better results when combined together.

Figure 3.2 illustrates the method. In the case of Random Forests, many slightly different regression trees are built. To avoid training the same tree over and over, some randomness must be introduced in each tree. This is first done by building each tree on a bootstrap sample of the data. For a dataset of N samples, a bootstrap is done by drawing N samples randomly, with replacement. Because of the replacement, not all samples are taken, on average only $\approx 67\%$ of them, the rest being duplicates.

A second way to add randomness is by imposing constraints at the nodes. Instead of finding the best variable to split on between all p variables, a subset of K variables is selected randomly. The split is then done on the best of these K variables. A common practice is to take $K = \sqrt{p}$.

Once all trees have been built, a prediction can be done by averaging the prediction of all trees. The variable importances can be computed in the same way by averaging or summing the importances given by each tree. Doing this, we reduce the high accuracy variance of the single regression tree method.

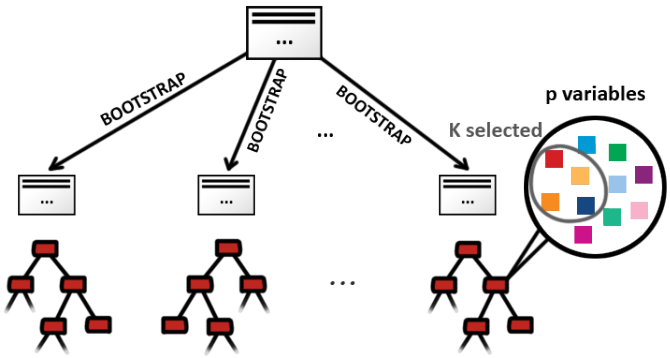


Figure 3.2: Abstract illustration of Random Forests

An alternative to Random Forests is the Extra-Trees method [6]. This technique builds trees based on all samples, and selects the best split at each node from K possible splits selected randomly (by choosing a variable and a threshold randomly).

Given an ensemble of T trees, built on N samples, the algorithmic complexity of these methods is on the order of $O(TKNlogN)$.

3.2.3 GENIE3

We finally arrive at the method that interests us. GENIE3 [10], as mentioned before, is a GRN inference method that uses ensembles of trees. In particular, it makes use of the feature importance computation.

If we denote by $\mathbf{X}_k^{-j}$ the k-th sample from which we have removed the expression value of the j-th variable

$$\mathbf{X}_k^{-j} = (X_k^1, ..., X_k^{j-1}, X_k^{j+1}, ..., X_k^p)^T$$

then we can define the method as follows :

1. For j = 1 to p :
   - Build a learning sample containing all expression values except the ones of gene j as inputs, and the normalized expression of gene j as output :
   $$LS^j = \{(\mathbf{X}_k^{-j}; X_k^j), k = 1, ..., N\}$$
   - Use the Random Forest or Extra-Trees method and make it learn from this data.
   - Use the variable importances returned by the ensemble as interaction scores, representing whether variables are good potential regulators of gene j or not. The importance value of variable i corresponds to the score of interaction $i \rightarrow j$.
2. Aggregate all p rankings obtained before into one single global ranking.

Remark : the normalization of the output when building the learning sample is crucial to obtain correct aggregation at the end. Indeed, the sum of all feature importances is often close to the total variance in the output vector. So, even if the ranking for one Random Forest will be correct, the values will not be comparable at the aggregation phase, as the variables with a higher variance will get higher scores. Note that despite this correction, it is still possible to have a perfect ranking at a gene level, but an imperfect global ranking after the aggregation.

Since GENIE3 simply consists of p random forests, with p the number of genes, its complexity is $O(pTKNlogN)$. Being able to handle non-linear relations and to decompose the dataset into subgroups where the relations are more apparent, GENIE3 can provide better results than many other methods. It is however also sensitive to indirect effects or confounding factors.

3.3 Weighted GENIE3

As such, GENIE3 uses knockout data in the same way as any other kind of data. What we would like to have is a way to integrate some a priori information obtained from knockout (typically with Z-scores) so as to guide GENIE3 in a better direction. The hope is that, Z-scores information being complementary to that of GENIE3, their combination will yield better results that each of them alone.

The following methods have been inspired by the idea of *Enriched Random Forests* [1], first developed by Amaratunga et al. in 2008. The idea was already used as a way to improve GENIE3 by Petralia et al. [19] in 2016. Our methods are built from the same idea, but we will try new variations.

3.3.1 Weights

For now, we will ignore the specificities linked to knockouts and just assume that we have some a priori information about gene interactions, whatever the origin of this information. Basically, we consider that we have a numerical value for each interaction, indicating how much we think the regulatory links exists. This corresponds to the output of most GRN inference techniques.

These numerical scores are first converted to other numerical scores, according to several possible methods that will be explained in later chapters. These new numerical values will be called *weights*. The weight associated to interaction $i \to j$ will be written w_{ij}.

3.3.2 Local approach (GENIE3-LW)

In the local approach of the method (Figure 3.3), we choose to influence GENIE3 at a node level. When building trees in a Random Forest, we select randomly K variables at each node from which the best one is chosen. This random selection is uniform : each variable (gene) has the same probability to be picked than any other.

We can then use our a priori information to give more chance to the more important variables to be picked. Concretely, the Random Forest responsible for finding regulators of gene j (scores $s_{.j}$) is given weights $w_{.j}$. Then, at each node of each tree, K variables are selected using weighted sampling, so that the higher weighted variables are more frequently picked.

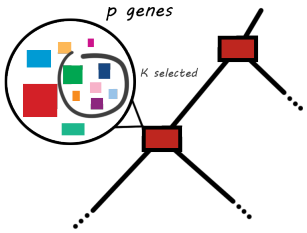


Figure 3.3: Drawing at each node. Each variable has a different weight.

If a variable has a weight which is two times higher than another variable, then it is twice as likely to be chosen. More precisely, we assign to each variable the probability $P_i = \frac{w_{ij}}{\sum_{i \neq j} w_{ij}}$ and select a variable based on these probabilities. Once a variable has been chosen, it is not replaced and the selection probabilities of the remaining variables are adapted.

Figure 3.4: Another way to visualize the drawing : each variable occupies some space on an interval, this space being proportional to its weight. We then select a random number in this interval and look in which variable space it fell

3.3.3 Global approach (GENIE3-GW)

The global approach (Figure 3.5) uses the same idea of a weighted drawing but in a different way and at a different moment. Here, we select a subset of all genes *before* building a tree. The tree will then be developed using only these genes, and without doing any subselection at the nodes.

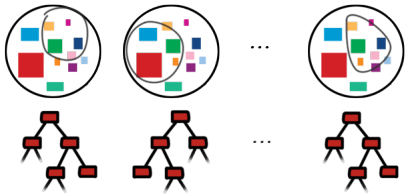


Figure 3.5: Selection before each tree. Each variable has a different weight.

For this method, we also decided to try another way to select genes randomly, although the previous selection technique could have been applied as well (we will compare them later in the case of this global approach). The weights in this approach will be chosen such that each value is in the interval $[0, 1]$ and can be seen as a probability. We then select each variable or not based on this probably. If the weight is one, then the

variable is always chosen. If it is zero, then it is never used. Practically, the probabilities are chosen so that the expected number of drawn genes at each tree is K. Mathematically, this is done by enforcing $\sum_{i \neq j} w_{ij} = K$.

This new selection method has several different characteristics compared to the local drawing. First, the exact number of genes at each tree is variable and will not necessarily be equal to K, although the average number will be close to K. Another difference is that values are bounded (maximum of one), which forbids too extreme values. Finally, we have a way to enforce the use of some gene, by setting its probability to one, while an infinite value would have been necessary for the "local" selection. We chose to test this global approach for all these differences although there was no initial motivation for this idea in particular.

Weights creation

We must discuss some problems about how we create these weights and make them follow the previous constraints. We will actually start from the weights of the local approach, $w_{.j}$, to derive the corresponding weights $w'_{.j}$.

We first impose the constraint $\sum_{i \neq j} w'_{ij} = K$ simply by multiplying each weight by K over the total sum :

$$w'_{ij} = \frac{w_{ij}}{\sum_{i \neq j} w_{ij}} \times K$$

In some cases, this might be enough, but very often, the difference between different weights is so large that the highest ones will have a weight higher than one. If it appears, we set these weights to 1, but now the previous constraint does not hold anymore. We define the **residual quantity**, i.e. the quantity that was cut off by setting weights to one :

$$K_{res} = K - \sum_{i \neq j} w'_{ij}$$

To restore the constraint and have a correct sum, this quantity must be distributed among other weights (< 1). Weights will be increased uniformly. So, for t weights for which we increase the value, the individual increase will be equal to K_{res}/t. This number t is not necessarily equal to the number of weights smaller than one as some of them can be close to 1 and would be again too high after the increase. However, t will be chosen as the highest possible, to perturb values as little as we can.

We might think of other methods but this one seemed to be correct enough. Other techniques were tested but usually did not change results significantly. This will be described later and this method will be considered as the default one.

3.3.4 Transcription factors

A last comment is necessary. When the transcription factors of the network are known, we do not need weights for the interactions $i \rightarrow j$ where i is not a transcription factor. Indeed, GENIE3 can use this information to ignore the genes which are not transcription factors when building trees. For weights creation, we must adapt methods by simply ignoring the interactions that will not be taken into account.

3.4 Evaluation techniques

Evaluating how well we did at predicting the network is a more complex problem than it seems. As we can impose many threshold values to go from the score matrix to the network, we would like to be able to evaluate a method independently of this threshold. In this section, we explain several classic methods that will be used throughout the next chapters. More specific techniques will be developed inside their corresponding chapter when necessary.

3.4.1 Confusion matrix

We first assume that we have predicted a single network, by setting a threshold on the score matrix. We would like to evaluate the correctness of this network compared to the real network. Each interaction can either be predicted true or not, and it can be actually true or not. Thus each interaction can fall into one of these four categories : true positive (TP), false positive (FP), true negative (TN) or false negative (FN). By counting how many interactions fall into each category, we obtain the **confusion matrix** (see example on Figure 3.6).

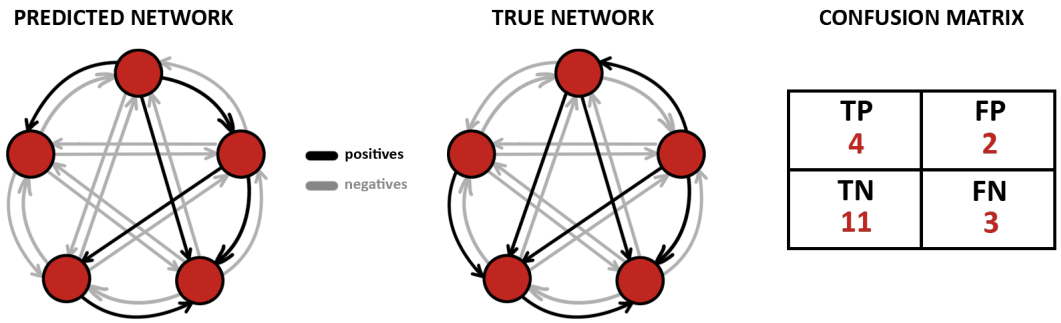


Figure 3.6: Example of a confusion matrix for a predicted network of five nodes.

From these value, we can derive other quantities, namely :

- Precision $= \frac{TP}{TP+FP}$, the proportion of elements predicted as positives that are actual positives.
- Recall or sensitivity $= \frac{TP}{TP+FN}$, the proportion of actual positive elements that have been found.
- Specificity $= \frac{TN}{TN+FP}$, the proportion of negatives that are correctly identified.

3.4.2 Precision-recall curve

Now that we have several ways to assess the prediction for a single threshold, we would like to assess a method independently of this threshold. A first way to do that is by computing the **precision-recall curve**. Basically, the idea is to compute the (precision ; recall) pair for each possible threshold and plot these points on a graph. From there we can analyze the curve's shape. A perfect network should have a point located in (1;1). The perfect curve should look like a square (see Figure 3.7). The closer we are to the perfect curve, the higher the area under the curve (with a maximum of 1). We can thus use a single measure, the Area Under the Precision-Recall curve (AUPR) to assess the performance of a method.

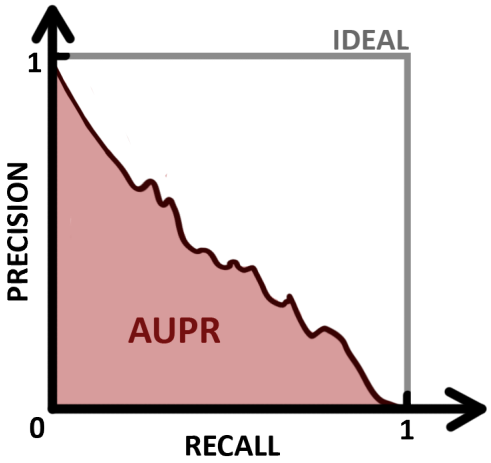


Figure 3.7: Illustration of classic and ideal PR curves

A random guess, on the contrary, will look like a very unstable curve converging progressively towards a precision equal to $P/(P+N)$ as the recall approaches 1. A network containing a lot of interactions (a lot of positives) will thus tend to have a higher AUPR even with random predictions. This should be taken into account as a single AUPR value in itself will not be fully interpretable without information about the network. However, the comparison of AUPR results between different tested methods is not a problem.

3.4.3 ROC curve

Another curve that can be used is the Receiver Operating Characteristic or ROC curve (see Figure 3.8). It displays the sensitivity as a function of $1 - specificity$, also called the false positive rate. The best point is the (0;1) point and the perfect curve a square with area 1. However, in this case, the expected curve for a random method is a 45° line with area 0.5. The AUROC (for Area Under ROC curve) will then be comprised between 0.5 and 1 for all methods. It can be interpreted as the probability that a randomly chosen positive is ranked before a randomly chosen negative.

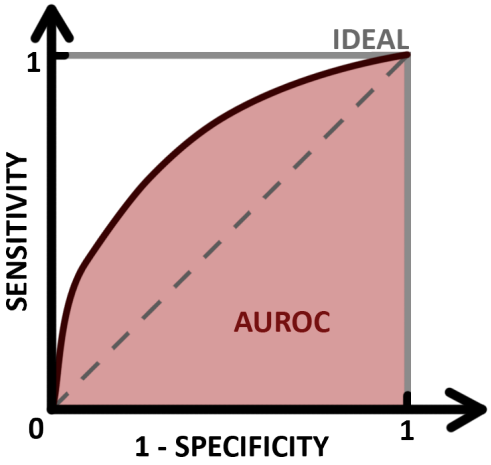


Figure 3.8: Illustration of classic and ideal ROC curves

3.4.4 DREAM challenges

In order to evaluate the quality of methods, several challenges, namely the DREAM challenges (for Dialogue for Reverse Engineering Assessments and Methods) took place over the years. They provide gene expression data of various types, as well as a gold standard network (the true network to find) for evaluation. These datasets will be used later to evaluate our own methods.

Chapter 4

Knockout integration I

In this chapter, we will start using the previously described methods so as to achieve our objectives. We will thus integrate knockout information with the weighted GENIE3 techniques. In this first part, we will consider only an ideal case where a knockout is available for every gene. We will first see the different techniques tested to generate weights from knockouts, then present and analyze results on an artificial dataset. Finally, we will compare these results with a few interesting variants and discuss before moving to more realistic datasets in the next chapter.

4.1 Tested approaches

Let's start by describing the different knockout integration techniques that will be tested and analyzed in the next sections. As explained before, we would like to be able to combine two methods : Z-scores, that provide information specifically for knockouts, and GENIE3, that is more general. We will assume the use of Random Forests (instead of Extra-Trees) every time GENIE3 or one of its variants is used. For Z-scores, means and standard deviations were computed from knockout samples only.

4.1.1 Simple product

Before using the new weighted GENIE3 techniques, we would like to compare results to a much easier approach : multiplying GENIE3's output with Z-scores. Basically, the new score p_{ij} for an interaction $i \to j$ is obtained by taking the product of the score s_{ij} of GENIE3 and the Z-score z_{ij} :

$$p_{ij} = s_{ij} \times z_{ij}$$

We will later refer to this technique as GENIE3 $\times$ Z.

4.1.2 Weighted GENIE3

More complex methods will use the previously described weighted GENIE3 methods. The weights w_{ij} necessary for both local and global approaches will be derived from Z-scores z_{ij} according to different ideas that we will describe now.

Z-scores `(Z)` The simplest method is to use directly Z-scores as weights : $w_{ij} = z_{ij}$. However, we would like a way to control the range of values so that high values are not too high compared to the lowest ones, or the contrary. We basically want a way to control the ratio between the highest values and the lowest ones, so that the difference of power can vary. A simple way to do that is by using an exponent β : $w_{ij} = z_{ij}^{\beta}$. For low values of β, interactions with higher values are not given much more importance than interactions with low values (in the limit case $\beta = 0$, we obtain the basic GENIE3 method when using the local approach). On the contrary, when β is large, the low-weight interactions are almost completely ignored.

p-values `(P-VALUES)` The idea here is to create weights from p-values instead of Z-scores, although these p-values can be computed from Z-scores. Intuitively, we can define the p-value as the probability to obtain a given value or a higher one at random, according to some probability distribution. Here, we will assume that the expression of a gene follows a normal distribution, whose mean and standard deviation are given by the available data (the same way as for Z-scores). We will thus use a similar idea as for Z-scores and look at the p-value associated to the expression of gene j after a knockout of gene i in order to compute w_{ij}. Since p-values decrease when Z-scores increase, we must take the inverse of it to have usable weights :

$$w_{ij} = \frac{1}{pvalue} - 1 = \frac{1}{2 \times (1 - \mathcal{F}_X(z_{ij}))} - 1$$

where $\mathcal{F}_X$ represents the cumulative distribution function of the normal distribution $\mathcal{N}(0, 1)$. We subtract one to force every value in the interval $[0, +\infty[$. This technique is the one used by Petralia et al. [19] in their article. We will again apply an exponent β to control the amplitude of variations.

Binary weights `(BINARY)` Previous ideas use a large range of values as weights. It might be interesting to study the effect of having only two possible values : a high value for interactions considered important, and a low one for the others. Practically, we will choose a given ratio $\frac{high}{low}$, the low value being equal to 1. The important interactions will be chosen according to p-values. We can either take the n interactions with the lowest p-value, or take all interactions with a p-value $\leq \alpha$, which is more generalizable since it does not depend on the number of possible interactions.

Binary weights, per gene `(PERGENE)` When using binary weights, it might appear that for some genes, the weights $w_{\cdot j}$ of interactions going to that genes are all low, which would be equivalent to using GENIE3 with no weight. It is normal, as some genes have indeed no regulators, but it might be possible that because of a large number of interactions having a high Z-score without being regulatory links (indirect links, etc.), some true interactions are not taken into account. It is also possible that too many interactions are considered as important, leading to the $w_{\cdot j}$ being almost all high. This would mean that some gene is regulated by all the others, which is unlikely. To counter these effects, we will use binary weights but decide the high and low ones gene by gene. The weights in $w_{\cdot j}$ getting a high value

will be the ones corresponding to the n highest Z-scores in $z_{.j}$. The value n is the same for all genes.

Uniform lowest (`UNILOW`) This last method is the most complex. Basically, it is like a combination of the Z-SCORES and the BINARY method. We will select a certain percentage of interactions, starting from the highest Z-scores, and make the other ones have the same weight (the name *uniform lowest* was chosen because the lowest-Z-score interactions have a uniform distribution for their weights). We thus assume that we cannot really make assumptions on the importance of these interactions. The high Z-score interactions get a weight proportional to their Z-score. These interactions are selected based on a threshold α on the p-values, just as in the BINARY case. Finally, we would like to be able to decide the global importance of the low weights compared to the other ones. To do this, we choose a value $k \in [0, 1]$ and scale each group of weights such that

$$\frac{\sum_{i \in I^*} w_i}{\sum_{i \in I} w_i} = k$$

, where I is the set of all interactions and I^* the set of top interactions.

Figure 4.1 shows an abstract illustration of these weighting methods to help visualize them intuitively. Note that the representation is done for the local method. It is important to realize that the conversion done for the global method can result to a different behavior from what was initially expected. For example, it is not possible to impose a ratio as large as we want for the BINARY method. Indeed, the high weights will inevitably become too large and be redistributed among the low ones, leading to binary weights too but with another ratio. Because of that, the ratios will be different for each gene. The same problem appears when β is chosen too high in the two first techniques.

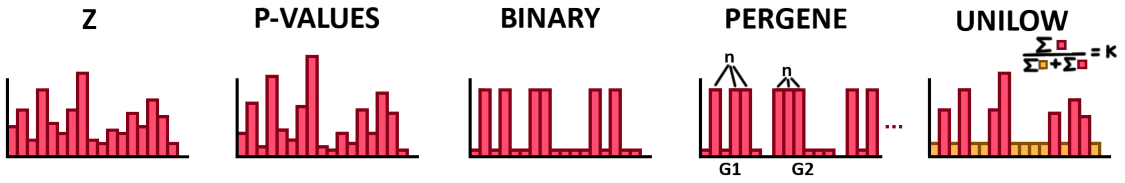


Figure 4.1: Abstract representation of weight creation methods

4.2 DREAM4 results

We now analyze the results of these different techniques on a particular dataset. We will use the data of the DREAM4 challenge, which is composed of five artificial networks of 100 genes each. For each of these five networks, we have time-series and knockout data, and only one wild-type sample. Knockdowns were also provided but we will ignore them. Time-series samples are divided into ten groups of 21 samples measured at successive moments in time. Knockouts are composed of 100 samples, one knockout for each gene. A gold standard network was provided for each network. No transcription factors were given, meaning that any gene can be a regulator.

4.2.1 Influence of the parameters

Before comparing all the methods with each other, let us first analyze each method separately in order to have an idea of the influence of the different parameters on the results. To keep things clear, we will show, for each value of the parameters tested, the average AUPR of all five networks (using 300 trees per Random Forest). AUROCs tend to follow the same trends but with much less variation, which makes them less interesting.

Of course, in a real case, we could not fine-tune parameters like this since we would have no way to assess the performance. However, some parameters such as α should be quite independent of the network. For other parameters, results will at least give us a better intuition about the best value to choose.

Influence of the power β

The power β is used in the Z and in the P-VALUES methods to influence the difference between high and low values. We tested values from 0.2 to 5.

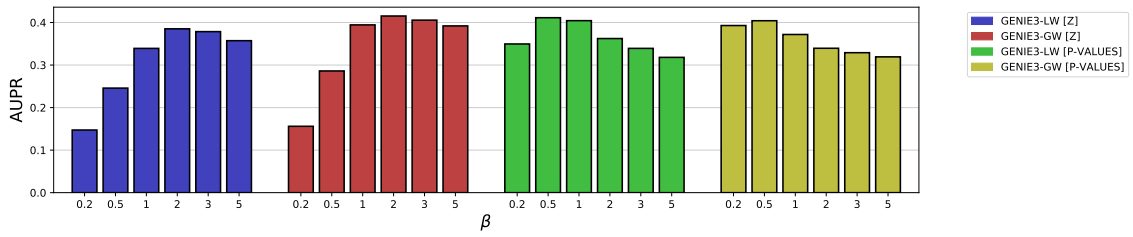


Figure 4.2: AUPR results (average over 5 nets) showing the influence of β

From Figure 4.2, it seems that there is a peak corresponding to the optimal value of β. When using Z-scores, it is around 2, while for p-values, it is around 0.5. It is not surprising since p-values tend to be very small (and the corresponding weights large) for exceptional values, given the shape of a normal distribution. Therefore we expect that it is necessary to reduce this exaggerated difference between values.

Influence of the ratio $\div$

The ratio $\div$ represents the ratio between high and low values in the BINARY and PERGENE methods. We chose values from 2 to 1000. From Figure 4.3, we immediately observe some kind of asymptotic behavior. We already knew it would be the case for the global approach : when the ratio exceeds some value and if the number of high weights is less than K, the high weights are cut off (because they exceed 1 after normalization) and the low values are increased, thus only one situation is possible after this limit value. However, the same behavior appears for the local approach.

Note that in both approaches, a very high ratio means that high weighted interactions will be selected almost everytime, but it does not imply that other interactions will never be chosen or that splits will always be on these interactions. For later tests, the best solution is thus probably to take $\div = 100$ or more.

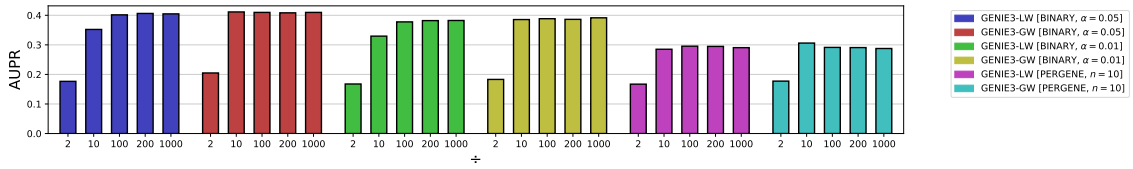


Figure 4.3: AUPR results (average over 5 nets) showing the influence of ÷

Influence of the threshold α

The threshold α on the p-values is used in the BINARY and UNILOW methods to select the top interactions. We chose values between 0.01 (very few interactions) and 0.5 (lots of interactions). From Figure 4.4, we observe again a peak value. Choosing $\alpha = 0.05$ seems the best solution in most cases.

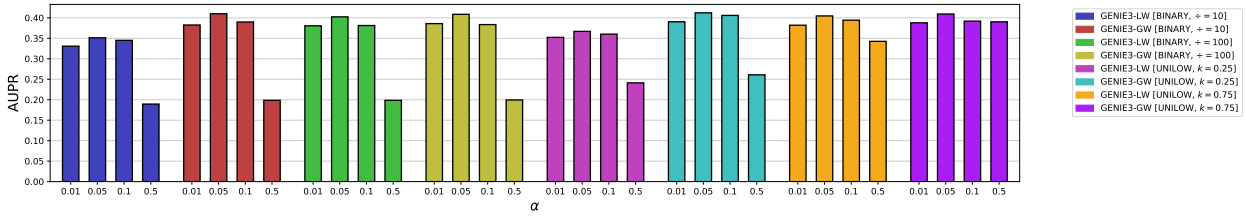


Figure 4.4: AUPR results (average over 5 nets) showing the influence of α

Influence of the proportion k

The proportion k is used only in the UNILOW method to determine the importance of top interactions compared to the low Z-score interactions. From Figure 4.5, we observe that it is better to choose a value higher than 0.5.

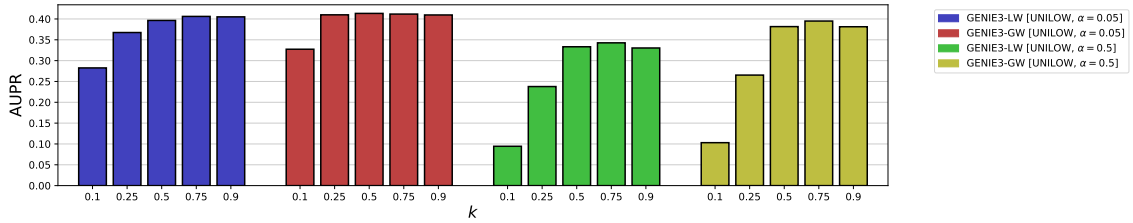


Figure 4.5: AUPR results (average over 5 nets) showing the influence of k

4.2.2 Complete results

We will now compare all methods between each other. We selected the following set of techniques and parameters :

- GENIE3
- Z-SCORES
- GENIE3 × Z

- Weighted GENIE3 (local + global) :
  - Z : $\beta \in \{1, 3\}$
  - P-VALUES : $\beta \in \{0.5, 1\}$
  - BINARY : $\alpha \in \{0.01, 0.05, 0.1\}$, $\div \in \{100\}$
  - PERGENE : $n \in \{5, 10\}$, $\div \in \{100\}$
  - UNILOW : $\alpha \in \{0.05, 0.5\}$, $k \in \{0.5, 0.75\}$

Nothing prevents us from using the product with Z-scores in the case of weighted GENIE3. For this reason, for every weighted GENIE3 output, we will compute the performance of the product as well, thus integrating the Z-scores at two different steps.

We will currently use knockouts only to compute Z-scores. GENIE3 will be trained on the time-series data.

Figures 4.6 and 4.7 show AUPR diagrams for networks 1 and 5. Diagrams for networks 2 to 4, as well as the exact numerical values are available in Appendix B.1.1 and B.1.2 for clarity. For bars corresponding to weighted GENIE3 results, the thiner, lighter bar corresponds to the product with Z-scores.

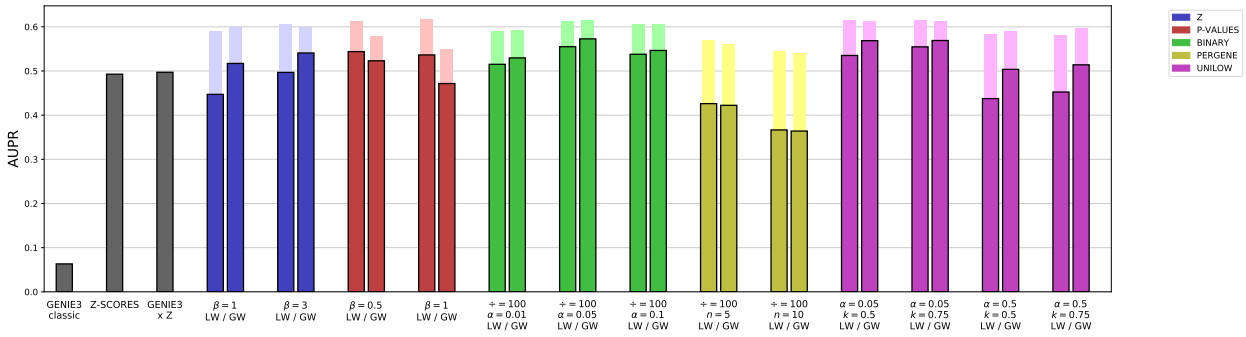


Figure 4.6: AUPR results for Network 1 using different weights with the local or global approach. Lighter bars correspond to the additional product with Z-scores.

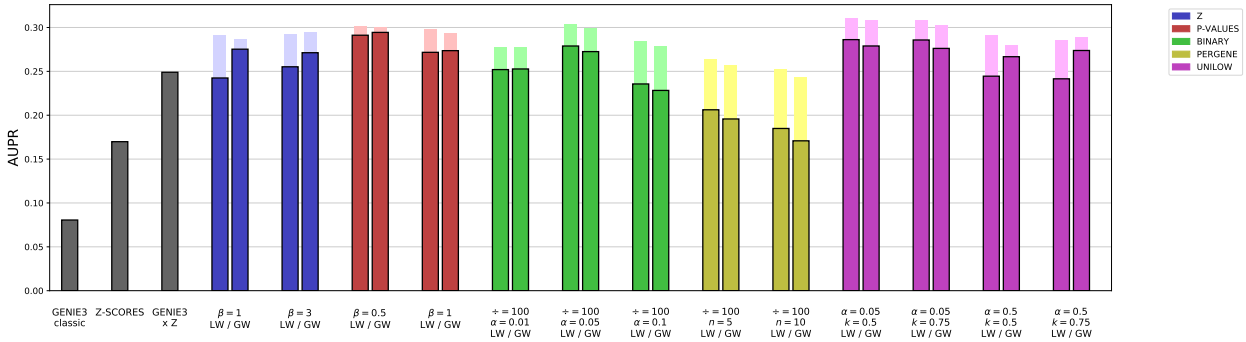


Figure 4.7: AUPR results for Network 5 using different weights with the local or global approach.

4.3 Results analysis

4.3.1 First observations

First, by looking at the two first bars, it seems obvious that Z-scores are quite powerful compared to GENIE3, at least in this context where all knockouts are available. This clearly outlines the importance of taking them into account whenever possible, even if GENIE3 is usually better in most cases.

Then we observe that the simple product of both methods yields even better AUPR results, except for network 2. The effect is however not that significant here. Although this will be shown later, we can already mention that this effect is clearer when training GENIE3 on all data instead of only time-series data.

Moving to weighted GENIE3 results, we see that results are globally quite good and many methods manage to perform better, which is good news. Comparing the global method to the local method for equivalent weights, we see that the first one is usually better, except for the P-VALUES method. However, there is no clear winner and differences tend to stay quite small.

Looking at the additional Z-score product, we have here a clear result : multiplying the output of weighted GENIE3 by the Z-scores always improves the AUPR. Moreover, we see that the bar increasing the most are the lowest ones. For AUROC, however, the effect might be positive or negative, but the difference is not significant.

Let us now compare the different weight creation methods. The clearest result is that the PERGENE method does not work very well compared to the others. This could be caused by the fact that a high weight can somehow induce a higher interaction score even when the interaction is not a true one. Although the problem exists for all methods, it is probably more important for the PERGENE technique.

For other methods, however, there is no obvious winner. Using binary weights does not seem to degrade that much the AUPR quality, showing that precise weights are not really important.

4.3.2 Further analysis with gold standard based weights

In order to understand better the weighted GENIE3 techniques, we will take advantage of our knowledge of the gold standard network. Indeed, weights build from knockouts are supposed to give information about the true network. However, this information is not perfect, although it impacts the results positively. Now, what would happen if these weights were better, and in particular if they represented the true network without errors ? What we can do is building artificial weights using the gold standard, which will allow us to control the information integrated in GENIE3. This will give us hopefully interesting theoretical results.

We will first build our weights in a similar way as in the BINARY method. True interactions will get high weights while the others will get low weights. We test both local and global approaches for different ratio values. AUPR results are displayed in Figure 4.8 for network 2 (see Appendix B.1.3 for other networks).

These results show that the weighted GENIE3 method is able to use correct a priori information, although it will never give perfect results. This also shows the superiority of the global approach, at least when a priori information is close to the true network. Now let's assume that we were able to obtain a superset of all

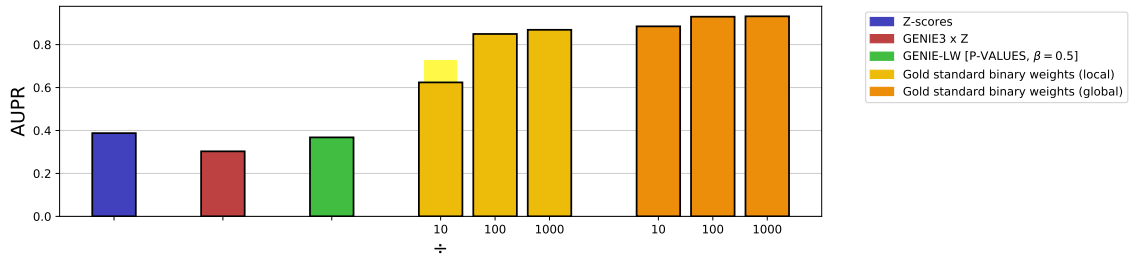


Figure 4.8: Analysis of the weighted GENIE3 method with binary weights, when assuming perfect a priori information (Network 2)

true interactions, i.e. a set of interactions which contains the gold standard and additional false positives. This can be quite close to reality as Z-scores tend to find the correct interactions but indirect links as well. However, here, the wrong interactions will be chosen randomly.

For a fixed n (the number of high value weights), weights are built like this :

1. Every true interaction obtains a high value.
2. Given $|G|$ true interactions, $n - |G|$ false interactions are drawn randomly and are given a high value.
3. All other interactions obtain a low value (equal to 1).

We will compare the use of these weights with the results of the BINARY method. Additionally, we show results for the BINARY method when we have artificially given very high Z-scores to all true interactions, so that they are systematically chosen. The difference with the method being tested is that the false positives are not random in this case.

Figure 4.9 shows AUPR results for the global approach on Network 3. Results are similar for the local approach and other networks, but more visible in Network 3 (see Appendix B.1.4 for additional figures). The AUROC follows the same trends too. Obviously, increasing the number of "false positives" decreases the AUPR. Also, despite the false positives added, giving a high weight to all true interactions improves the AUPR score quite a lot with respect to the classic BINARY method. More surprisingly, we obtain slightly better results when the false positives are not chosen at random. However, when we add the Z-score product at the end, we observe the opposite. Intuitively, it is understandable that Z-scores provide more new information to the first method than the last, which already used them during weight creation.

Another interesting result is that even when $n = 5000$, i.e. half of all interactions, we still manage to obtain better results than GENIE3 alone. When looking at the last bar for the "gold standard" technique, the additional Z-score product even outperforms the best previous results. This suggests that a method good enough to place all true interactions in the first half of the ranking could be used to create weights. If this method is not correlated with Z-scores, then we could potentially obtain better results at the end (if we still use Z-scores for the final product).

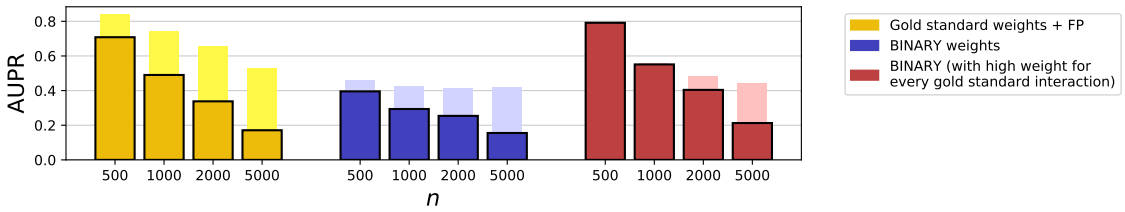


Figure 4.9: Comparison of different methods using binary weights (global approach, network 3)

4.3.3 Influence of the type of data

All previous results were obtained by training GENIE3 on time-series data only, using knockouts only to provide Z-scores. We would like to know if this separation is wise or if it might be preferable to train on all data instead. We will thus repeat some of the previous methods, training GENIE3 on (1) time-series data, (2) knockout data, (3) time-series + knockouts.

As we will see, knockouts tend to be more useful, we thus add a fourth case where we use, again, all data. The difference will be in the importance assigned to each sample. Knockouts sample will have an importance value of 2 instead of 1, telling GENIE3 to focus more on these samples. This is done by using the associated weight of each sample when computing impurity to determine the best split at nodes.

Figure 4.10 shows AUPR results for networks 1, 3 and 5, using the local approach. We can immediately see an increasing trend when we go from case (1) to case (4). Using knockouts only tend to be, with very few exceptions, better than time-series only, although there are two times less samples in this case. However, when multiplying by the Z-scores, time-series outperform quite often knockouts, probably because of Z-scores being actual new information in the first case. Using all data seems to be the best strategy, although knockouts alone are sometimes slightly better. By giving knockouts a higher weight, we indeed improve results systematically, even if the increase is very small.

Note that we used time-series in the most simple way, although there exists ways to use them more cleverly. By introducing a time lag (shifting output by one time step), GENIE3 is able to reach much better results with time-series [9].

4.4 Variants

In this section, we will quickly analyze a few variants that were tested. As we will see, none of them seems to be practically interesting in the end. That is why we will not go into too much details. However, these negative results help us to better understand our problem and solutions, which is why we will still talk about them.

4.4.1 Global method variants

Other weight corrections

We described in chapter 3 the method used to derive global weights from local weights and explained the problem of very large weights that had to be cut off. We

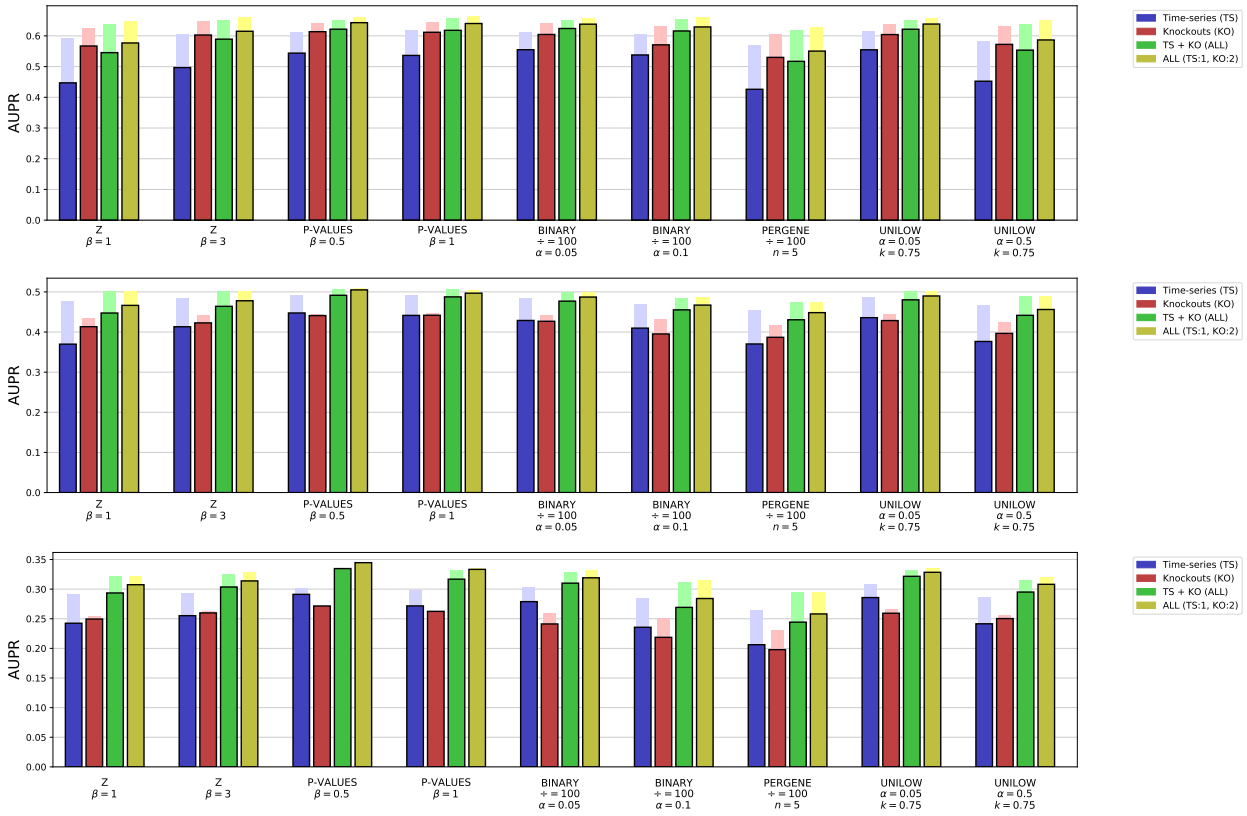


Figure 4.10: Influence of the type of data used for training on AUPR results (from top to bottom : networks 1, 3 and 5), local approach

explored some variants. In all cases, we start by scaling weights so that their sum is K and then set the weights to 1 when they are too high. Then, we can :

1. Use the previous procedure.
2. Increase the highest less-than-one weights first. Formally :
   (a) Compute the residual K_{res}.
   (b) Repeat until the residual is zero :
      i. Select the highest weight w_{ij} which is not equal to one.
      ii. Increase its value by K_{res} if $K_{res} < 1 - w_{ij}$, or by $1 - w_{ij}$ otherwise (set it to one).
      iii. Decrease K_{res} by the same amount.
3. Do nothing (simple cut) : the sum will not be equal to K but we will not increase some weights artificially.

Figure 4.11 summarizes the different methods.

"Local" sampling

In the global method, we use a different type of random selection than for the local approach. However, nothing prevents us from using the same type of selection.

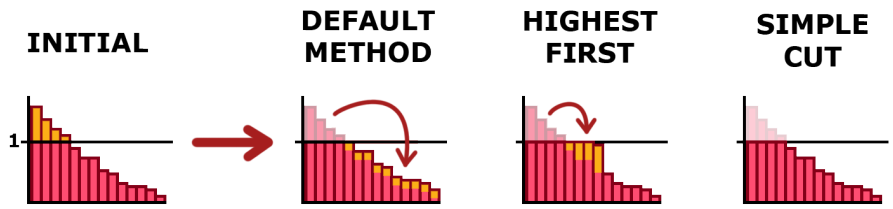


Figure 4.11: Summary of weight correction methods.

That is what we will try here. Previous problems with weight creations do not apply anymore.

Results

Results for the three previously described methods, compared to the original global approach, are shown in Figure 4.12 for network 1. Except for the "highest first" correction which yields strange decreases, other methods tend to be equivalent. It is thus probably safe to keep the initial method.

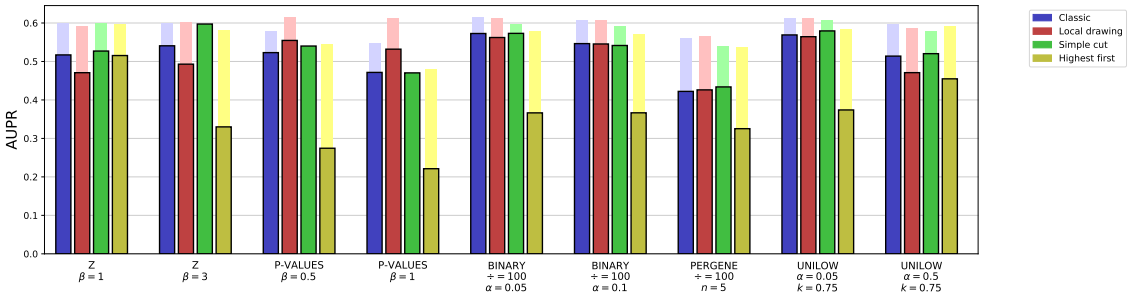


Figure 4.12: Comparison of AUPR results for different variants of the global approach (Network 1). Mean values of K_{res}/K, in order : 0.0332, 0.5020, 0.5651, 0.7033, 0.3938, 0.3920, 0.2397, 0.4544, 0.1124.

4.4.2 Filtered Z-scores

We know that Z-scores are sensitive to indirect links, like most other existing GRN inference techniques. Many methods have been proposed to counter this problem by applying some post-processing treatment to the score matrix, in order to "remove" the feedforward interactions. Although feedforward interactions exist in real networks, we usually find too much of them and it is then better to remove all of them. That is the principle of the method of Pinna et al. [20] which was one of the best performers in the DREAM4 challenge.

We would like to know if using this kind of filtering on Z-scores before integrating them can give better results. The technique we will use works like this :

1. A threshold value t is imposed to extract a first directed network from Z-scores. In our experiment, we chose $t = 2.5$.

	Net 1	Net 2	Net 3	Net 4	Net 5
With filter	0.683	0.385	0.373	0.435	0.206
Without filter	0.493	0.388	0.381	0.368	0.17

Table 4.1: Comparison of the AUPR of Z-scores with and without feedforward links filter.

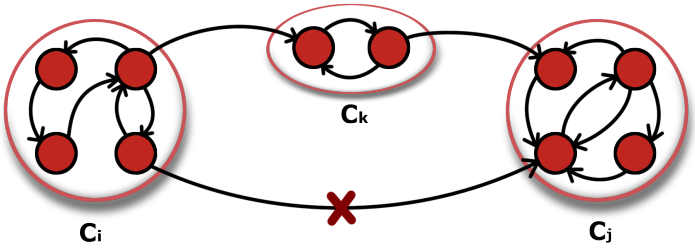


Figure 4.13: Illustration of the filtering process

2. From this network, we compute the associated condensation network, which is built by aggregating every strongly connected component[1] into a single vertex. In practice, this was done using Kosaraju's algorithm.

3. The links to delete are the links for which : (1) the endpoints belong to two different strongly connected components C_i and C_j, and (2) there is a path of length 2 or more between C_i and C_j (see Figure 4.13).

4. These links get a Z-score equal to zero (this step is different from the method of Pinna et al., to keep a correct distribution of values).

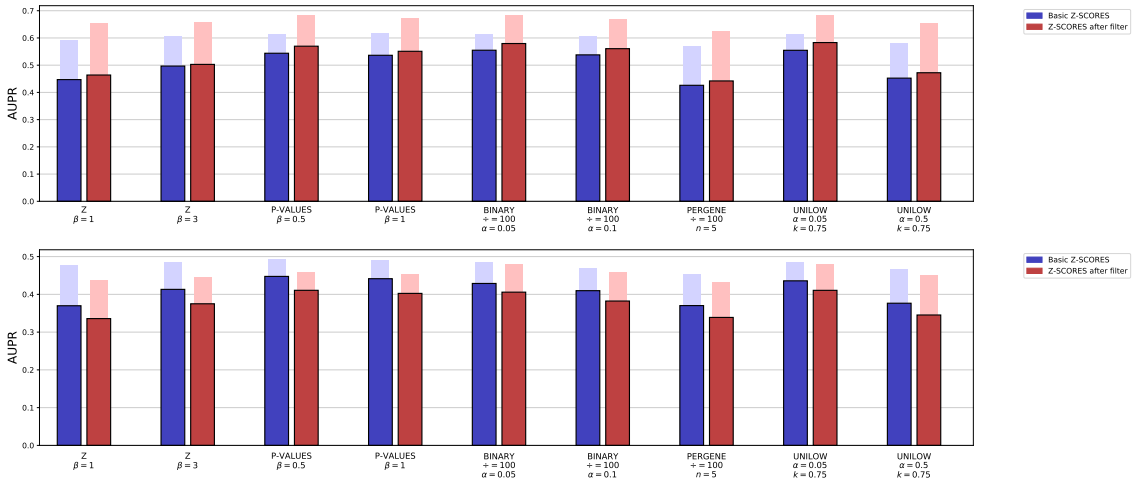


Figure 4.14: Comparison of AUPR results between the use of classic Z-scores or Z-scores filtered with Pinna et al. method (local approach, networks 1 and 3). Lighter bars correspond to products with the initial, unfiltered, Z-scores.

[1] A strongly connected component corresponds to a set of vertices for which there is a directed path between every pair of vertices

These filtered Z-scores were used at the weight creation step but not for the additional product at the end. AUPR results for two networks are given in Figure 4.14. Nothing surprising appears : when the filter improved Z-scores results, it usually provides better weights too. The opposite is also true as shown for network 3.

4.4.3 Best Z-scores as regulators

To end this section, we will try another simple method that was tested, based on similar ideas, but showed less good results. The idea was to fix a threshold on Z-scores to keep the highest ones. The corresponding interactions are then used to choose the only possible regulators for each gene. When using GENIE3, we then train a random forest for each gene (so as to find its regulators) but limit the possible regulators to the ones chosen before. Note that each gene has different potential regulators, as well as a different number of them. The possible interactions are chosen by fixing a threshold α on p-values, similarly to what was done before.

Results (see Figure 4.15) show that, as we decrease the threshold α, AUPR values outscore GENIE3 alone and come closer to Z-scores result. However, they tend to stay close to the Z-score level, with sometimes a slight improvement. When multiplying again by Z-scores, we see that classic GENIE3 gives the best results, showing that the method has thus no real interest.

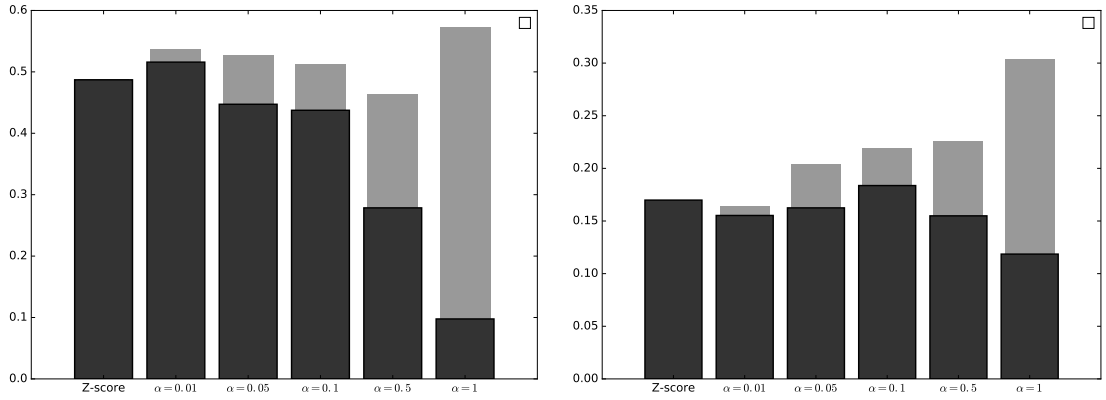


Figure 4.15: AUPR results of the "Best Z-scores as regulators" for different values of α (Networks 1 and 5)

4.5 Conclusion for the ideal case

Before moving to the difficulties of realistic cases, let us first summarize the outcomes of this chapter. First, we have seen the superiority of Z-scores over GENIE3 when all knockouts are available, showing the importance of taking them into account when possible. We have also seen that a product between GENIE3 and Z-scores, although a very simple method, could already be very interesting.

Then, we analyzed more complex techniques. These new methods proved to be successful (except for the PERGENE one), yielding better AUPR results than the

simple product. The only condition is to choose parameters wisely. Among the successful methods, no clear winner could be chosen. Even the BINARY method, which uses much less diverse weights, works quite well. There was no clear winner between the global and the local approach too, although the first one has the advantage of being faster (one selection per tree instead of one selection per node) and easier to implement.

A last and important result on these methods was the possibility to further improve results almost systematically (at least for the AUPR) by multiplying the outcome by Z-scores.

We then performed some analysis by using gold standard based weights. It showed some theoretical limits obtainable with our approach, since results were not perfect even with "perfect" weights. We also highlighted the possibility of improvements.

Finally, we also concluded that using all available data to train our Random Forests was usually the best method, even if knockout samples are then used at two different steps. We also analyzed many variants with less interesting conclusions.

All these results were obtained on the artificial networks of the DREAM4 challenge, composed of only 100 genes. We should thus stay cautious about these conclusions.

Chapter 5

Knockout integration II

In the previous chapter, we analyzed different methods for knockout integration in a very ideal case where all knockouts would be available. In this second part, we will move to more realistic situations where only a few of them is given and where networks are much larger. In section 5.1, we will start by detailing the particularities of real datasets and decide which problems we will try to handle. Then we will see what we can do to tackle these problems in section 5.2. Finally, we will show results on different experiments and datasets, and discuss these results (section 5.3).

5.1 Particularities of real datasets

In this first section, we give an overview of different particularities that one can encounter in real datasets. We briefly explain what will be done later about them.

5.1.1 Limited number of knockouts

The main problem of real datasets is their limited number of knockouts. Indeed, obtaining a knockout sample is more difficult and time-consuming than a simple sample, since it requires to perform a precise manipulation. In practice, this can thus be done for only a few genes.

This is very problematic since our previous methods were all based on the knowledge of Z-scores for all $p \times p$ interactions. If we have only N_{KO} knockouts, then we can only compute $p \times N_{KO}$ Z-scores. If we want to still be able to use the same techniques, then we need a way to infer values differently for the missing Z-scores. This will be discussed in the next section.

5.1.2 Repeated knockouts

When computing Z-scores, we assumed the existence of **one** knockout for each gene. But it might appear sometimes that several knockouts have been done for one single gene. In this case, we thus have redundant information, which is more of an advantage than a problem. We decided to handle such cases by computing the corresponding Z-scores for each sample corresponding to the same gene and averaging them in the end.

5.1.3 Double knockouts

We also assumed that when a knockout was done, one single gene was knocked out at a time. However, nothing prevents the simultaneous knockout of two, three or even more genes. Again, the Z-score method is not adapted to that kind of data. Given the very few occurences of such data in the datasets we will use later, we will simply ignore them.

However, it might be interesting to think of more clever ways to use such data when they are more numerous. For example, if we have a knockout of gene A, and a simultaneous knockout of genes A and B, then it might be possible to observe the deviations after a knockout of gene B only by comparing the two samples.

5.2 Missing Z-scores completion methods

We will now explain the different methods we tried for handling the missing knockouts problem. Let's start by giving some notation. We are trying to obtain "Z-scores" or some values replacing them, for a network of p genes. We thus need to obtain $p \times p$ values. We have only N_{KO} knocked out genes, and thus $N_{KO} \times p$ true Z-scores. If genes are numbered from 1 to p, then $KO \subseteq \{1, ..., p\}$ is the set of knocked out genes and $|KO| = N_{KO}$. We want to get replacement Z-scores for genes in $\overline{KO} = \{1, ..., p\} \setminus KO$.

The problem is thus the following : using available knockouts and other samples, find an appropriate value for Z-scores $z_{ij} \quad \forall (i, j) \in \overline{KO} \times \{1, ..., p\}$. Our ideas are presented in the next paragraphs.

Zero In some cases, keeping a value of zero for all missing Z-scores might be good enough, mainly when using binary weights or the UNILOW method. Indeed, in these cases, zero values will still be converted to a non-zero weight in the end, although associated interactions will be seen as "probably not interesting" for GENIE3.

Mean Another simple idea is to use the mean of other Z-scores to replace the missing ones. The mean is computed for interactions going to one particular gene, and replaces all missing values of interactions going to that same gene. Formally, Z-scores z_{kj} for $k \in \overline{KO}$ obtain a value equal to

$$\frac{\sum_{i \in KO} z_{ij}}{N_{KO}}$$

This way, missing values obtain an average score, which roughly corresponds to a "we don't know" position, since high Z-scores are seen as more important interactions and low Z-scores as not important ones. **Remark** : since many interactions will be given the exact same value, techniques that select the "top n" interactions to build weights (for example, the BINARY method where the high-value interactions are selected like this instead of using a threshold α) might have problems since the ranking would be arbitrary. We will thus avoid techniques based on such "top n" selection.

Fictional KO Instead of using simple constant values, we might want to think of more clever methods. What we would like is replacement values that convey some information too. One way of doing that is using the "no knockout case" that was described in Chapter 3. For each gene in $\overline{KO}$, we choose one sample that takes the role of the missing knockout sample. The chosen sample is the one were the gene expression is minimal, since it is the closest to zero, which is what we would have in a knockout. The Z-scores can then be computed from this sample.

However, since these are not real knockouts, the deviations tend to follow a different distribution than the real knockouts. When mixing both types of Z-scores, the ranking will not be consistent. It is thus necessary to modify the inferred values so as to get similar distributions. Z-scores distribution looks like a half bell-shaped curve starting at approximately 0 (see Figure 5.1). The way that was chosen to modify the values is to "contract" or "dilate" values so that the 95th percentile for fictional knockouts is the same as for real knockouts (this way we make sure we have a similar range of values for "normal" values). Contrary to the mean method, this correction is done at a global level, not a gene-by-gene level. Figure 5.2 illustrates the idea. In practice, we modify the range of values by multiplying them by a constant factor f = (95th percentile of true Z-scores)/(95th percentile of fake Z-scores), so that their 95th percentiles are equal. All these choices were made because they had positive impacts for the DREAM4 networks, but it is difficult to find a method which would be a true, theoretically correct method. We should thus stay critical of this method since it remains quite arbitrary.

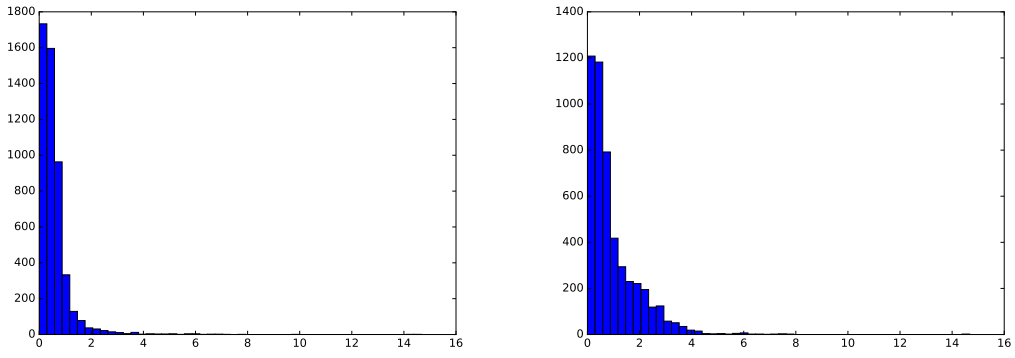


Figure 5.1: Examples of Z-scores distribution (DREAM4 Net 1) for true Z-scores (left) and inferred Z-scores (right)

Petralia method Another more clever method was proposed by Petralia et al. [19]. Their idea is to borrow information from available Z-scores to infer the other ones. It works in three steps :

1. For any gene $g_k \in KO$, p-values $P_{k \to j}$ are computed. We then define the set of true regulatory events as all $g_k \to g_j$ where $P_{k \to j} < 0.01$.

2. For each pair of genes (g_h, g_k), a measure of similarity is computed :

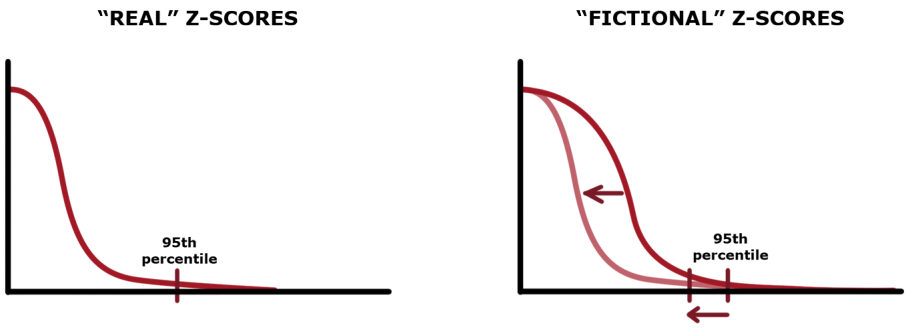


Figure 5.2: Illustration of the distribution correction idea

- First, we compute (C_h, C_k), the sets of knocked-out genes which affect genes (g_h, g_k). When $g_h \in KO$ AND $g_k \in KO$, we also compute (E_h, E_k), the sets of genes affected by knocking them out.
- We use the Jaccard index $J(A, B)$[1] to define the similarity measure $G_{h,k}$ between g_h and g_k :
  - $G_{h,k} = (J(E_h, E_k) + J(C_h, C_k))/2$, or
  - $G_{h,k} = J(C_h, C_k)$

3. For genes $g_s \in \overline{KO}$, missing weights are defined using the formula :

$$w_{s \to j}^{\overline{KO}} = \frac{\sum_{l \in KO} G_{s,l} w_{l \to j}^{KO}}{\sum_{l \in KO} G_{s,l}}$$

In our case, the last step will be slightly different since we will not directly "interpolate" weights but instead Z-scores. Weights will be derived from the inferred Z-scores afterwards. For the Z method ($w_{ij} = z_{ij}$) this does not change anything.

Co-occurence This last idea is based on the following intuition : if two genes have extreme values at the same time, then it is more likely that they are regulating each other. The method is able to compute a value for each z_{ij} even when no knockouts are available. It works like this :

1. For each gene expression value, a "deviation from the mean" value is deduced, similarly to how Z-scores are computed. The deviation dev_k^j for gene j in sample k is :

$$dev_k^j = \left| \frac{X_k^j - \mu_j}{\sigma_j} \right|$$

where μ_j and σ_j denote the mean and standard-deviation of gene j computed from all samples.

2. A threshold t is chosen, and the values higher than this threshold are marked as *exceptional values*. In practice, we will always choose $t = 1.5$, unless another value is specified.

[1] $J(A, B) = \frac{|A \cap B|}{|A \cup B|}$, which measures the similarity of two sample sets

3. For each possible interaction $i \rightarrow j$, set the score s_{ij} to 0.

4. Browse every sample again. For each sample, when two genes i and j have both exceptional values in this sample :

   - Increase s_{ij} by the product of deviations $dev_i \times dev_j$.
   - Increase s_{ji} by the product of deviations $dev_i \times dev_j$.

The scores matrix is thus symmetric. We then keep only the scores corresponding to missing Z-scores, and apply a distribution correction (similarly as for fictional KO). The idea of this method was mostly to give high weights to more *potentially interesting* interactions, even if we know that high scores could be due to other reasons.

5.3 Experiments and results

Now that we have defined how we are going to deal with special problems of real datasets, let us test these ideas in practice.

5.3.1 DREAM4

Tests description

We will start by re-using the DREAM4 dataset, from which we will artificially remove some knockouts. This will give us more control and will allow us to ignore other problems for the moment. Moreover, given the small size of the network, we will be able to do a lot of tests and already select the best methods, in order to spare computational time for later tests on larger networks.

We thus use the same five networks of the previous chapter, composed of 100 genes each, with 210 time-series samples and one knockout for each gene. Since we want to test the missing Z-score inference techniques, we remove randomly a chosen number of knockouts and compare techniques between each other. Multiple percentages of removed knockouts will be tested to observe how AUPR/AUROC scores evolve according to it. For each percentage, the tests will be repeated a certain number of times with different knockouts to reduce variance due to better or worse knockouts being chosen.

Here is a precise summary of the tests performed :

- For each number N_{KO} of kept knockouts in $\{50, 20, 10\}$:
  - Repeat 10 times :
    - Select N_{KO} knockouts randomly.
    - Compute Z-scores from these knockouts.
    - For each missing Z-scores completion method in {Zero, Mean, Fictional KO, Petralia, Co-occurence} :
      - Compute missing Z-scores using selected method.
      - For each GRN inference method to test (see below) : run using available data and Z-scores, and compute AUPR and AUROC.

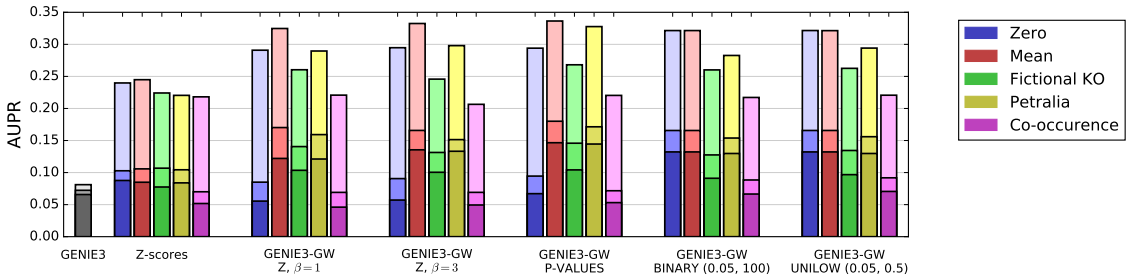


Figure 5.3: AUPR results for DREAM4 Net1. From darkest to lighest : 10, 20 and 50 available knockouts.

        - Additionally, compute results for classic GENIE3 (using time series + selected knockouts).
    - Average AUPR and AUROC scores over all ten iterations.

- Show results

The GRN inference methods that will be tested are the following :

- Z-scores only
- Weighted GENIE3 (global, $T = 500$ trees) :
    - Z : $\beta = 1$, $\beta = 3$
    - P-VALUES : $\beta = 1$
    - BINARY : $\alpha = 0.05$ and $\div = 100$
    - UNILOW : $\alpha = 0.05$ and $k = 0.5$

The PERGENE method has been rejected since it obviously gave bad results in the ideal case. Given the huge number of combinations, tests will be performed on networks 1, 3 and 5 only.

Results are given in Figure 5.3 for network 1. Other networks are available in Appendix B.2.1, as similar conclusions will be deduced from them.

Observations

A first observation is that, even with a limited number of knockout, weighted GENIE3 manages to outperform both GENIE3 and Z-SCORES again. This shows the interest of the method in non ideal cases. However, scores were of course better when all knockouts were available (best results around 0.57 for the AUPR in network 1).

When comparing the different types of weights, it is again difficult to identify a distinct winner. Looking at the highest bar among each group, the binary method seems to have slightly less important performances, especially for the AUROC, but it is not very significant.

More interesting conclusions can be drawn concerning the different techniques to replace missing Z-scores. The ZERO and the CO-OCCURENCE methods seems to be below other methods in most cases (especially the AUROC for the Zero method) and should thus probably be discarded later. It is more difficult to decide between the

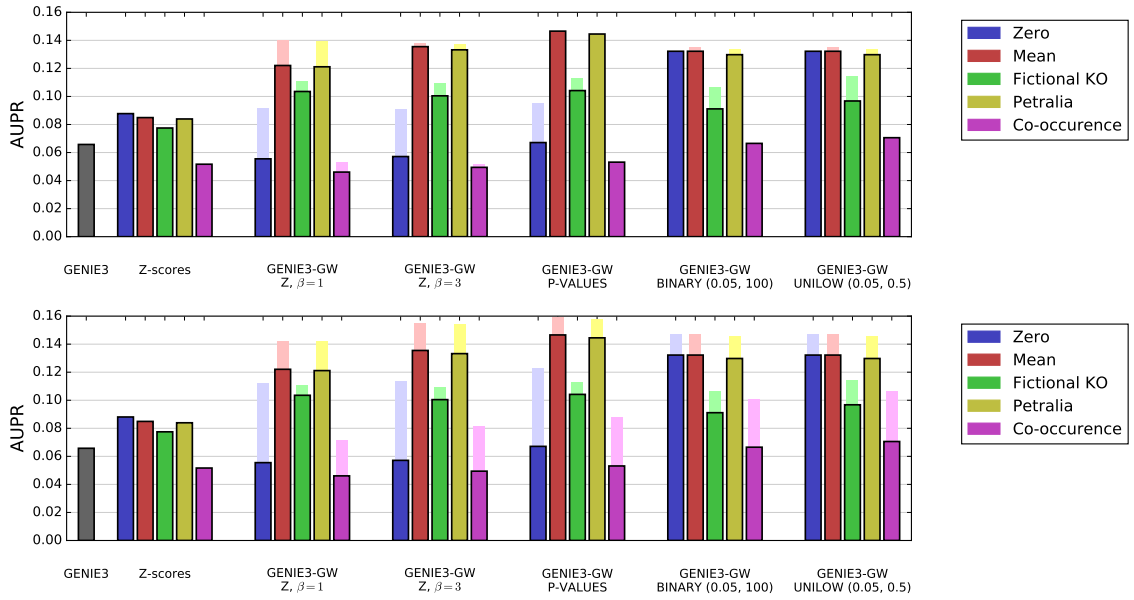


Figure 5.4: Results for Network 1 of the DREAM4 dataset, with only 10 available knockouts. Lighter bars correspond to a product with the corresponding Z-score (top) or the Fictional KO-corrected Z-score (bottom)

three others : Mean, Fictional KO and Petralia. When we look at Z-scores only, Fictional KO often seems better, especially in the AUROC category where the difference is huge. On the other hand, when used as weight, it is usually less successful than the two others. Surprisingly, the very simple Mean method yields very similar, even better (according to AUPR) results than more complex methods. Petralia is very close, although its results seems on average slightly lower despite its complexity.

We might also wonder what would be the effect of multiplying GENIE3's output by the Z-scores again, as it was done before. We will use the same Z-scores as used to build the weights. So it will again depend on the completion method. Figure 5.4 (top) shows AUPR results for network 1 with $N_{KO} = 10$. The lighter bar indicates the increase when doing the product. As we can see, it is mostly the Zero, Fictional KO and Co-occurence methods which are impacted positively, and the improvement stays quite small.

Since the best Z-scores are provided by the Fictional KO method, we could also try to use these ones systematically for the additional product, even when they were not used for the weights creation. Results are shown in Figure 5.4 (bottom). Now, we have a systematic increase, and the increase is more significant. The effect is similar for other networks although it is less visible. This shows that it might be interesting to combine several completion methods according to the final usage. We will limit ourselves to this combination since testing them all would result in too much data.

Further analysis using gold standard

Similarly to what was done in Chapter 4, we can use our knowledge of the gold standard network in order to get a better idea of the behavior and limits of our

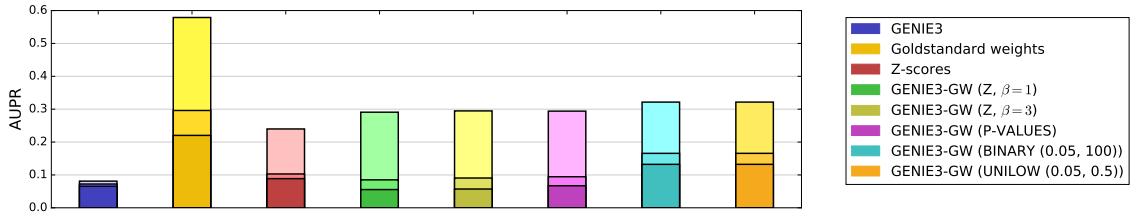


Figure 5.5: Comparison of AUPR results between gold standard weights ($\div = 100$) and other methods (DREAM4 dataset, Network 1, ZERO completion). From darkest to lighest : 10, 20 and 50 knockouts.

methods. Since this chapter is focused on the problem of limited available knockouts, we will create weights based on the gold standard again but we will simulate our ignorance of some knockouts.

As a reminder, the idea in the previous chapter was to build binary weights where the high-value interactions correspond to the true interactions and the low-value ones correspond to non-existent interactions. This way, we simulated a case where the Z-scores would have been "perfect". We saw that despite this perfect information, weighted GENIE3 did not reach a perfect AUPR in the end, although close to one. We had also added false positives to get closer to more realistic weights, which gave us some interesting results.

Here, we want to simulate the fact that weights were built from partially known Z-scores. In a first approach, we will assume that the limited number of Z-scores that could be obtained provided a perfect information:

- For interactions whose Z-score is supposed to be derived from a real knockout, a high weight is given to each true interactions (and only them).
- For other interactions, a low weight is systematically given (this would correspond to the ZERO completion method). It would be difficult to simulate other completion methods in this case.

We use these weights with the same set of experiments as before ($N_{KO} = 50, 20$ or 10, ten iterations, networks 1, 3 and 5). We compare results with the other results for the ZERO completion method. Results are given in Figure 5.5 for Network 1.

We naturally observe that these gold standard based weights give better results. However, it is clear that the limitation of "knockouts" also limits the final AUPR. The AUPR actually seems approximately proportional to the percentage of available "knockouts" in this case. Therefore it is difficult to obtain very good results when knockouts are so limited, at least with the ZERO method. Results are similar for other networks.

Gold standard based weights with random removal

Out of curiosity, we also performed similar tests using gold standard based weights, but instead of removing true interactions corresponding to specific regulators, we removed a fixed percentage of them randomly. In other words, given n true interactions, we remove a percentage P of them, and build binary weights from these $n - Pn$ remaining true interactions, just like before. We used a ratio $\div$ of 100. Results are

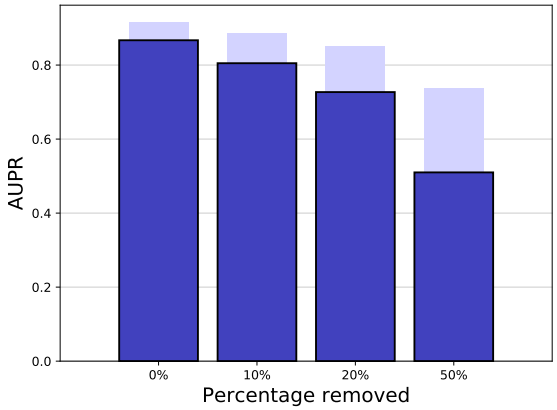


Figure 5.6: AUPR results for GENIE3-GW with gold standard weights, where a percentage of true interactions was randomly removed (DREAM4 Network 1)

displayed in Figure 5.6 for Network 1 (again, results are similar for other networks). Obviously, as the percentage of true interactions removed increases, the performance decreases. Also, we observe again that the effect of the additional product is more important when more interactions were removed.

Moreover, we were interested to know how well GENIE3-GW could predict the removed interactions. To do so, we evaluated the AUPR again but we ignored the true interactions that were used to build the weights. We thus evaluate the prediction of the unknown subpart of the network (non-existent interactions + removed true interactions). Results are compared with GENIE3 alone (no weights). The AUPR is given for Network 1 in Figure 5.7. The difference of value between different percentages is not important here. Indeed, the results for different percentages are not comparable since they were computed from different sets of interactions. However, we can compare GENIE3-GW and GENIE3 inside each group. It is clear that using weights is not better than using none. This thus shows that using weights that give more importance to some interactions will only improve the prediction of these interactions and will not have an indirect positive effect on the others. This problem is somehow related to the problem that will be treated in Chapter 6.

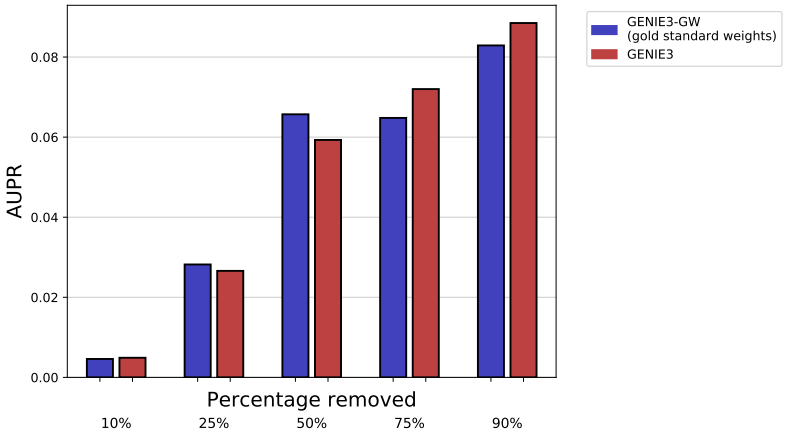


Figure 5.7: Comparison of GENIE3-GW (with gold standard based weights) and GENIE3 alone on the prediction of removed interactions.

5.3.2 DREAM5

Dataset description

While results on the artifical networks of the DREAM4 dataset are quite good, real networks tend to be much more difficult. The DREAM5 challenge, which took place in 2010, one year after the DREAM4 challenge, proposed to evaluate GRN inference techniques on new more realistic networks. Four networks, with the following characteristics, were included :

	Name	**Nb of genes**	**Samples**	**KO genes**	**Nb of TF**[2]
Net1	In silico	1643	805	35	195
Net2	S. aureus	2810	160	2	99
Net3	E. coli	4511	805	39	334
Net4	S. cerevisiae	5950	536	13	333

Table 5.1: DREAM5 networks characteristics

We are thus dealing with much larger networks, with 2% of knockouts at most, which makes the inference much more difficult. Note that several knockouts were usually provided for the same genes, as well as double knockouts involving genes already individually knocked out. The "KO genes" column counts the number of genes, not the number of samples. The first network is again an artificial one while the three others correspond to real organisms. Gold standards were provided for networks 1, 3 and 4. Although a gold standard for network 2 has been provided later on, we will ignore this network and use the MATLAB scripts that were given for evaluation at that time. These scripts compute the AUPR and the AUROC based on the 100,000 first interactions of the ranking, which yields quite different values than using all interactions. This will allow comparison with other methods.

Tests description and results

Given previous observations, we will not test every Z-scores completion technique and instead focus on two of them. First, we test the MEAN method because it was the most successful. Second, we chose to test the FICTIONAL KO because it is more complex and we would like to see its effect on other datasets, although its performance was lower. Given the datasets dimensions, tests are more computationally demanding, which forces us to limit ourselves. We also limit the number of trees per random forests to $T = 500$ for the same reason. Results are displayed in Table 5.2.

Observations

It is immediately clear that our new methods did not work well on this dataset. Indeed, we never manage to obtain better results than GENIE3 alone, except for Network 1 with the P-VALUES method. Even worse, there is a net decrease for most cases. Given the very poor information provided by Z-SCORES, it is clear that they influenced GENIE3 negatively this time.

[2]Transcription factors

	AUPR			AUROC		
Method	**Net1**	**Net3**	**Net4**	**Net1**	**Net3**	**Net4**
GENIE3	0.288	0.095	0.021	0.812	0.618	0.518
GENIE3 x Z (mean corrected)	0.319	0.093	0.02	0.815	0.621	0.515
GENIE3 x Z (fictional KO)	0.239	0.056	0.020	0.788	0.595	0.514
Mean :						
BINARY ($\div = 100$, $\alpha = 0.05$)	0.225	0.047	0.02	0.807	0.611	0.515
P-VALUES ($\beta = 1$)	0.304	0.078	0.021	0.815	0.621	0.516
UNILOW ($\alpha = 0.05$, $k = 0.5$)	0.225	0.045	0.02	0.804	0.609	0.515
Z ($\beta = 1$)	0.285	0.072	0.02	0.8	0.612	0.514
Z ($\beta = 3$)	0.286	0.075	0.02	0.801	0.612	0.515
Z-SCORES (only)	0.077	0.018	0.018	0.599	0.521	0.501
Fictional KO :						
BINARY ($\div = 100$, $\alpha = 0.05$)	0.193	0.031	0.019	0.746	0.587	0.514
P-VALUES ($\beta = 1$)	0.195	0.030	0.020	0.748	0.573	0.516
UNILOW ($\alpha = 0.05$, $k = 0.5$)	0.197	0.031	0.019	0.746	0.591	0.514
Z ($\beta = 1$)	0.245	0.045	0.020	0.769	0.592	0.517
Z ($\beta = 3$)	0.225	0.028	0.020	0.731	0.559	0.515
Z-SCORES (only)	0.036	0.015	0.017	0.577	0.504	0.500

Table 5.2: DREAM5 results using scripts

Nevertheless, the good result for the P-VALUES method shows that it was still possible to have a positive influence despite that. However, it is difficult to understand why this method worked and the others did not. This result is also not as interesting as it seems. Indeed, using GENIE3 $\times$ Z on Network 1 gives an AUPR of 0.319 which is better, while applying the additional product on weighted GENIE3 results do not improve them (see Table B.2 in the appendices).

Contrary to previous experiments on the DREAM4 dataset, the mean inference method is distinctly better than the fictional KO one. This could be due to our imperfect distribution correction, but we noted that Z-SCORES were still better with this correction than without it in this case.

This experiment showed that we must stay very cautious when we use weighted GENIE3 on datasets with so few knockouts compared to the number of genes.

5.3.3 Other yeast dataset

We performed similar tests on another dataset corresponding again to the S.cerevisiae (yeast). This dataset is the one used by Meinshausen et al. in their paper about causal inference from gene perturbation experiments [14], which will be discussed in chapter 6. Contrary to the samples of DREAM5, this dataset contains much more knockouts and is thus more interesting. The precise characteristics are listed in Table 5.3. As no gold standard and transcription factors were provided, we reused the ones of the DREAM5 challenge. Some differences occured between the two gene sets, but as they were small, the gold standard could still be used quite reliably. Only five transcription factors were not in the set of genes of this yeast dataset. Only $\approx 24\%$ of all genes have been knocked out, but this number reaches 82.6% if

we look only at transcription factors. This means that the problem is much easier since we have a knockout for the majority of genes for which it is necessary.

Number of genes	6170
Number of knockouts	1479
Number of wild-types	262
Number of TF	328
Number of TF with a KO	271

Table 5.3: Yeast dataset characteristics

We test the same methods as for DREAM5. Results are given in Table 5.4. This time, contrary to DREAM5 results, an improvement is observed when we use weights. However, we never manage to beat to simple product GENIE3 × Z. The MEAN method gives superior results than the FICTIONAL KO one, except when the additional product is involved. However, this additional product is not always beneficial, especially with the MEAN method.

Weights	**AUPR/AUROC**	**AUPR/AUROC (x Z)**
None	0.00323 / 0.577	0.00918 / 0.614
Mean		
Z ($\beta = 1$)	0.00817 / 0.615	0.00777 / 0.613
Z ($\beta = 3$)	0.00835 / 0.614	0.00779 / 0.612
P-VALUES ($\beta = 1$)	0.00593 / 0.585	0.00709 / 0.614
UNILOW ($\alpha = 0.05$, $k = 0.5$)	0.00600 / 0.556	0.00745 / 0.581
BINARY ($\alpha = 0.05$, ratio = 100/1)	0.00588 / 0.559	0.00751 / 0.587
Z-SCORES (only)	0.00476 / 0.600	—
Fictional KO		
Z ($\beta = 1$)	0.0074 / 0.6175	0.0076 / 0.6167
Z ($\beta = 3$)	0.0065 / 0.6179	0.0077 / 0.6189
P-VALUES ($\beta = 1$)	0.0057 / 0.5921	0.0071 / 0.6223
UNILOW ($\alpha = 0.05$, $k = 0.5$)	0.0054 / 0.5698	0.0074 / 0.592
BINARY ($\alpha = 0.05$, ratio = 100/1)	0.0051 / 0.5669	0.0072 / 0.5931
Z-SCORES (only)	0.00545 / 0.607	—

Table 5.4: Yeast results

Mix of datasets

Additionally, since the network 4 of DREAM5 and this dataset correspond to the same organism, we could mix both datasets and see how well results are improved. We will use the Meinshausen dataset to infer Z-scores and the DREAM5 one as input data for GENIE3. We test the same methods as before, but only with the MEAN technique for Z-scores completion. Results are given in Table 5.5.

We can observe that results are indeed improved compared to the previous results on the DREAM5 dataset. However, there is no real improvement if we compare to Z-SCORES only, which shows that weighted GENIE3 is not really interesting here except with the additional product.

	AUPR	AUROC	AUPR (x Z)	AUROC (x Z)
Z-SCORES (only)	0.036	0.531	—	—
Z ($\beta = 1$)	0.03	0.549	0.041	0.558
Z ($\beta = 3$)	0.029	0.545	0.04	0.556
BINARY ($\alpha = 0.05$, $\div = 100/1$)	0.031	0.553	0.035	0.555
UNILOW ($\alpha = 0.05$, $k = 0.5$)	0.03	0.553	0.035	0.556
P-VALUES ($\beta = 1$)	0.031	0.553	0.036	0.557

Table 5.5: Mix of yeast datasets results

5.4 Conclusion

In this chapter, we have continued the work started in the previous chapter by focusing on more realistic cases. This included knockout duplicates and, more importantly, a limited number of knockouts. We have developed a number of techniques to complete the weights in order to be able to use the methods of the ideal case.

We have performed several tests on different types of datasets. Experiments on the DREAM4 dataset showed the interest of using the weighted GENIE3 method even when many knockouts are not available, although the performance is indeed reduced. Some completion methods showed better performances, mainly the Mean and the Fictional KO methods. Taking advantage of the gold standard, we showed the inability of the weighted method to correctly predict interactions that were not emphasized by the weights, as the method gave similar results as GENIE3 without weights.

However, on the DREAM5 dataset, which is more realistic and contains almost no knockouts, results were much more mitigated. Indeed, our new methods were almost always outperformed by the usual GENIE3 and never managed to do better than the simple GENIE3 $\times$ Z product. On the yeast dataset which contains many more knockouts, results were better but they did not beat the simple product either. The weighted GENIE3 method should thus maybe be avoided when there are too few knockouts, but perhaps simple modifications exist in order to resolve these problems.

Chapter 6

Unknown knockouts prediction

In the two previous chapters, we talked about using knockouts to infer the gene regulatory network. In this chapter, we will tackle the second problem of this master thesis : the prediction of new knockouts. We will first analyze the problem and see how we can validate our predictions. Then we will explain the methods tested and perform several tests.

6.1 Problem definition

The problem can be defined as the following. Given observational data and interventional data, including a certain number of knockouts, find the effect of new knockouts (i.e. knockouts not available). We will focus on single knockouts. The "effect" of a knockout, that we want to retrieve, is defined as the set of genes that are largely impacted, directly or indirectly, by the gene being knocked out.

To be able to validate our results, we need to define clearly these impacted genes. In their paper, Meinshausen et al. [14] proposed to define SIEs (Strong Interventional Effects). They correspond to interactions $i \rightarrow j$ with the property that j has an extreme value (it is the maximum or minimum value for this gene in the whole dataset) in the sample where i is knocked out. Although interesting for other reasons developed in their paper, we preferred to follow another approach. Indeed, their definition is very restrictive and many actual effects are not taken into account. Instead, we will use a definition based on Z-scores. A causal effect will be defined like this :

Causal effect. *Given a dataset including a certain number of knockouts, an interaction $i \rightarrow j$ is a causal effect if and only if the corresponding Z-score is large enough.*

The definition of "large enough" depends on a chosen threshold α. A Z-score is "large enough" if its corresponding p-value (as defined previously) is below α.

Z-scores are computed as defined in Section 3.1, using available knockouts to compute the mean and standard deviation. Unless otherwise stated, we will assume $\alpha = 0.05$. Of course, this definition is not perfect as it depends on an arbitrary value α. Also, a gene could have an extreme value at some moment without it being caused by the knockout, although it should not be too frequent. Having several measures corresponding to a knockout of the same gene could help reduce this problem.

6.1.1 Evaluation

In order to evaluate the performance of our techniques on the prediction of some knockout, we thus need to have access to a measure corresponding to this knockout. We will obviously ignore this sample when doing the prediction and use it for validation only.

Similarly to the GRN inference problem, our methods will not directly return a list of affected genes but will instead associate a score to each gene. This score can then be used to rank genes from the most likely to be affected to the least likely. From there, we can compute the AUPR and the AUROC to evaluate the prediction for one knockout, without the need of a fixed threshold. When doing several predictions, we will compute the average AUPR and average AUROC, instead of aggregating the rankings. Indeed, we want to assess individual predictions so there is no need to aggregate rankings since that would imply other problems (these problems were already discussed in previous chapters).

6.1.2 Comparison with the GRN inference problem

Although the idea of both problems is to find a ranking of interactions in the end, there are many differences that show that using the same techniques as for network inference is not a perfect solution.

First, the goal of network inference is to find direct interactions, but many causal effects are not part of the gold standard. This can be explained by chain reactions. Some indirect links are thus to be found here while they were avoided before.

However, the problem seems more complex than that. We can perform some analysis on available datasets. Statistics for the DREAM4 dataset are displayed in Table 6.1. We computed the transitive closure of the gold standard (i.e. the supergraph where $i \to j$ occurs iff there is a path from i to j in the gold standard). Surprisingly, we observe that many causal effects are not part of the closure either, although we would expect that if there is an effect from one gene to another, then there is a corresponding chain of links in the graph. This problem could be due to α being too high (although some of these problematic links remain with very low values of α) or to noise. We will just ignore this problem in the rest of this chapter. However, this might indicate that our evaluation method is not perfect and needs improvements such as only keeping causal effects which are in the transitive closure.

We also find several "gold standard" interactions which are not causal effects according to the definition above. This might depend on the way true causal effects are computed (choice of the "normal" expression values, choice of the threshold), but it can also result from more complex behaviors of the network.

Finally, we also added statistics about SIEs. We can see that they are far less numerous than causal effects. An important part of them are also causal effects, although some of them are not. However, despite being more restrictive, the problem with the transitive closure is more important since a higher percentage of the causal effects is not in the closure. Using SIEs instead of our definition would thus not be a solution.

	Net1	Net2	Net3	Net4	Net5
Number of GS links	176	249	195	211	193
Number of GS closure links	639	667	2375	2316	953
Number of causal effects	252	236	255	300	355
| GS ∩ causal effects |	116	114	103	110	79
| GS closure ∩ causal effects |	217	165	218	258	280
| causal effects \ GS |	136	122	152	190	276
| causal effects \ GS closure |	35	71	37	42	75
| GS \ causal effects |	60	135	92	101	114
Number of SIE	57	57	62	61	65
| SIE ∩ causal effects |	50	53	51	57	58
| SIE ∩ GS |	35	37	40	38	25
| SIE ∩ GS closure |	44	43	50	47	49

Table 6.1: Summary of statistics about the different types of interactions in the DREAM4 dataset

6.2 Methods

Many methods can be imagined to solve this problem. Previous methods used for GRN inference are still interesting despite the differences mentioned before. Additionally, other more specific methods can be imagined. In this section, we will list these methods and discuss the new ones in detail.

6.2.1 Previous Z-score inference techniques

The idea of predicting the effect of "missing" knockouts sounds very similar to the problem we had in Chapter 5. Indeed, the idea was to find appropriate values for missing Z-scores. The problem was different since we mostly wanted the values to be adapted when used as weights for GENIE3. The "mean" technique, for example, would make no sense here. However, other methods were supposed to have some predictive power. These are the following, summarized (for full details see Section 5.2) :

- **Fictional KO** : Z-scores corresponding to a given (not available) knockout are computed by assuming that the knockout measure is the one where the gene to knock out is the lowest. Contrary to the previous chapter, no distribution correction is necessary since we evaluate predictions individually (no aggregation with other knockouts is done).
- **Co-occurence** : When two genes appear to both have exceptional values at the same moment, we increase the scores of the interaction in both direction. For this reason, this method has the big problem of not taking causality into account, which is an important part here. The same remark about distributions can be applied.
- **Petralia method** : This method tries to borrow information from available Z-scores to infer the other ones. Missing values are inferred by using a linear combination of available values.

In the previous chapter, the output of these methods was converted into weights using different techniques (BINARY, PVALUES, ...). In this case, this must not be done.

6.2.2 Weighted GENIE3

Similarly, the previously used weighted GENIE3 can be used directly for this problem, despite the differences. Again, the different weight creation methods (Z, P-VALUES, BINARY, UNILOW) can be tested. To infer the missing Z-scores before building weights, all methods could potentially be used too. In practice, we will focus on the most successful ones, based on our previous results.

Obviously, the particular case of GENIE3 with no weights (which corresponds to having uniform weights) can be used too. It is worth noting that GENIE3 has an opposite approach compared to what we do here. Indeed, GENIE3 tries to find the regulators for a particular gene, while here we would like to find the genes regulated by another gene. For one knockout to predict, every score will be provided by a different random forest, and the problem of comparability of scores between different forests still holds.

6.2.3 Prediction with forests of GENIE3 (`GENIE3-predict`)

GENIE3 uses Random Forests to learn from data, but then only feature importances are used to predict interaction scores. This makes sense in the case of GRN inference. However, in the current problem, we want to predict the effect of a knockout on gene expressions. Therefore, it might be useful to actually use the Random Forests to do predictions.

The way this will be done is the following. We use the same procedure as GENIE3, up to the creation of the Random Forests (included). Then, instead of computing importances, we use the forests to do predictions on artificially generated samples. Suppose that we want to predict the expression of gene j after a knockout of gene k. We must use the Random Forest predicting the expression of gene j and give him appropriate samples for prediction. We first generate a complete sample where each gene is assigned a random value drawn from a gaussian distribution (where the mean and the variance are the ones of the gene expressions in the dataset). The second sample is built by setting $X^k = 0$ in the first sample, in order to simulate the knockout. We then predict the value of X^j for each sample. The absolute difference $|X^j_{(1)} - X^j_{(2)}|$ is set as the score of interaction $k \to j$.

Since the result might depend a lot on the generated sample, we can repeat that with several samples for one interaction and average the scores at the end.

Weighted GENIE3 methods can be used as well for the first part of this technique. However, we will limit ourselves to the classic GENIE3 for experiments.

6.2.4 Indirect links

As mentioned before, a particularity of the KO prediction problem compared to GRN inference is that we do not want to keep only direct links. Some indirect interactions might be strong enough to appear as causal effects. Therefore, if a

method manages to find direct links correctly, it could be interesting to artificially increase some indirect links to improve results.

A directed graph can be described by its adjacency matrix G, such that $G(i,j) = 1$ if there is a direct link between genes i and j, and $G(i,j) = 0$ otherwise. An interesting property of this matrix is that its powers can be used to compute easily the number of walks from one gene to another. Indeed, the value of $G^k(i,j)$ indicates the number of directed walks of length k from i to j.

To compute the transitive closure of a graph, we must find all indirect links. This can be done by summing all the powers of G. For a graph with p genes :

$$\sum_{k=1}^{p} G^k(i,j) > 0$$

if and only if there is a path from i to j in the graph.

Another interesting operation is the matrix exponential

$$e^G(i,j) = \sum_{k=1}^{\infty} \frac{1}{k!} G^k(i,j)$$

It has the same property as the previous formula and can thus be used to compute the transitive closure. But the advantage of this operation lies in the coefficients that gives higher powers (and thus longer walks) a lower importance than shorter walks.

Although this formula makes the most sense when dealing with an adjacency matrix, it could potentially be useful with score matrices too. Indeed, the score matrix S gives some positive weight to every link in the graph. Let us define the weight of a walk in the graph as the product of the weights of all links used. If a link has a weight of zero (it does not exist) then the walk has a weight of zero. The value of the power $S^k(i,j)$ is then equal to the sum of the weights of all walks of length k going from i to j. The proof is given in Appendix B.3.1. From this property, the idea of using the exponential of the score matrix does not seem completely irrational, although some pre-processing on the scores would be probably necessary for that to work correctly. Since many methods are already sensitive to indirect links to some degree, this could also be useless.

In the following experiments, we will focus only on testing this exponential formula as a postprocessing of the scores given by other methods.

6.2.5 Invariant Causal Prediction

Meinshausen et al. [14] already worked on the same problem and proposed their own method, based on a more statistical approach. It relies on the possibility to have gene expressions coming from different experimental settings. The main idea is that conditional probabilities of gene expressions should remain invariant across different experiments.

Their method assumes that a response variable Y can be formulated as a linear combination of other variables $X = (X_1, ..., X_p)$:

$$Y^e = X^e \gamma^e + \epsilon^e,$$

where $e \in \mathcal{E}$ correspond to the experimental setting. We wish to find the true set S of parents of Y (the variables which have a causal effect on Y). For such a set, $\gamma(e)$ should be identical in all environments $e \in \mathcal{E}$. We can then estimate γ for all possible subsets S of the set of all variables and keep the subsets for which we have the invariance of γ and that can also not reject the hypothesis that the residual variance $V_S(e) = Var(Y - X_S\gamma_S(e))$ is constant in all environments. The final subset $\hat{S}$ is the intersection of all found subsets.

As the algorithm returns a set of variables instead of a score, it is repeated a given number of times with bootstrap samples. The score of a variable then corresponds to the number of times it was selected.

For more information and implementation details, see the authors' paper. In practice, we used a modified version of a script provided by the authors. Since we do not have access to different experimental settings, we will assume knockouts are in a different setting than time series data, although the assumptions will then probably not hold totally. Steady-state measures and time series being very different types of measures, observing invariance is also unlikely.

6.3 Experiments and results

6.3.1 DREAM4 dataset

We will again use the five networks of the DREAM4 dataset to assess the performance of the different methods. Since we want to predict unknown knockouts, we need again to remove some of them artificially. The way this will be done is by using a five-fold cross-validation. Formally, the procedure to test a particular method on one network is the following :

1. Divide knockout samples into five blocks B_i $(i = 1, ..., 5)$ of the same size.
2. For each i in $\{1, ..., 5\}$:
   (a) Training set = $X_{KO} \setminus B_i \cup X_{obs}$, the set of all samples except block B_i.
   (b) Testing set = B_i
   (c) Train the tested method using the training set.
   (d) Predict the effect of a knockout for each gene having a corresponding knockout in B_i. Evaluate the prediction individually (AUPR and AUROC).
3. After this five-fold cross-validation, average all the individual evaluations.

The tested methods are detailed in Table 6.2. In addition to the real methods described before, we also test the performance of the gold standard and its exponential to analyze the exponential idea in an "ideal" case.

Method	Precisions
GENIE3	Random Forests, T = 1000 trees, all data (time series + KO) • Classic • With exponential
GENIE3-GW (Mean and Fictional KO completion)	Random Forests, T = 1000 trees, all data (time series + KO) • BINARY ($\alpha = 0.05$, $\div = 100$) • P-VALUES ($\beta = 1$) • UNILOW ($\alpha = 0.05$, $k = 0.5$) • Z ($\beta = 1$ and $\beta = 3$)
GENIE3-predict	10, 50 and 200 predicted samples per interaction.
Fictional KO	
Co-occurence	Threshold $t = 1.5$ and $t = 1$
Petralia	
ICP	
Gold standard	• Normal • With exponential Remark : given the ranking problem (many scores with the same value), results are averages over several permutations

Table 6.2: Prediction methods tested for the DREAM4 dataset

AUPR results are displayed in Figure 6.1 for networks 1 and 2. Gold standard results are not shown to keep the figure readable. The complete numerical values can be found in Table B.3 in the appendices. Let us analyze the results. If we compare with the results we would obtain on average with random prediction (rightmost bar), it is clear that most methods have some predictive power. There are still two exceptions : the Petralia and the ICP methods. Concerning the Petralia method, although it sometimes gives results better than random (network 4), this is not significant enough. However, since the initial goal of this technique was not exactly to predict knockouts, this is not surprising. Concerning the ICP method, results are clearly random. As it was stated before, the assumptions of the methods (different experimental conditions) are not fulfilled which is probably the reason why it does not work.

The prediction of GENIE3, although far from perfect, is still significantly better than random. This seems logical, given the similarities between the GRN inference problem and this KO prediction problem that were explained previously. Contrary to the results of Chapter 4, using weights does not seem to improve significantly the performance. Albeit some bars are higher than the first one, the difference is small and some other bars are lower. Moreover, we cannot find a weighting method that would systematically increase the score. This confirms what we had already discovered in Section 5.3.1, that knowing Z-scores for some genes does not help GENIE3 in the prediction of the others.

Concerning the exponential, it seems that it slightly improves the performance of GENIE3 in most networks, except network 3. However, the increase or decrease is always small. With the gold standard, the interest of the operation is clearer as increases are more significant.

GENIE3-predict is clearly working too. Overall, using more samples seems to

improve the prediction. Compared to GENIE3, however, the method seems quite equivalent. Depending on the network, it will perform similarly, less well or better. With more samples, it is possible that the comparison would be more consistent.

Concerning the Fictional KO and the Co-occurence methods, we observe that the former is usually better than the latter. However, although they are significantly better than random too, they rarely outperform GENIE3.

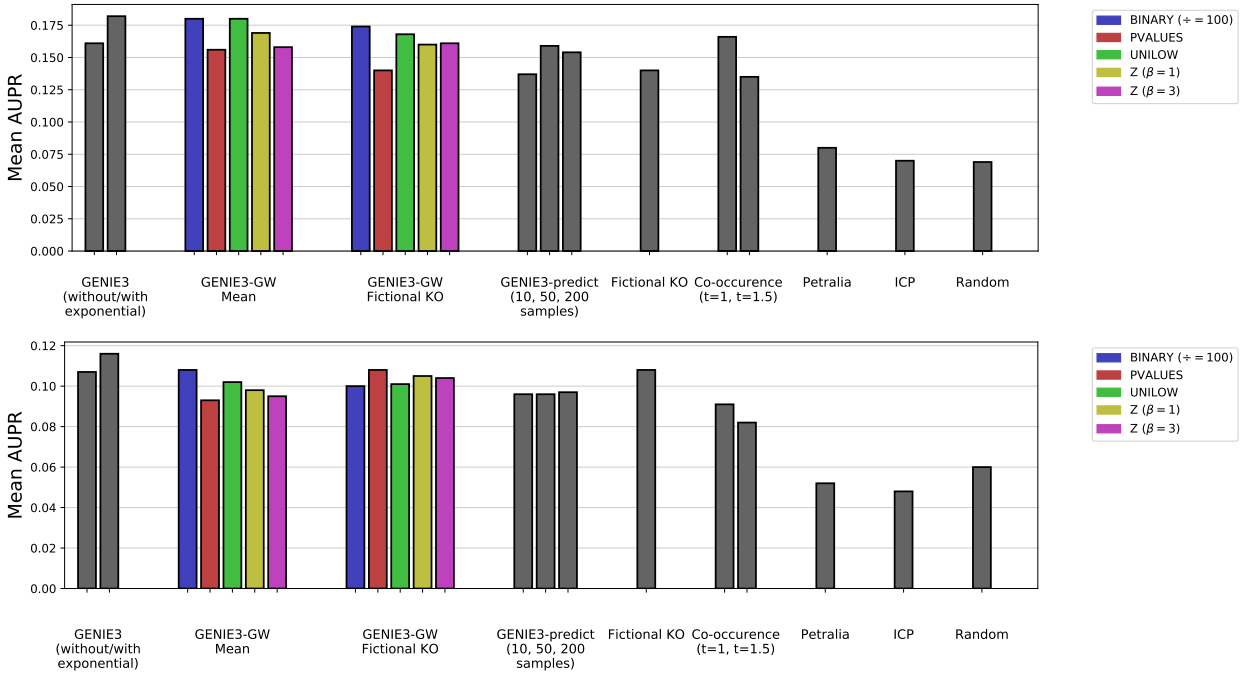


Figure 6.1: Mean AUPR results for networks 1 (top) and 2 (bottom).

6.3.2 Yeast dataset

Another interesting dataset for this problem is the yeast dataset, which was already described in Section 5.3.3, as it contains many individual knockouts. Out of the 6170 genes, 1479 of them have a knockout. 271 of these knocked out genes are also a transcription factor. While we could try to predict the 1479 knockouts, we will limit the prediction to the 271 transcription factors. Indeed, if a gene does not regulate any other gene, its knockout should in theory have no effect. However, as we discussed before, some of them are still usually found. To avoid at most this problem, it is thus preferable to limit the prediction to transcription factors. Despite that, only 1326 out of the 91351 causal effects to find are in the transitive closure of the gold standard, supporting even more the need for a better evaluation, as explained previously.

A five-fold cross-validation is used again. In one fold, one fifth of the 271 knocked out transcription factors is removed and the rest is used as the training set. There are thus $1479 - 271/5 \approx 1425$ knockout samples in the training set, in addition to the 262 wild type samples.

Given the size of the network, it was necessary to drastically reduce the number of tested methods, as the required time was much higher. The list of methods is

displayed in Table 6.3. Although the ICP method would probably have worked with this dataset, it was unfortunately impossible to test it because of the huge computational resources required and a lack of time.

Method	Precisions
GENIE3	Random Forests, T = 500 trees, all data (wild type + KO) • Classic • With exponential
GENIE3-GW (Mean completion)	Random Forests, T = 500 trees, all data (wild type + KO) • BINARY ($\alpha = 0.05$, $\div = 100$) • P-VALUES ($\beta = 1$) • Z ($\beta = 1$)
GENIE3-predict	10 predicted samples per interaction.
Fictional KO	
Co-occurence	Threshold $t = 1.5$ and $t = 1$
Petralia	
Gold standard	• Normal • With exponential

Table 6.3: Prediction methods tested for the yeast dataset

Results are given in Table 6.4. The scores are much worse than for the DREAM4 dataset. This time, we see that the gold standard and its exponential are not good at all. Most methods show a very similar AUPR around 0.06, which is barely better than random. Only the co-occurence method is slightly higher (0.072), although the AUROC seems to indicate that it is actually a bad method. The Petralia method is also the second highest (0.069), with a much higher AUROC (0.603). From these results, it thus seems to be the best choice. It is a completely different conclusion than for the previous experiment. However, given the very low scores, it is difficult to draw relevant conclusions.

6.4 Conclusion

In this chapter, we have shown the possibility to predict in part the effect of an unknown knockout. Despite similarities with the GRN inference problem, many differences were also highlighted, making the problem more interesting.

An evaluation method, based on Z-scores and a threshold, was decided. However, by analyzing the true network, it was observed that some effects to find are not supposed to exist. Whether this evaluation method is correct is still questionable and the question needs more analysis.

Several methods were tested, including methods already used in GRN inference, but also techniques specific to this problem (GENIE3-predict, ICP, matrix exponential). Their performance was assessed on two datasets : the DREAM4 dataset and another yeast dataset. The first dataset showed relatively good results while the second one had very bad scores.

It was difficult to distinguish a best method. It was clear that weighted GENIE3 is not really better than GENIE3 for this problem. The exponential seemed rather

Method	Mean AUPR	Mean AUROC
GENIE3	0.061	0.541
GENIE3 (exponential)	0.061	0.541
GENIE3-GW (mean) :		
BINARY	0.052	0.485
Z	0.058	0.534
PVALUES	0.059	0.524
GENIE3-predict :		
10 samples	0.061	0.544
Fictional KO	0.063	0.505
Co-occurence (t = 1)	0.072	0.502
Co-occurence (t = 1.5)	0.072	0.484
Petralia	0.069	0.603
Gold standard	0.062	0.507
Gold standard (exponential)	0.062	0.508
Random	0.056	0.500

Table 6.4: KO prediction results on the yeast dataset

beneficial although it was not really significant. The margin for improvement is thus still very important. The methods tested here were either too simple or not designed for this particular problem. It might be interesting to think of new methods better suited to this problem. Also, given the similarities between GRN inference and KO prediction, it is probable that improving one of them could be helpful to the other.

Chapter 7

Interest in no knockout situations

The purpose of this master thesis was to develop ways to use knockout information in order to improve current methods. To do so, several techniques have been designed as explained in Chapter 3. As stated before, the weighted GENIE3 techniques are not specific to knockouts and can actually be interesting in other cases. In this chapter, we show an example of such cases.

7.1 Weighted GENIE3 with no knockouts

Weighted GENIE3 techniques try to integrate a priori information using weights. While these weights were derived from knockouts in the previous chapters, this is only one particular case. Indeed, Petralia et al. [19] already showed that these weights could also be derived from other types of particular data such as time series.

Now, what if we do not have any particular data, i.e. no knockouts and time series, just unlabeled samples of gene expressions ? Actually, the weights matrix is just like any matrix of values given as output of all GRN inference methods. Thus, nothing prevents us from using this output as weights for the weighted GENIE3 techniques. The idea is thus to have two steps :

1. Use GENIE3 to infer a first scores matrix.
2. Use the scores matrix to build weights for a second iteration, using weighted GENIE3.

Actually, the first step could be replaced by any other method other than GENIE3. This could be interesting to analyze but we will focus on this case to see if using the (almost) same method two times can still be beneficial.

7.2 Tests

7.2.1 DREAM4 multifactorial tests

We are first going to test the idea on DREAM4 multifactorial dataset, which is another dataset of the DREAM4 challenge, different from the one used in previous chapters. It consists of five networks of 100 genes. For each network, 100 samples are provided. They represent steady states of gene expressions after multifactorial

perturbations. They can thus be seen as unlabeled (we have no information about knocked out genes).

We will follow the previous idea. We convert GENIE3 scores to weights using the two following methods :

- Direct method (with β) : weights are simply the GENIE3 scores, with a potential exponent β applied to influence the distribution :

$$w_{ij} = s_{ij}^{\beta}$$

- Binary method : as before, weights can have only two possible values, the ratio of them being $\div$. Since we can hardly define p-values from GENIE3 scores to apply a threshold, this will be done by giving the n highest scores a high weight, and the other ones a low weight.

Results for the local approach are given on Table 7.1 for several methods and parameters. Similar results are obtained with the global approach. We compare these results with the performance of the first GENIE3 iteration. Additionally, we compare with the GENIE3 method were parameter K (the number of variables selected randomly at each node of a tree) is equal to the number of genes p instead of $\sqrt{p}$. Indeed, previous results on this dataset showed it was better to keep all variables at each node.

	Net1		Net2		Net3		Net4		Net5	
	AUPR	AUROC	AUPR	AUROC	AUPR	AUROC	AUPR	AUROC	AUPR	AUROC
GENIE3 (K=$\sqrt{p}$)	0.149	0.746	0.155	0.726	0.223	0.774	0.198	0.790	0.194	0.798
GENIE3 (K=p)	0.174	0.747	0.148	0.695	0.261	0.769	0.243	0.797	0.226	0.770
Direct ($\beta = 0.3$)	0.163	0.751	0.157	0.732	0.233	0.779	0.225	0.799	0.213	0.804
Direct ($\beta = 1$)	0.175	0.756	0.167	0.729	0.258	0.793	0.247	0.814	0.239	0.802
Direct ($\beta = 3$)	0.178	0.751	0.157	0.730	0.270	0.787	0.246	0.807	0.249	0.804
Binary ($n = 200$)	0.168	0.736	0.149	0.713	0.271	0.785	0.262	0.819	0.233	0.780
Binary ($n = 500$)	0.167	0.733	0.144	0.710	0.268	0.793	0.238	0.799	0.241	0.773

Table 7.1: Results of two-step GENIE3 for DREAM4 multifactorial networks. For binary methods, $\div = 100$

It immediately appears that the results are improved quite a lot by adding the second iteration, except for a few cases. Using the direct method with $\beta = 1$ or $\beta = 3$ seems to be the best choice. The intuition is that the first iteration helps GENIE3 focus more quickly on the right variables, leading to better trees on average. Note that adding more similar steps was unsurprisingly unsuccessful, so we cannot improve results continuously by relaunching GENIE3 again with previous scores.

When looking at results for a single iteration of GENIE3 with $K = p$, we observe improvements of the same order for the AUPR. The direct method still manages to slightly outperform it most of the time. However, there is a big advantage in using the new method instead of using $K = p$. Indeed, GENIE3's complexity is $O(pTKNlogN)$. Therefore the algorithm complexity is not changed (constant factor 2) by using the two-steps approach while it is multiplied by $\sqrt{p}$ when taking $K = p$, compared to a single iteration of GENIE3 with $K = \sqrt{p}$. This makes the execution time much shorter.

It might be possible that the ranking of regulators, taken gene by gene, is actually not improved, but that the values yield a better aggregation at the end (and thus a better global ranking). To see if it is the case, we can compute the AUPR and the

AUROC for each gene individually and take the mean. These values are showed in Table 7.2 for the direct method with $\beta = 1$ and 3. The results confirm our intuition : no real improvement is made on mean AUPRs and mean AUROCs. It even tends to be worse, thus the improvement on the global ranking has to come from a better aggregation.

	Net1		Net2		Net3		Net4		Net5	
	AUPR	AUROC	AUPR	AUROC	AUPR	AUROC	AUPR	AUROC	AUPR	AUROC
GENIE3 (K=$\sqrt{p}$)	0.293	0.762	0.310	0.745	0.464	0.794	0.424	0.838	0.393	0.841
GENIE3 (K=p)	0.278	0.755	0.291	0.708	0.444	0.778	0.408	0.833	0.359	0.808
Direct ($\beta = 1$)	0.291	0.766	0.297	0.741	0.464	0.801	0.418	0.844	0.390	0.842
Direct ($\beta = 3$)	0.284	0.762	0.296	0.743	0.469	0.797	0.418	0.842	0.380	0.841

Table 7.2: Mean AUPR and AUROC for DREAM4 multifactorial results (local approach)

Since $K = p$ gives better initial results, we could wonder whether using this in the first or the second step (or both) would be beneficial or not, despite the time performance decrease. We give the results for the direct method with $\beta = 1$ in Table 7.3. Clearly, results show that it is not interesting as they stay more or less the same. This is not surprising giving the results of the previous paragraph.

	Net1		Net2		Net3		Net4		Net5	
	AUPR	AUROC	AUPR	AUROC	AUPR	AUROC	AUPR	AUROC	AUPR	AUROC
GENIE3 (K=$\sqrt{p}$)	0.149	0.746	0.155	0.726	0.223	0.774	0.198	0.790	0.194	0.798
GENIE3 (K=p)	0.174	0.747	0.148	0.695	0.261	0.769	0.243	0.797	0.226	0.770
$K_1 = \sqrt{p}$, $K_2 = \sqrt{p}$	0.175	0.756	0.167	0.729	0.258	0.793	0.247	0.814	0.239	0.802
$K_1 = p$, $K_2 = \sqrt{p}$	0.179	0.747	0.149	0.704	0.265	0.767	0.248	0.799	0.238	0.778
$K_1 = \sqrt{p}$, $K_2 = p$	0.175	0.747	0.149	0.699	0.263	0.764	0.243	0.805	0.227	0.772
$K_1 = p$, $K_2 = p$	0.175	0.744	0.147	0.687	0.260	0.761	0.242	0.791	0.223	0.767

Table 7.3: Results for the direct method with $\beta = 1$ with different values of K in the first (K_1) and second (K_2) iteration (local approach)

7.2.2 DREAM5 results

The same idea was tested for the DREAM5 dataset, using the global approach and direct weights ($w_{ij} = s_{ij}$). Although we know the knockouts, we will use them as any type of data here.

Method	**Net 1** AUPR / AUROC	**Net 3** AUPR / AUROC	**Net 4** AUPR / AUROC
GENIE3 (first step)	0.288 / 0.812	0.095 / 0.618	0.021 / 0.518
GENIE3 (second step)	0.382 / 0.831	0.071 / 0.612	0.020 / 0.518

Table 7.4: DREAM5 results

A huge improvement is observed for network 1, while a decreasing occurs for the others. This suggests that the initial iteration must be good enough to benefit the second one. Since results are very bad for networks 3 and 4, they can hardly help the second step.

Chapter 8

Conclusion

This master thesis analyzed gene regulatory networks, and in particular the effect of gene knockouts in such networks. In the first part, the use of knockout samples to improve GRN inference (especially GENIE3) was studied. The second part tried to determine whether it was possible to predict these knockouts.

For the GRN inference problem, two main methods served as the basis of the analysis : Z-scores, which work well with knockouts data, and GENIE3, which works with any kind of data but does not use the full potential of knockouts. From these observations, two versions of a weighted GENIE3 method were proposed, GENIE3-LW (local approach) and GENIE3-GW (global approach). Both ideas tried to influence GENIE3 using information provided by Z-scores, so as to combine the benefits of the two methods. Several ways of building weights from Z-scores were also proposed.

As a first step, Chapter 4 analyzed these new methods on an ideal case where a knockout of every gene is available. Results were very positive as the new methods outperformed both Z-scores and GENIE3 in most cases. It was also shown that using all data was better and that performances could be improved even further by multiplying again the output of GENIE3 by the Z-scores at the end. However, it was not really possible to distinguish a better approach (between local and global) or a better type of weights. Most variants analyzed at the end of the chapter were also not really interesting.

After that, Chapter 5 focused on more realistic cases, with much larger real networks (instead of artificial ones) where only a few knockouts were available. It was necessary to tackle the problem of missing Z-scores, since only the ones corresponding to the available knockouts could be computed. Several methods were proposed, some of them being rather complex, but in the end the simple Mean method showed the best results. On the DREAM5 and yeast datasets, the performance of weighted GENIE3 was sadly less encouraging. Even though it was sometimes better than GENIE3 or Z-scores alone, it never outperformed the simple product of them. It would thus be useful to determine when the method is interesting or not, or to find improvements that would make it work better in these real datasets.

For the KO prediction problem, developed in Chapter 6, several similarities with the GRN inference problem were explained but also many differences. It was thus necessary to develop another evaluation procedure. However, this way of evaluating might not be perfect and more reflection on this should be interesting. To actually predict knockouts, several methods used in the previous chapters were reused, but

a few others such as GENIE3-predict, ICP or the exponential were also introduced. Results showed the possibility to have some predictive power, but it stayed quite poor. No method really managed to beat GENIE3, although the exponential had some slightly positive effect. It was also clearly showed that knowing some knockouts does not particularly help to predict the others. Overall, the margin for improvement is still very important.

Finally, Chapter 7 showed the possibility of using the weighted GENIE3 techniques even when no knockouts are available. Indeed, using the output of GENIE3 as weights for weighted GENIE3 could help improving the performance. Although the same performance could be obtained by using systematically all genes at each node in the trees, our way had the advantage of being computationally faster. This chapter showed only one way of using weighted Random Forest differently, but it is likely that other cases could benefit from that.

In conclusion, we demonstrated that GENIE3 could actually be improved so as to use all the data in a better way. While it is still difficult to know when our new methods should be used or not, it is clear that they are useful and could probably be developed further. As for KO prediction, our first results are also quite encouraging and call for more developments.

Bibliography

[1] Dhammika Amaratunga, Javier Cabrera, and Yung-Seop Lee. Enriched random forests. Bioinformatics, 24(18):2010, 2008.

[2] R. Bonneau, D. J. Reiss, P. Shannon, M. Facciotti, L. Hood, N. S. Baliga, and V. Thorsson. The Inferelator: an algorithm for learning parsimonious regulatory networks from systems-biology data sets de novo. Genome Biol., 7(5):R36, 2006.

[3] A. J. Butte and I. S. Kohane. Mutual information relevance networks: functional genomic clustering using pairwise entropy measurements. Pac Symp Biocomput, pages 418–429, 2000.

[4] Jeremiah J Faith, Boris Hayete, Joshua T Thaden, Ilaria Mogno, Jamey Wierzbowski, Guillaume Cottarel, Simon Kasif, James J Collins, and Timothy S Gardner. Large-scale mapping and validation of escherichia coli transcriptional regulation from a compendium of expression profiles. PLOS Biology, 5(1):1–13, 01 2007.

[5] T. S. Gardner, D. di Bernardo, D. Lorenz, and J. J. Collins. Inferring genetic networks and identifying compound mode of action via expression profiling. Science, 301(5629):102–105, Jul 2003.

[6] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. Machine Learning, 63(1):3–42, 2006.

[7] Anne-Claire Haury, Fantine Mordelet, Paola Vera-Licona, and Jean-Philippe Vert. Tigress: Trustful inference of gene regulation using stability selection. BMC Systems Biology, 6(1):145, 2012.

[8] J. D. Hunter. Matplotlib: A 2d graphics environment. Computing In Science & Engineering, 9(3):90–95, 2007.

[9] Vân Anh Huynh-Thu. Machine learning-based feature ranking: Statistical interpretation and gene network inference. PhD thesis, Université de Liège, 01 2012.

[10] Vân Anh Huynh-Thu, Alexandre Irrthum, Louis Wehenkel, and Pierre Geurts. Inferring regulatory networks from expression data using tree-based methods. PLOS ONE, 5(9):1–10, 09 2010.

[11] Nicole Krämer, Juliane Schäfer, and Anne-Laure Boulesteix. Regularized estimation of large-scale gene association networks using graphical gaussian models. BMC Bioinformatics, 10(1):384, 2009.

[12] S. R. Maetschke, P. B. Madhamshettiwar, M. J. Davis, and M. A. Ragan. Supervised, semi-supervised and unsupervised inference of gene regulatory networks. Brief. Bioinformatics, 15(2):195–211, Mar 2014.

[13] A. A. Margolin, I. Nemenman, K. Basso, C. Wiggins, G. Stolovitzky, R. Dalla Favera, and A. Califano. ARACNE: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. BMC Bioinformatics, 7 Suppl 1:S7, Mar 2006.

[14] Nicolai Meinshausen, Alain Hauser, Joris M. Mooij, Jonas Peters, Philip Versteeg, and Peter Bühlmann. Methods for causal inference from gene perturbation experiments and validation. Proceedings of the National Academy of Sciences, 113(27):7361–7368, jul 2016.

[15] Patricia Menéndez, Yiannis A. I. Kourmpetis, Cajo J. F. ter Braak, and Fred A. van Eeuwijk. Gene regulatory networks from multifactorial perturbations using graphical lasso: Application to the dream4 challenge. PLOS ONE, 5(12):1–8, 12 2010.

[16] P. E. Meyer, K. Kontos, F. Lafitte, and G. Bontempi. Information-theoretic inference of large transcriptional regulatory networks. EURASIP J Bioinform Syst Biol, page 79879, 2007.

[17] Fantine Mordelet and Jean-Philippe Vert. Sirene: supervised inference of regulatory networks. Bioinformatics, 24(16):i76, 2008.

[18] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12:2825–2830, 2011.

[19] Francesca Petralia, Pei Wang, Jialiang Yang, and Zhidong Tu. Integrative random forest for gene regulatory network inference. Bioinformatics, 31(12):i197, 2015.

[20] Andrea Pinna, Nicola Soranzo, and Alberto de la Fuente. From knockouts to networks: Establishing direct cause-effect relationships through graph analysis. PLOS ONE, 5(10):1–8, 10 2010.

[21] Robert J. Prill, Daniel Marbach, Julio Saez-Rodriguez, Peter K. Sorger, Leonidas G. Alexopoulos, Xiaowei Xue, Neil D. Clarke, Gregoire Altan-Bonnet, and Gustavo Stolovitzky. Towards a rigorous assessment of systems biology models: The dream3 challenges. PLOS ONE, 5(2):1–18, 02 2010.

[22] M. Schrynemackers, L. Wehenkel, M. M. Babu, and P. Geurts. Classifying pairs with trees for supervised biological network inference. Mol Biosyst, 11(8):2116–2125, Aug 2015.

[23] G. Yona, W. Dirks, S. Rahman, and D. M. Lin. Effective similarity measures for expression profiles. Bioinformatics, 22(13):1616–1622, Jul 2006.

[24] Yu Zhang, Zhidong Deng, Hongshan Jiang, and Peifa Jia. Inferring Gene Regulatory Networks from Multiple Data Sources Via a Dynamic Bayesian Network with Structural EM, pages 204–214. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

Appendix A

Technical implementation details

The content of this master thesis focused mostly on the methods and their analysis but we did not give any explanation about how we implemented these methods. This short appendix will explain these personal implementation details.

A.1 Language and libraries

Most of the methods and tests were programmed in Python 2.7. The use of version 2.7 instead of 3.0 was chosen because of compatibility problems. All GENIE3 tests and variants were done starting from the initial Python implementation available in http://www.montefiore.ulg.ac.be/~huynh-thu/GENIE3.html. This implementation is based on the Random Forest implementation of the Scikit-learn library [18]. The SciPy and NumPy libraries were heavily used for data manipulation and mathematical functions. All bar charts were generated with the Matplotlib library [8].

A.1.1 Weighted Random Forests

The Weighted Random Forests, used for the GENIE3-LW and GENIE3-GW methods were implemented by modifying locally the source code of Scikit-learn's Random Forests implementation.

A.1.2 Evaluation

The computation of AUPR and AUROC values was done using custom functions. When computing the precision-recall curve, we assumed the curve started in (0, 1).

A.2 Computational resources

A small part of the tests were run on a personal laptop (8GB RAM, CPU Intel Core i5-5200U) under Linux. GENIE3 was parallelized in order to run on several threads in parallel (this could be done easily since all of the p Random Forests can run independently). However, most of the tests were very heavy and would not run on a simple laptop in less than a day. For such tests, we had access to the CÉCI clusters (http://www.ceci-hpc.be) which allows for much larger parallelization.

Appendix B

Additional figures and tables

B.1 Chapter 4 - Knockout integration I

B.1.1 Complete time-series results (figures)

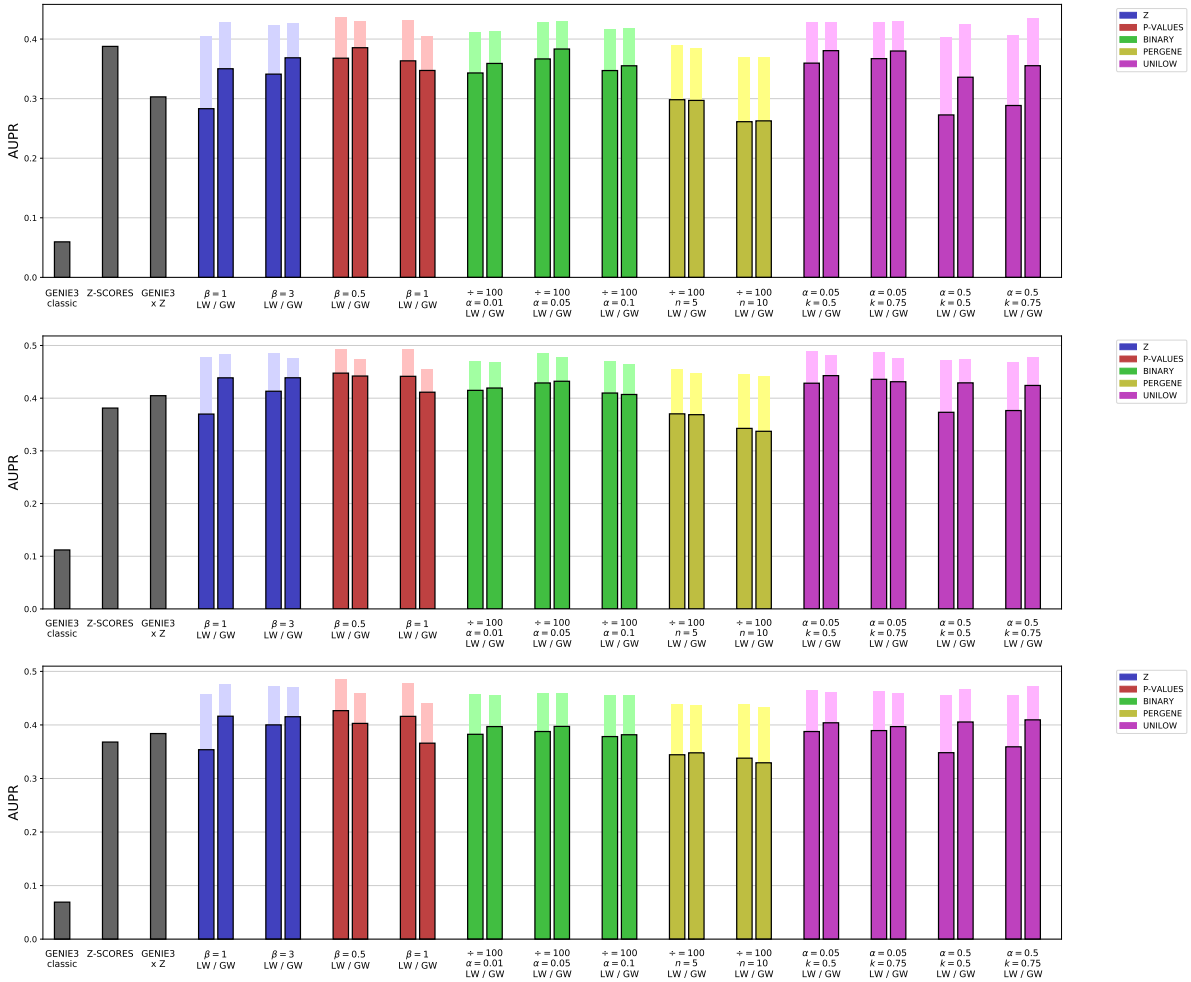


Figure B.1: AUPR results for Networks 2 to 4 of DREAM4 with GENIE3-LW (first bar) and GENIE3-GW (second bar). Lighter bars = product with Z-scores.

B.1.2 Complete time-series results (numerical values)

	AUPR				
Method	**Net 1**	**Net 2**	**Net 3**	**Net 4**	**Net 5**
GENIE3	0.063	0.06	0.112	0.069	0.081
GENIE3 x Z	0.497	0.303	0.405	0.384	0.249
Global approach					
BINARY ($\alpha = 0.01$, $\div = 100$)	0.53	0.359	0.419	0.397	0.253
x Z	0.591	0.413	0.468	0.456	0.277
BINARY ($\alpha = 0.05$, $\div = 100$)	0.573	0.383	0.432	0.397	0.273
x Z	0.615	0.43	0.477	0.46	0.3
BINARY ($\alpha = 0.1$, $\div = 100$)	0.546	0.355	0.407	0.382	0.228
x Z	0.606	0.419	0.464	0.455	0.279
PERGENE ($n = 5$, $\div = 100$)	0.422	0.297	0.369	0.348	0.196
x Z	0.561	0.384	0.447	0.437	0.257
PERGENE ($n = 10$, $\div = 100$)	0.364	0.263	0.337	0.329	0.171
x Z	0.541	0.369	0.441	0.433	0.243
PERGENE ($n = 20$, $\div = 100$)	0.307	0.208	0.253	0.283	0.166
x Z	0.516	0.358	0.402	0.422	0.236
P-VALUES ($\alpha = 0.5$)	0.523	0.385	0.442	0.403	0.294
x Z	0.578	0.43	0.473	0.46	0.301
P-VALUES ($\alpha = 1$)	0.472	0.347	0.411	0.366	0.274
x Z	0.548	0.405	0.454	0.441	0.293
UNILOW ($\alpha = 0.05$, $k = 0.5$)	0.569	0.381	0.443	0.404	0.279
x Z	0.613	0.428	0.481	0.461	0.308
UNILOW ($\alpha = 0.05$, $k = 0.75$)	0.569	0.38	0.431	0.397	0.276
x Z	0.613	0.43	0.475	0.459	0.302
UNILOW ($\alpha = 0.5$, $k = 0.5$)	0.504	0.336	0.429	0.406	0.267
x Z	0.59	0.425	0.473	0.467	0.28
UNILOW ($\alpha = 0.5$, $k = 0.75$)	0.514	0.355	0.424	0.409	0.274
x Z	0.597	0.436	0.477	0.473	0.289
Z ($\beta = 1$)	0.517	0.35	0.439	0.416	0.275
x Z	0.599	0.429	0.484	0.475	0.287
Z ($\beta = 3$)	0.541	0.369	0.439	0.415	0.271
x Z	0.6	0.426	0.475	0.47	0.295
Local approach					
BINARY ($\alpha = 0.01$, $\div = 100$)	0.515	0.343	0.415	0.383	0.252
x Z	0.589	0.411	0.471	0.458	0.278
BINARY ($\alpha = 0.05$, $\div = 100$)	0.555	0.367	0.429	0.388	0.279
x Z	0.612	0.429	0.484	0.46	0.304
BINARY ($\alpha = 0.1$, $\div = 100$)	0.538	0.347	0.41	0.378	0.236
x Z	0.606	0.416	0.471	0.456	0.285
PERGENE ($n = 5$, $\div = 100$)	0.426	0.298	0.37	0.344	0.206
x Z	0.57	0.389	0.454	0.439	0.264
PERGENE ($n = 10$, $\div = 100$)	0.367	0.261	0.343	0.338	0.185
x Z	0.545	0.37	0.445	0.438	0.253
PERGENE ($n = 20$, $\div = 100$)	0.299	0.209	0.264	0.279	0.169
x Z	0.513	0.355	0.407	0.421	0.246
P-VALUES ($\alpha = 0.5$)	0.544	0.368	0.448	0.427	0.291
x Z	0.612	0.436	0.493	0.486	0.301
P-VALUES ($\alpha = 1$)	0.536	0.363	0.441	0.416	0.272
x Z	0.617	0.431	0.492	0.477	0.298
UNILOW ($\alpha = 0.05$, $k = 0.5$)	0.535	0.36	0.428	0.388	0.286
x Z	0.615	0.428	0.489	0.465	0.311
UNILOW ($\alpha = 0.05$, $k = 0.75$)	0.555	0.367	0.436	0.389	0.286
x Z	0.615	0.428	0.486	0.463	0.308
UNILOW ($\alpha = 0.5$, $k = 0.5$)	0.437	0.273	0.373	0.348	0.245
x Z	0.583	0.403	0.472	0.456	0.291
UNILOW ($\alpha = 0.5$, $k = 0.75$)	0.452	0.288	0.377	0.359	0.242
x Z	0.582	0.406	0.468	0.455	0.286
Z ($\beta = 1$)	0.447	0.283	0.37	0.354	0.242
x Z	0.59	0.405	0.477	0.458	0.291
Z ($\beta = 3$)	0.497	0.341	0.413	0.4	0.255
x Z	0.605	0.424	0.486	0.472	0.293

Table B.1: Complete AUPR results of different methods on the networks of the DREAM4 challenge, using only time series as training data

B.1.3 Complete results for gold standard based weights

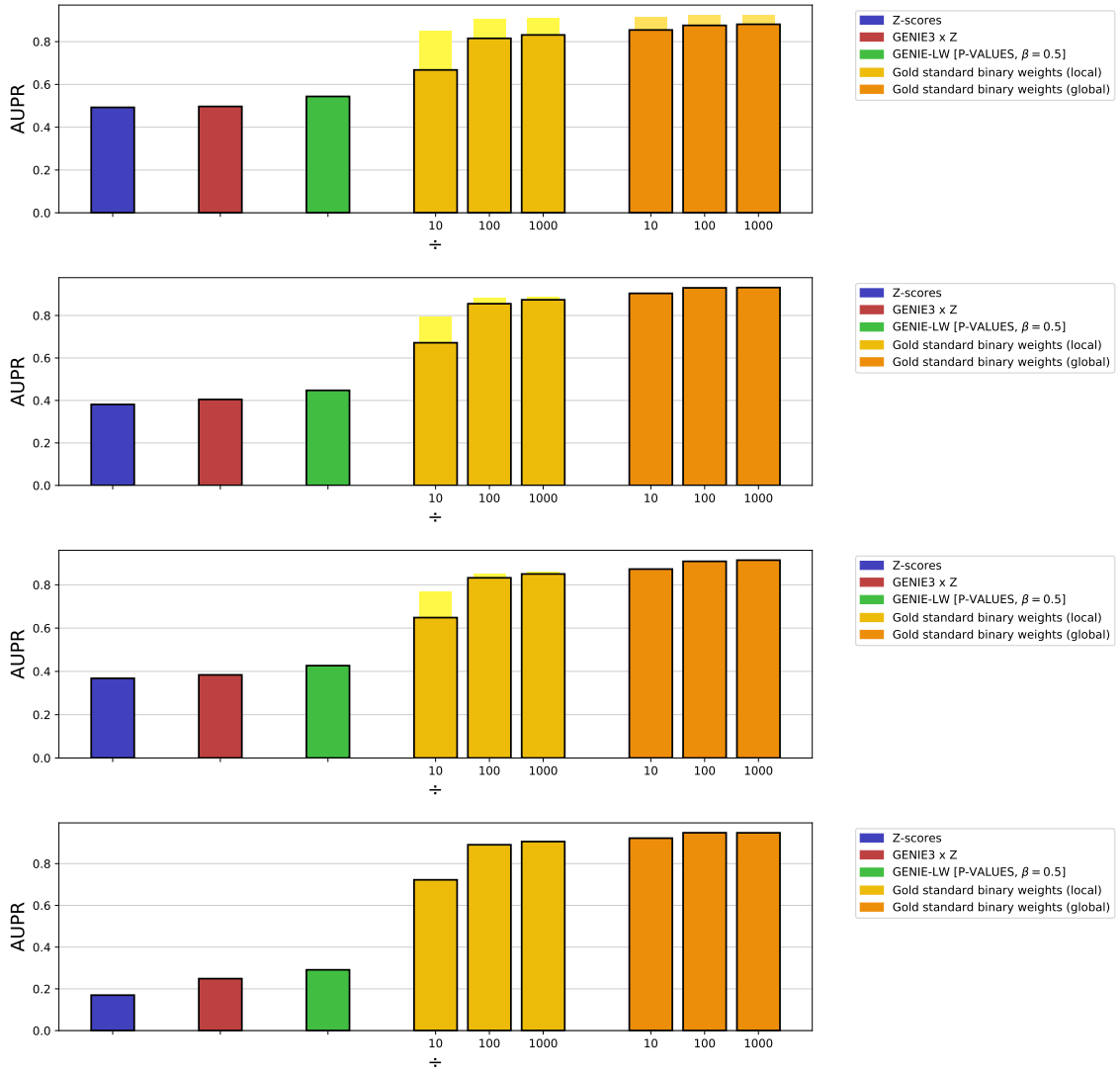


Figure B.2: AUPR results using weights built from gold standard (Networks 1, 3, 4 and 5 of the DREAM4 challenge)

B.1.4 Complete results for gold standard weights with false positives

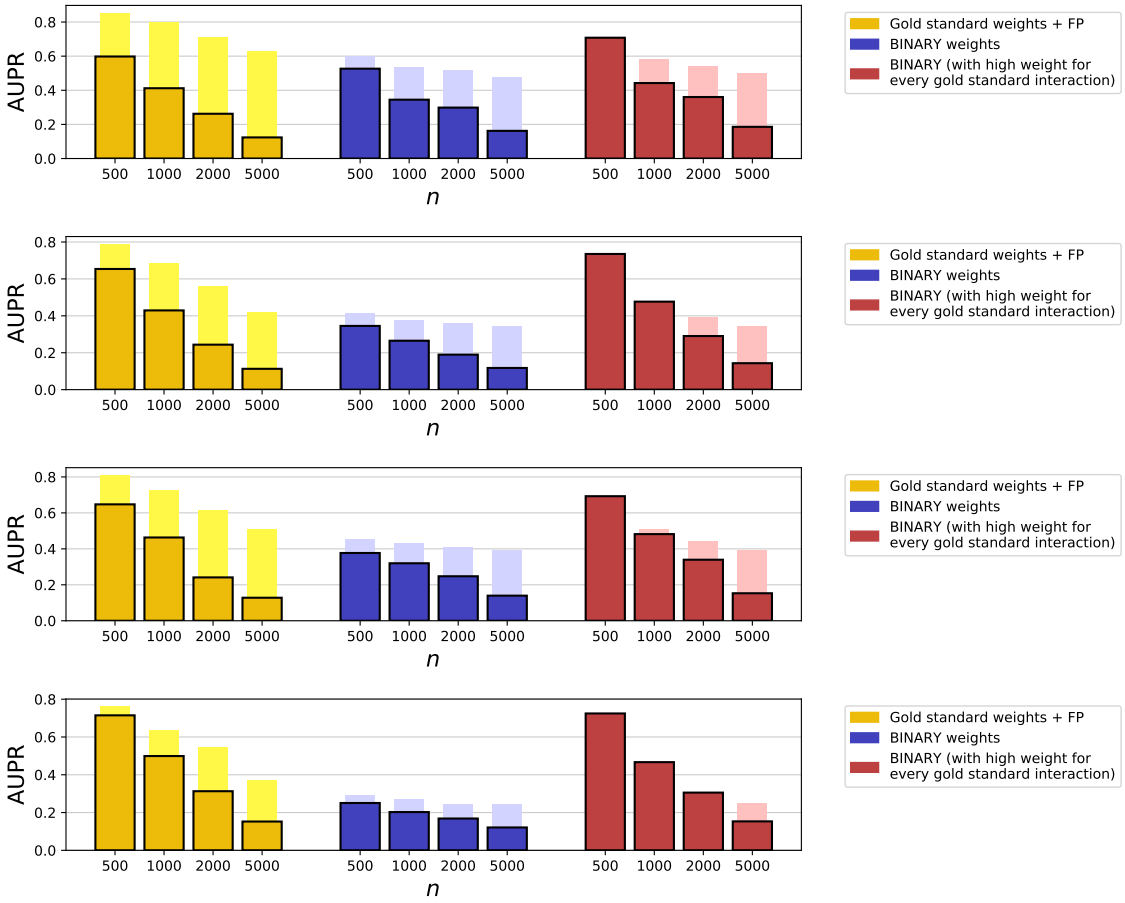


Figure B.3: AUPR results using weights built from gold standard with added false positives (global approach, Networks 1, 2, 4 and 5 of the DREAM4 challenge)

B.1.5 Complete results for the influence of the type of data

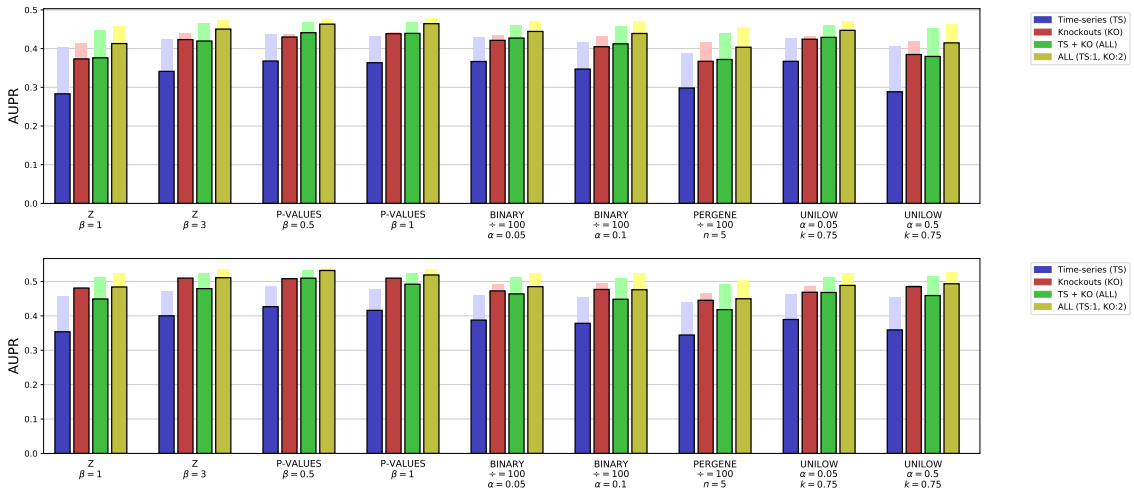


Figure B.4: Type of data. Local approach, networks 2 and 4, DREAM4 challenge

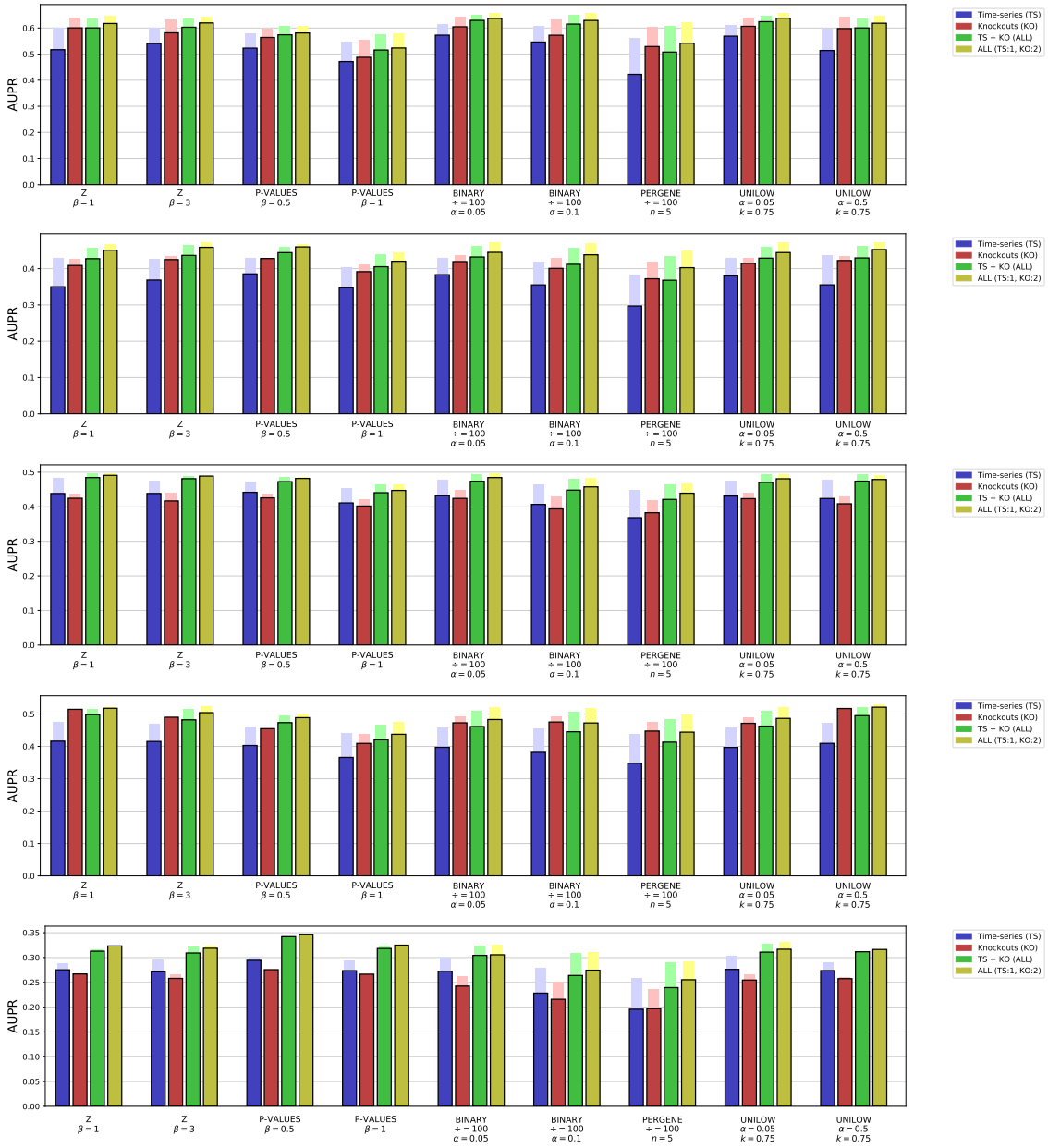


Figure B.5: Type of data. Global approach, networks 1 to 5, DREAM4 challenge

B.1.6 Complete results for the analysis of global variants

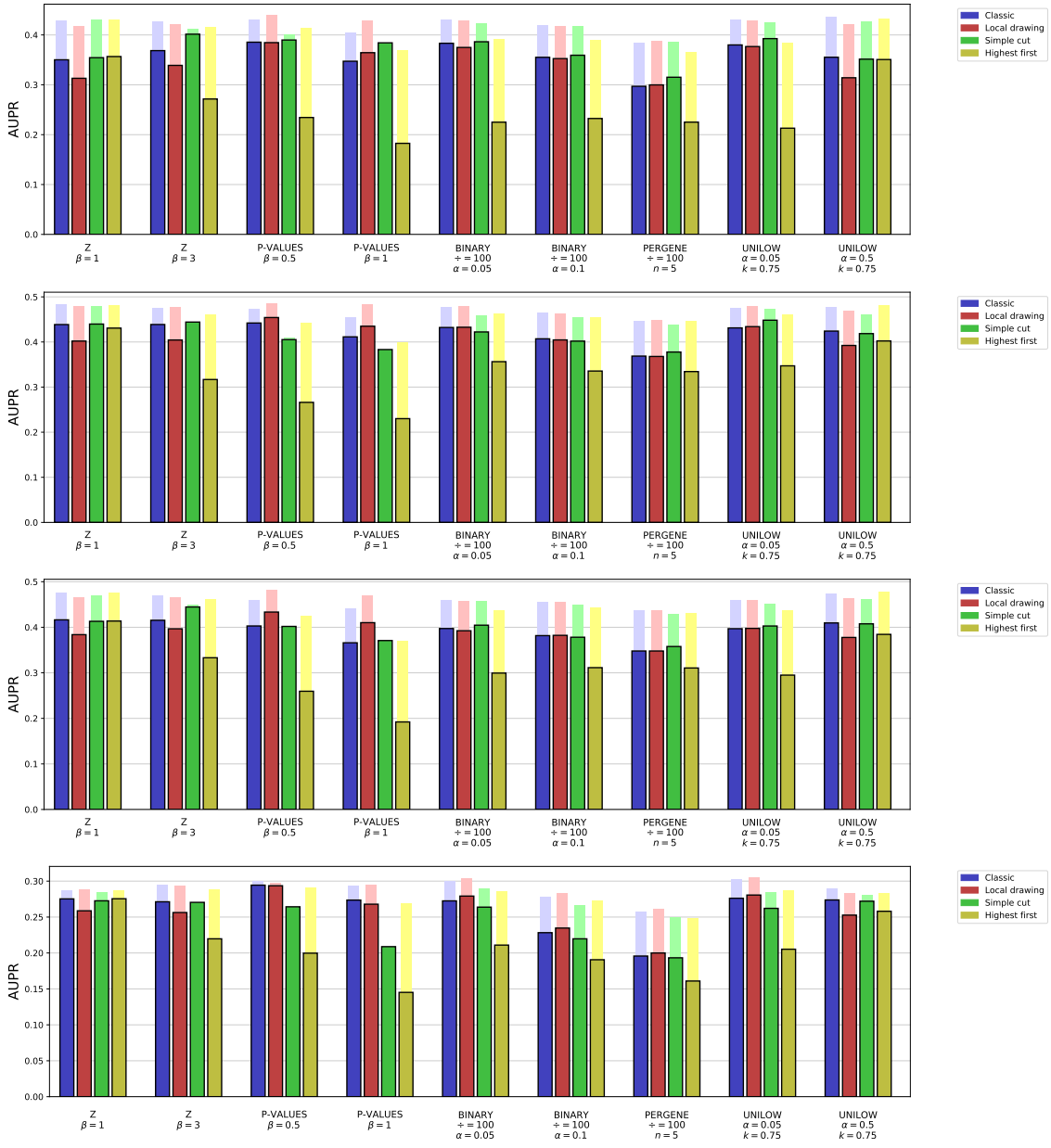


Figure B.6: Global variants. Networks 2 to 5, DREAM4 challenge.

B.1.7 Complete results for the filtered Z-scores method

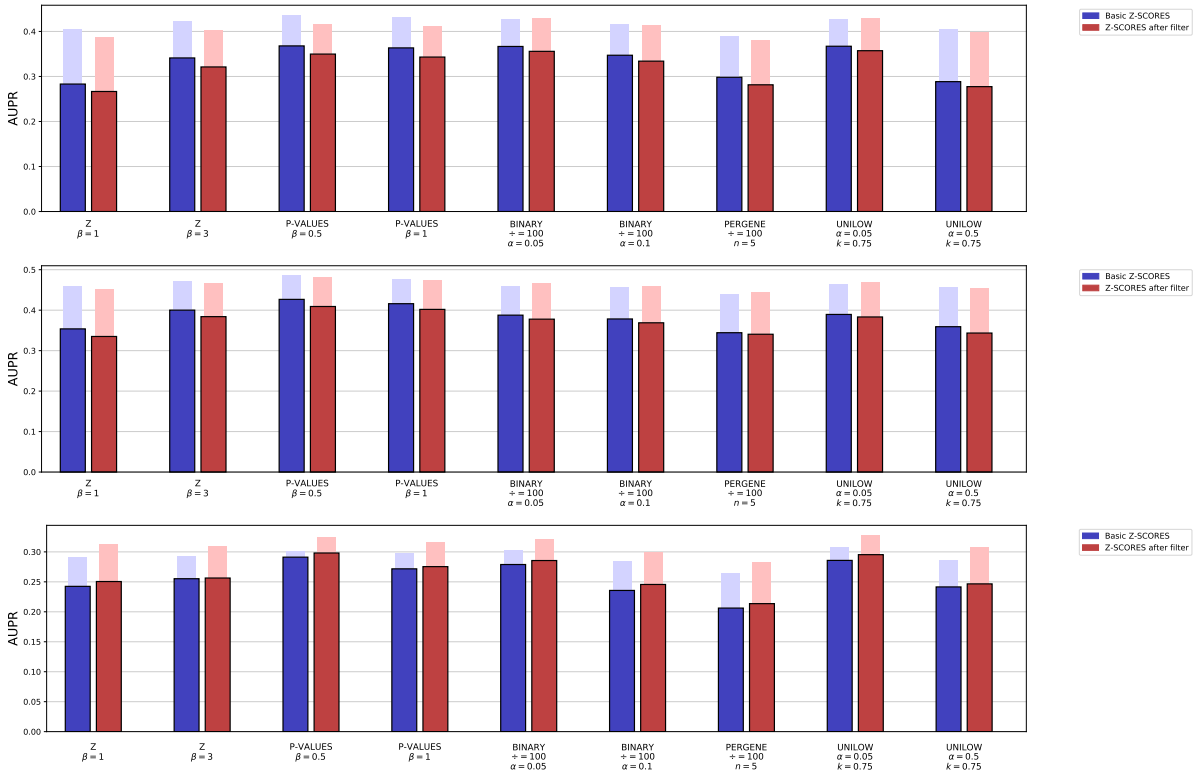


Figure B.7: Results with weights built from Pinna results. Networks 2, 4, 5 of the DREAM4 challenge.

B.1.8 Complete results for the "Best Z-scores as regulators" method

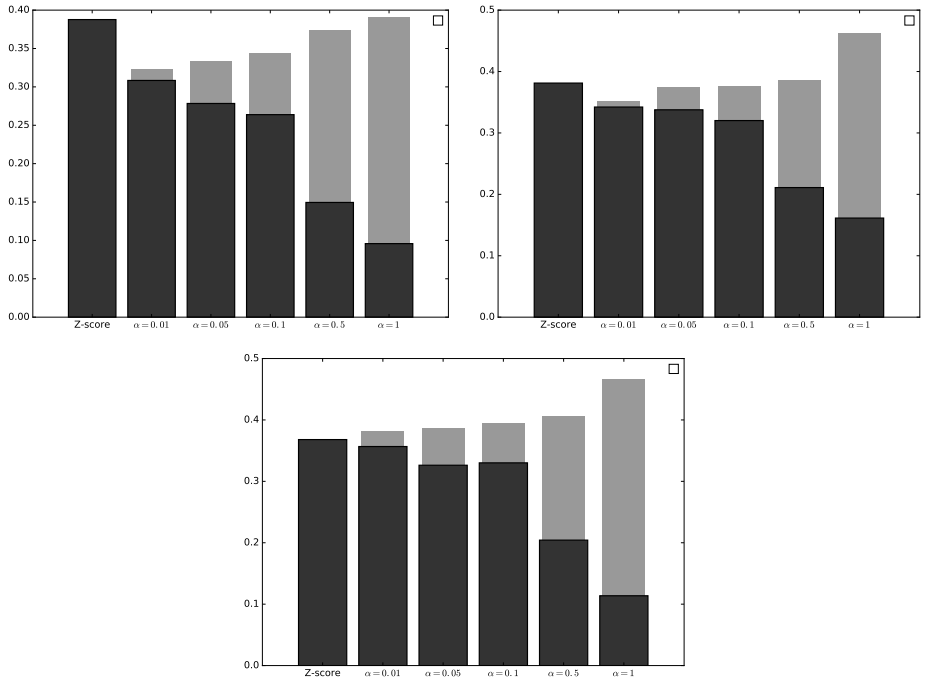


Figure B.8: AUPR for the "Best Z-scores as regulators" method. Networks 2, 3 and 4 of the DREAM4 challenge.

B.2 Chapter 5 - Knockout integration II

B.2.1 Complete results for the DREAM4 dataset experiments

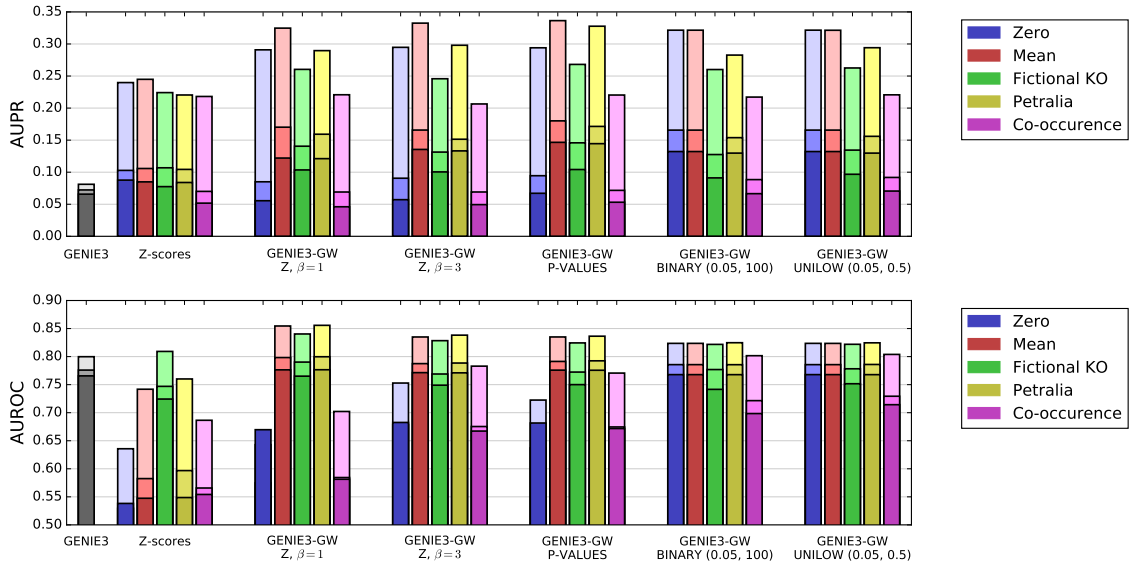


Figure B.9: AUPR and AUROC results for DREAM4 Net1. From darkest to lighest : 10, 20 and 50 available knockouts.

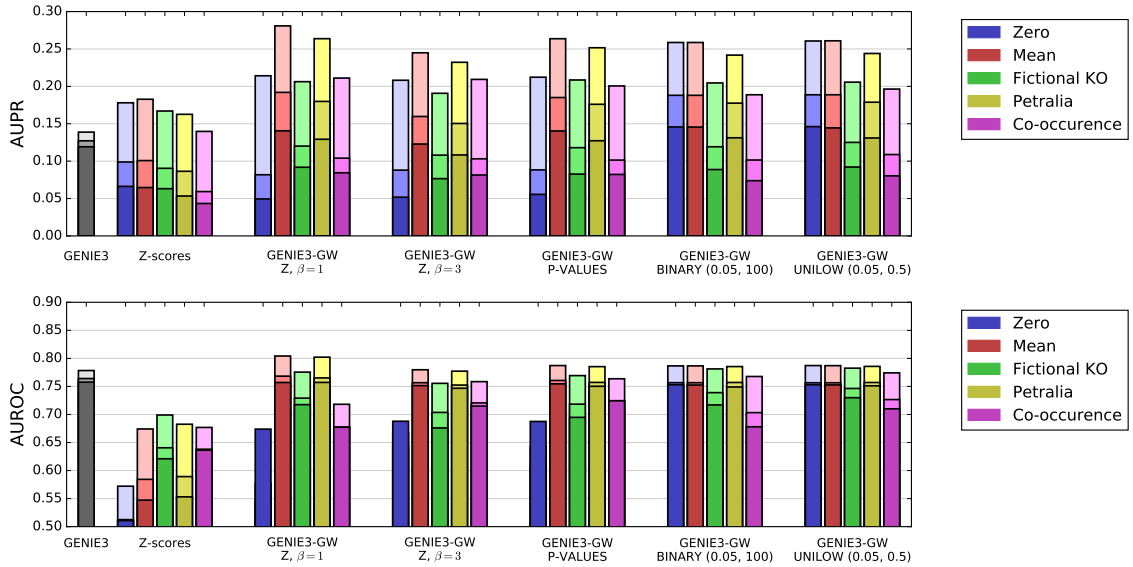


Figure B.10: AUPR and AUROC results for DREAM4 Net3. From darkest to lighest : 10, 20 and 50 available knockouts.

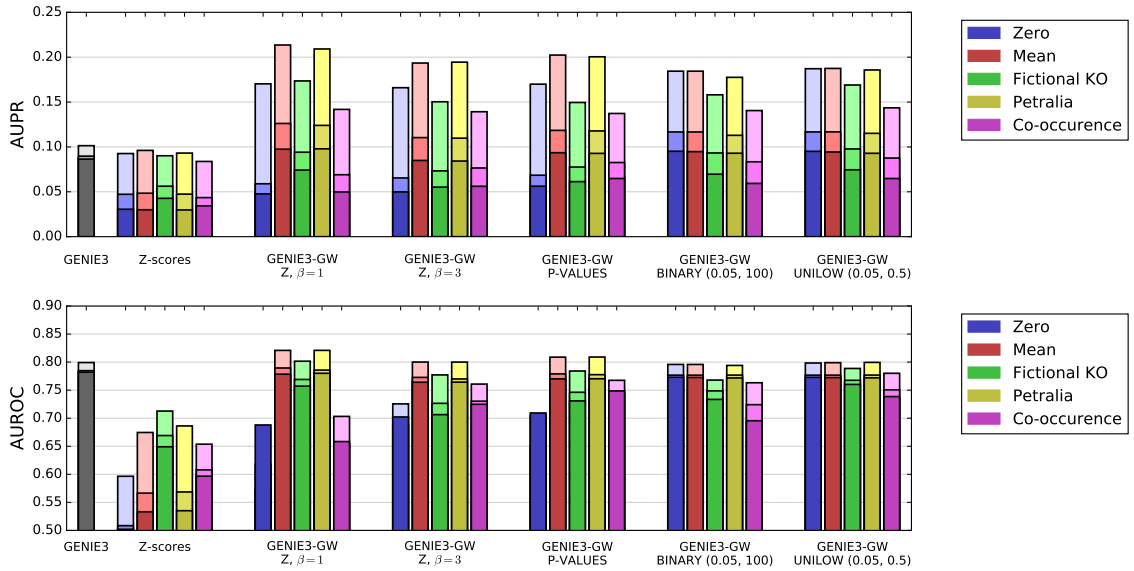


Figure B.11: AUPR and AUROC results for DREAM4 Net5. From darkest to lighest : 10, 20 and 50 available knockouts.

B.2.2 Complete results for DREAM4 with gold standard weights and Zero completion

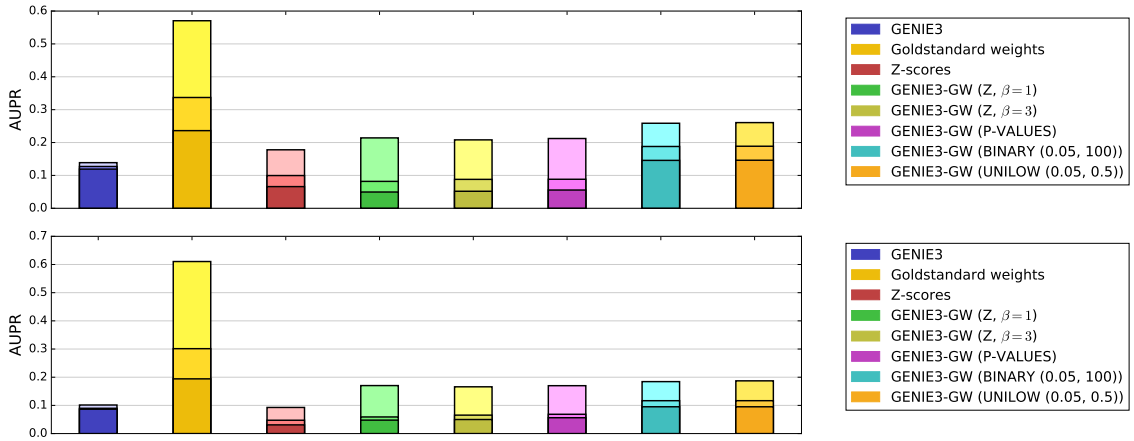


Figure B.12: Comparison of AUPR results between gold standard weights ($\div = 100$) and other methods (DREAM4 dataset, Network 3 and 5, Zero completion). From darkest to lighest : 10, 20 and 50 knockouts.

B.2.3 Results of the DREAM5 experiments with additional Z-scores product

	AUPR			AUROC		
Method	**Net1**	**Net3**	**Net4**	**Net1**	**Net3**	**Net4**
GENIE3	0.288	0.095	0.021	0.812	0.618	0.518
GENIE3 x Z (mean corrected)	0.319	0.093	0.020	0.815	0.621	0.515
GENIE3 x Z (fictional KO)	0.239	0.056	0.020	0.788	0.595	0.514
GENIE3-GW x Z [Mean] :						
BINARY ($\div = 100$, $\alpha = 0.05$)	0.230	0.048	0.019	0.807	0.615	0.513
P-VALUES ($\beta = 1$)	0.239	0.058	0.020	0.809	0.624	0.515
UNILOW ($\alpha = 0.05$, $k = 0.5$)	0.231	0.046	0.020	0.805	0.612	0.514
Z ($\beta = 1$)	0.223	0.057	0.020	0.794	0.615	0.512
Z ($\beta = 3$)	0.224	0.058	0.020	0.795	0.614	0.513
Z-SCORES (only)	0.077	0.018	0.018	0.599	0.521	0.501
GENIE3-GW x Z [Fict. KO] :						
BINARY ($\div = 100$, $\alpha = 0.05$)	0.189	0.028	0.019	0.732	0.574	0.512
P-VALUES ($\beta = 1$)	0.176	0.026	0.020	0.729	0.556	0.514
UNILOW ($\alpha = 0.05$, $k = 0.5$)	0.191	0.028	0.019	0.733	0.575	0.513
Z ($\beta = 1$)	0.196	0.033	0.020	0.734	0.573	0.514
Z ($\beta = 3$)	0.202	0.025	0.020	0.717	0.545	0.513
Z-SCORES (only)	0.036	0.015	0.017	0.577	0.504	0.500

Table B.2: DREAM5 results with additional Z-score product using scripts

B.3 Chapter 6 - Unknown knockouts prediction

B.3.1 Proof for the interpretation of the exponential of a weights matrix

Let S be a $p \times p$ matrix of weights, such that $S(i, j)$ represents the (positive) weight of interaction $i \to j$ in a graph. We define the weight of a walk in the graph as the product of the weights of all links used. We will prove that $S^n(i, j)$ is the sum of the weights of all walks of length n going from i to j, for all $n \in \mathbb{N}^+$. We can prove this property by induction.

Base case The property holds for $n = 1$. Indeed, the only walk of length 1 between i and j is the link between them itself. The weight of the walk is thus the value of $S^1(i, j)$.

Inductive step Assume the property holds for $S^n(i, j)$, then it must also hold for $S^{n+1}(i, j)$. By definition,

$$S^{n+1}(i, j) = \sum_{k=1}^{p} S(i, k) S^n(k, j)$$

Given the interpretation of $S^n(k,j)$, then $S(i,k)S^n(k,j)$ is the sum of weights of all walks of length $n+1$ going from i to k (initially, in one step) and then from k to j in n steps. Since the only way to go from i to j in $n+1$ steps is to first go to k and then go from k to j in n steps, and as we sum over every possible k, the property is true.

B.3.2 Complete AUPR and AUROC numerical values

	Mean AUPR					**Mean AUROC**				
	Net1	Net2	Net3	Net4	Net5	Net1	Net2	Net3	Net4	Net5
GENIE3	0.161	0.107	0.158	0.160	0.147	0.628	0.590	0.586	0.630	0.550
GENIE3 (exponential)	0.182	0.116	0.147	0.166	0.154	0.641	0.597	0.591	0.652	0.551
GENIE3-GW										
Mean :										
BINARY ($\div = 100$)	0.180	0.108	0.158	0.145	0.162	0.657	0.585	0.611	0.615	0.602
PVALUES	0.156	0.093	0.144	0.136	0.147	0.597	0.565	0.586	0.611	0.570
UNILOW	0.180	0.102	0.160	0.157	0.163	0.657	0.586	0.614	0.639	0.612
Z ($\beta = 1$)	0.169	0.098	0.167	0.146	0.161	0.656	0.574	0.608	0.632	0.570
Z ($\beta = 3$)	0.158	0.095	0.164	0.163	0.153	0.635	0.585	0.605	0.656	0.568
Fict. KO :										
BINARY ($\div = 100$)	0.174	0.100	0.161	0.129	0.138	0.632	0.590	0.628	0.636	0.578
PVALUES	0.140	0.108	0.130	0.169	0.141	0.557	0.529	0.574	0.594	0.564
UNILOW	0.168	0.101	0.150	0.135	0.141	0.623	0.583	0.623	0.644	0.588
Z ($\beta = 1$)	0.160	0.105	0.141	0.168	0.150	0.637	0.550	0.626	0.650	0.570
Z ($\beta = 3$)	0.161	0.104	0.140	0.165	0.151	0.645	0.555	0.624	0.661	0.561
GENIE3-predict :										
10 samples	0.137	0.096	0.101	0.142	0.156	0.576	0.593	0.513	0.636	0.537
50 samples	0.159	0.096	0.124	0.156	0.156	0.608	0.578	0.597	0.618	0.560
200 samples	0.154	0.097	0.109	0.178	0.147	0.641	0.546	0.585	0.643	0.554
Fictional KO	0.140	0.108	0.115	0.130	0.125	0.629	0.531	0.590	0.632	0.532
Co-occurence (t=1)	0.166	0.091	0.140	0.154	0.124	0.616	0.586	0.588	0.581	0.562
Co-occurence (t=1.5)	0.135	0.082	0.108	0.134	0.111	0.631	0.573	0.561	0.575	0.553
Petralia	0.080	0.052	0.063	0.105	0.079	0.549	0.511	0.567	0.553	0.530
ICP	0.070	0.048	0.070	0.060	0.087	0.549	0.525	0.478	0.486	0.507
Gold standard	0.399	0.240	0.299	0.274	0.190	0.698	0.628	0.662	0.662	0.588
Gold standard (exponential)	0.439	0.237	0.318	0.294	0.211	0.742	0.652	0.712	0.716	0.627
Random	0.069	0.060	0.062	0.072	0.078	0.5	0.5	0.5	0.5	0.5

Table B.3: Average prediction scores of different methods on the DREAM4 dataset