Study of Gene Regulatory Networks Inference Methods from Gene Expression Data

Pau Bellot

Advisor: Prof. Philippe Salembier

Prof. Patrick E. Meyer

Signal Theory and Communications Department

Universitat Politècnica de Catalunya

This dissertation is submitted for the degree of

Doctor of Philosophy

March 2017

I have not failed. I've just found 10,000 ways that won't work.

– Thomas A. Edison

"Begin at the beginning," the King said, gravely, "and go on till you come to an end; then stop."

– Lewis Carroll, *Alice in Wonderland*

Abstract

A cell is a self-sustaining protein-based robot that regulates itself. The cell eats to stay alive, it grows and develops; reacting to the environment, while subjected to evolution. It also makes copies of itself. These processes are governed by chain of chemical reactions, creating a complex system. The scientific community has proposed to model the whole process with Gene Regulatory Networks (GRN). The understanding of these networks allows gaining a systems-level acknowledgement of biological organisms and also to genetically related diseases.

This thesis focused on network inference from gene expression data, will contribute to this field of knowledge by studying different techniques that allows a better reconstruction of GRN. Gene expression datasets, are characterised by having thousands of noisy variables measured only with tens of samples. Moreover, these variables presents non-linear dependencies between them. Therefore, recovering a model that is capable of capturing the relationships contained in this data, constitutes a major challenge.

The main contribution of this thesis is a set of fair and sound studies of different GRN inference methods and post-processing algorithms. First, we present a novel approach for inferring gene networks and we compare it with other methods. It is inspired by the concept of "variable importance" in feature selection. However, many algorithms can be proposed to infer GRNs, so there is a need to assess the quality of these algorithms. Secondly, and motivated by the fact that the previous comparison was not informative enough, we introduce a new framework for in silico performance assessment of GRN inference methods. This work has led to an open source R/Bioconductor package called NetBenchmark. Finally, and thanks to this tool we have corroborated that inferring gene regulatory networks from expression data is a tough problem. The different algorithms have some particular biases and strengths, and none of them is the best across all types of data and datasets. Therefore, we present a framework for evaluating and standardising network consensus methods to aggregate various network inferences.

Acknowledgements

I would like to acknowledge and thank my parents, my sister, and specially my grandparents for the support and love they provided me during all these years. And most recently, to Gemma for being there, no matter what.

I would like to express my gratitude to my advisor at UPC, Prof. Philippe Salembier, for giving me the opportunity to contribute and love science. I would also like to express my sincerest gratitude to Prof. Patrick E. Meyer for granting me and opening to collaborate with him. This fruitful collaboration turned him my thesis advisor. To both of them, I would like to acknowledge all their time, advice and effort that they have given to my thesis, specially in the correction of this document.

I would also like to thanks Prof. Albert Oliveras-Vergés for among other things, opening the door at Gensips project. This experience introduced me to the topic of the thesis. Even though is not an official advisor he has help me as if he was.

A very special thanks goes to my colleagues and also friends at D5-120, and to my friends from Barcelona and Alicante.

Last but not least, I would like to thank the members of the jury who have accepted to review this work, namely Professors Pierre Geurts (University of Liège (ULg), Belgium), Frank Emmert-Streib (Tampere University of Technology (TUT), Finland) and Veronica Vilaplana (Universitat Politècnica de Catalunya (UPC)).

I also acknowledge that this thesis would not have been possible without the financial assistance of the Image Processing Group (GPI) at UPC and the Spanish Government through the FPU grant.

Table of contents

List of figures

List of tables

Glossary

AUPR Area under precision-recall curve.

AUROC Area under receiver operating characteristic curve.

ChIP Chromatin Immunoprecipitation (ChIP), a technique used to investigate the interaction between proteins and DNA.

DNA Deoxyribonucleic acid, a molecule that carries the genetic instructions.

DREAM Dialogue on Reverse Engineering Assessments and Methods.

FPR False positive rate.

GA genetic algorithm (GA).

GDD Graphlet degree distribution.

GDDA Graphlet degree distribution agreement.

GNW GeneNetWeaver.

GRN Gene Regulatory Networks.

GS Gold standard, a manually annotated training set or test set.

mRNA messenger Ribonucleic acid (RNA).

PR Precision vs. recall.

ROC Receiver operating characteristic.

TF Transcription Factor, a sequence-specific DNA-binding factor.

TG Target Gene.

TPR True positive rate.

Chapter 1

Introduction

1.1 Motivation

Every living thing on this planet is made of cells. A cell is a protein-based robot which regulates itself and maintains a constant state. To do so, in every cell, the biological machinery uses the instructions encoded in its genome. The genome contains the instructions encoded in a particular language, in the form of DNA, which we do not fully understand yet.

One of the major open question is to know how the elements of the DNA are organised. We would like to understand all the regulatory elements of the genome to be able to predict a particular circuit that responds to a specific stimulation, provokes a nutrient deprivation, generates specific compounds or is responsible for a particular disease, among other applications.

Being able to model a cell would allow us to design new therapies, even to create or change organisms for new purposes. For example, we could manipulate microbes or algae to generate biofuel or produce other useful goods. From a biomedical perspective, it opens the understanding of a disease's causes and not only its symptoms. It enables to design a suitable drug targeting a particular disorder and avoiding side effects of nowadays's drugs.

This goal could be achieved through reverse engineering processes which provide us with analogies and formal mathematical tools for reasoning about such systems. This thesis is focusing on network inference and will contribute to this area of knowledge with a study of different techniques to be able to achieve a better reconstruction of the models called Gene Regulatory Networks (GRN) from different gene expression data.

1.1.1 Reverse engineering of GRNs

Technologies such as microarrays or RNA-seq, allows us to collect gene expression of several genes at the same time in different experimental conditions. With this data, we can study how gene expression varies and therefore try to derive the interactions responsible for the observed data. Reverse engineering operates with this gene expression data to infer the regulatory interactions among genes using computational algorithms. In the last decade, many different approaches have been proposed to solve this Gene Regulatory Networks (GRN) reverse engineering problem based on collections of gene expression data.

Figure 1.1 illustrates the principle of reverse engineering of gene networks. As it has been said, there are several technologies available to measure gene expression levels like messenger RNA (mRNA) or protein concentrations. With such technologies it is possible to measure under different conditions a set of genes that conforms an unknown gene network called the target network in Figure 1.1 a). Once we performed different types of perturbations or experimental conditions to the network, it is possible to collect the gene expression data Figure 1.1 b). With this data as input and with some assumptions on the network's model, the inference method Figure 1.1 c) predicts one network as shown in Figure 1.1 d) that captures the relationships in the gene expression data. Finally, the predicted gene network should be validated with the target network Figure 1.1 e).

Two types of gene expression data are often used: time series and steady state. Time series data shows the dynamics of the genes' expressions. This kind of data, intuitively, may contain useful information but it is harder to obtain. In the other hand, the steady state data shows the state of genes' expressions at a time where it is assumed to be in equilibrium. Both data are typically measured after applying different perturbations or with different experimental setups to the network. Different inference methods have been proposed to deal with various types of available gene expression data. This thesis will only focus on algorithms of reverse engineering that works with steady state data.

Since many algorithms have been proposed, there is a need to assess the quality of these algorithms. If the target network is unknown, the inferred network is used to make predictions which should be validated with additional biological experiments, which is a slow and expensive process. However, this procedure is quite time-consuming and costly to evaluate the quality of an algorithm and compare a GRN inference method to other ones. This fact motivates the need for data, where the complete true underlying network is known, also called Gold standard (GS) or Ground Truth. Some real datasets may have such Gold Standard, but being sure if this underlying network is 100% correct is tricky. Also, in these datasets some external variables like noise are not easily controlled. These two facts motivate

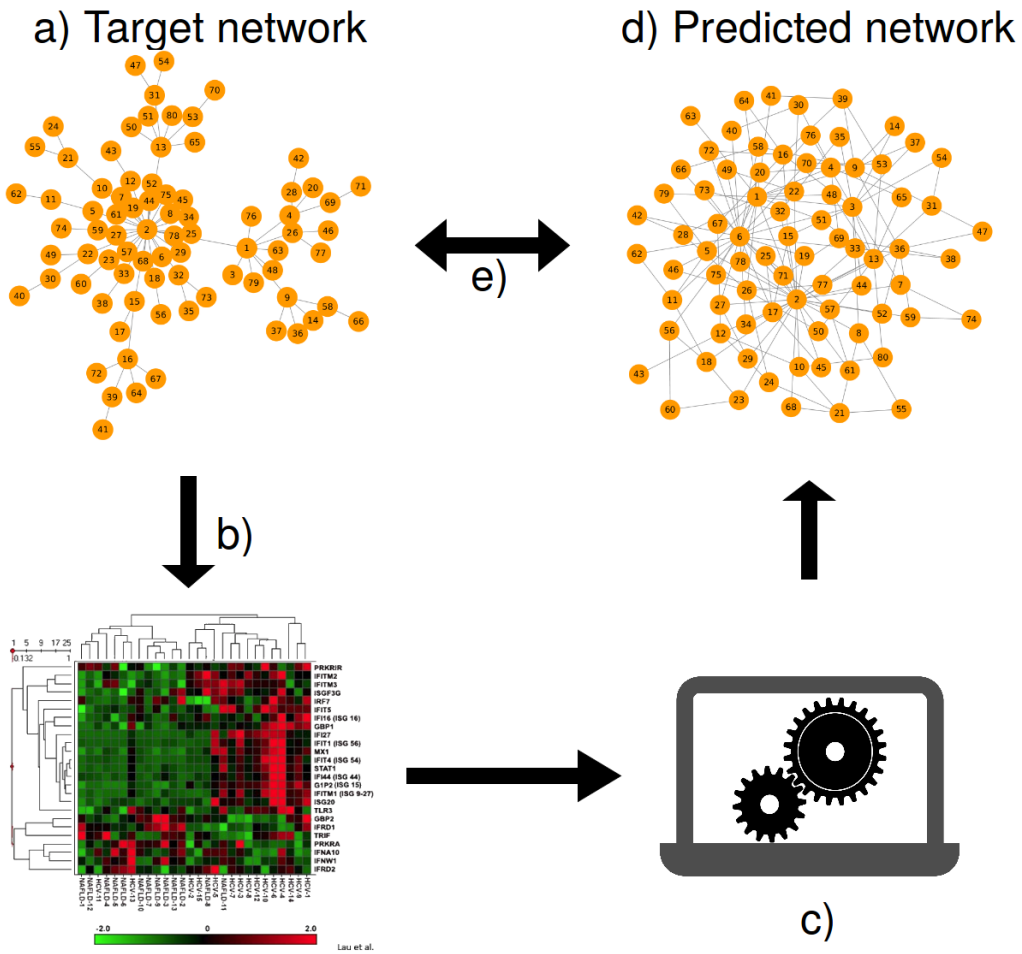


Fig. 1.1 Reverse engineering of gene networks. a) Unknown gene regulatory network (target network). b) Gene expression levels are measured after applying different types of perturbations to the network. c) The GRN inference method predicts one network with the input gene expression data. e) The predicted gene network is validated. Figure inspired by (Marbach, 2009)

the use of in silico data, which is an expression used in biology to mean "performed on a computer or via computer simulation." This approach enables one to check the performance of the algorithms against an entirely known ground truth (simulated networks in the computer model). However, it opens the question of how to generate this data and how to evaluate the predictions. We will discuss those issues in more detail in the Chapters 2 and 4.

1.2 Contributions

The main contribution of this thesis is a set of fair and sound studies of different GRN inference methods and post-processing algorithms. First, we present a novel approach for inferring gene networks and compare it with other methods. Secondly, and motivated by the fact that this comparison is not informative enough, we introduce a new framework for in silico performance assessment of GRN inference methods. Finally, we present a framework for evaluating and standardising network consensus methods. Specifically, the main contributions of this thesis are the following below.

1.2.1 GRN inference method

We present a new approach for reverse engineering gene regulatory networks, inspired by the concept of "variable importance". Research on VIGR (Variable Importance on Gene Relationships) and systematic comparison with another state-of-the-art techniques will also be described in this thesis using the state-of-the-art framework for evaluating them.

1.2.2 GRN performance evaluation: NetBenchmark

In the last decade, a significant number of methods for reconstructing gene regulatory networks from expression data have been proposed. However, very few tools and datasets allow evaluating accurately and reproducibly those methods. Hence, we propose here a new framework, able to perform a systematic, yet fully reproducible, evaluation of transcriptional network inference methods. This work has led to an open source R/Bioconductor package called NetBenchmark (Bellot et al., 2015a) that will also be described in this work.

1.2.3 Consensus networks analysis

Inferring gene regulatory networks from expression data is a thorny problem that has raised the interest of the scientific community. Different algorithms have been proposed to try to solve this issue. All different methods have some particular biases and strengths, and none of them is the best across all types of data and datasets. As a result, the idea of aggregating various network inferences through a consensus mechanism naturally arises. We will propose a common framework to standardise consensus network methods, followed by a systematic study of different options on synthetic and real datasets.

1.2.4 Publications

Some of these presented contributions have been presented in different conferences and journal papers. We have also made publicly available the code to evaluate GRN inference methods easily. Here, we present a selection of the papers produced during the Ph.D. either as a first author or as collaborator:

- Bosio, M., Bellot, P., Salembier, P., and Oliveras-Vergés, A. (2011). Feature set enhancement via hierarchical clustering for microarray classification. In Genomic Signal Processing and Statistics (GENSIPS). (Bosio et al., 2011)
- Bosio, M., Bellot, P., Salembier, P., and Oliveras-Vergés, A. (2012). Gene expression data classification combining hierarchical representation and efficient feature selection. Journal of Biological Systems, 20(04):349-375. (Bosio et al., 2012)
- Bellot, P. and Meyer, P. E. (2014). Efficient combination of pairwise feature networks. JMLR: Workshop and Conference Proceedings (Bellot and Meyer, 2014)
- Bellot, P., Salembier, P., Oliveras-Vergés, A., and Meyer, P. E. (2015). Study of Normalization and Aggregation Approaches for Consensus Network Estimation. In 2015 IEEE Symposium Series on Computational Intelligence, number 1, pages 1081-1086. (Bellot et al., 2015b)
- Bellot, P., Olsen, C., Salembier, P., Oliveras-Vergés, A., and Meyer, P. E. (2015). NetBenchmark: a bioconductor package for reproducible benchmarks of gene regulatory network inference. BMC bioinformatics, 16(1):312 (Bellot et al., 2015a)
- Pham, N. C., Haibe-Kains, B., Bellot, P., Bontempi, G., and Meyer, P. E. (2016). Study of Meta-Analysis Strategies for Network Inference using Information-Theoretic Approaches. Biological Knowledge Discovery and Data Mining. (Pham et al., 2016)

Note that some of these publications are not on the topic of this thesis, and will not be covered in the present document.

1.3 Outline

The next chapter explains the preliminary background required for network inference, such as the statistical foundations of machine learning and the basics of GRN evaluation metrics.

Chapter 3 will present the state-of-the-art in network inference as well as our proposal on network inference (VIGR), with its performance evaluation and comparison to other methods with state-of-the-art GRN benchmarking framework. The fourth chapter presents our contributions to GRN benchmarking framework (NetBenchmark), and experimental contributions showing the effectiveness of this proposed framework. Fifth chapter presents the basics of consensus networks along with its alternatives, and offers a framework to standardise consensus network methods. Finally, the sixth chapter presents the conclusions of this thesis.

Chapter 2

Gene Regulatory Network - State of the art

Abstract

This chapter provides a general introduction to the subject of modeling biological networks; a discussion of different biological networks used in systems biology research, the purpose of biological network inference and of transcriptional networks. Elementary concepts and definitions of network properties, and biological data used for network reconstruction are also discussed.

2.1 Gene regulation and gene regulatory networks

When we say that we understand a biological process and model it, it means that we are able to use this model to predict future events and/or manipulate the process to obtain a desired output. Gene regulation and gene regulatory networks are an attempt to understand and explain how a biological system transits from one state to another. Such models try to cover a wide range of biological phenomena from cell differentiation to cancer disease.

This is what we call systems biology, a field where complex biological systems are modeled with computational and mathematical tools. Therefore it is an engineering-based approach applied to biological scientific research that examines the interactions between the components of biological systems, and how these interactions give rise to the function and behavior within biological systems.

A powerful and general approach to this problem is network analysis and its two fundamental stages, network reconstruction and network interrogation. Multiple works have been using such methods to gain a systems-level understanding of biological processes and to reveal mechanisms of different diseases.

However, due to the quick evolution of techniques and omics datasets that are capable of measuring several thousands of gene expressions at the same time, we deal with bigger and bigger networks, adding a challenge in the comparison of models and increasing the need for systematic methods to infer networks in a scalable way (computationally speaking).

As we have already mentioned, this thesis will present a study in the field of inferring the GRN from gene expression data. These networks are represented in the form of graphs. A graph is a mathematical tool which consist in a collection of vertices - or nodes- that are connected by edges. GRN's may be represented in such a way: genes correspond to nodes and a relationship between two genes is encoded by an edge. If for example genes g_i and g_j are connected, the graph involves an edge between them denoted by a_{ij}. The basics of Graph theory useful in the context of this document will be presented in subsection 2.3.

2.2 Biological systems and network inference

The scientific community has gained an extensive knowledge of genes and proteins as individual components. However, we are still far from understanding the functioning of a biological cell. Chemical switches attached to the DNA turn genes expression on or off, controlling at the end which proteins to produce and in what quantities. That is why in order to gain a systems-level understanding, we also need to examine how the components interact.

The expression level of a gene can be understood through the central dogma of molecular biology (Crick, 1970): DNA has two parts, one is coding and the other is non-coding. The coding parts are what we call genes. All cells in the same organism contain the same genes, but only certain genes are active in each cell. When the gene is active, it produces a particular protein, this process is known as synthesis. The process consists of two main steps: first, the gene is copied into messenger RNA (mRNA), process known as transcription. Second, the messenger RNA is translated into a protein (translation step).

The transcription can occur by means of proteins called Transcription Factor (TF) that bind to DNA sequences, enabling or inhibiting their transcription. They are said to regulate

the transcription. The resulting phenotype of a gene being transcribed is called gene expression. It can be measured through the quantity of RNA present for each gene. In practice, among other alternatives, microarrays can be used to do this task, and researchers can analyse the expression patterns of tens of thousands of genes at the same time.

This data tend to have a large number of gene expression values per sample (several thousands to tens of thousands), and a relatively small number of samples (e.g., a few dozen samples in relatively rare types of cancer) due to the cost of the experiments.

2.2.1 Gene Regulatory Network

We further explain in this section, how the regulation of gene expression is possible. Some specific proteins called Transcription Factor (TF) binds to the promoter regions of their Target Genes (TG). Through this process TF can activate or inhibit their expression. Genes do not work in isolation; they are connected in highly structured networks where the information flows through the cell.

In gene network reverse engineering, gene expression levels are often measured in terms of mRNA concentrations. In this case, the graph nodes are associated with the mRNAs of the genes, and the regulatory influences can be seen of as the "projection" of the different types of regulatory interactions onto the "RNA space", as shown in Figure 2.1 which highlights the entire process: (A) Genes are transcribed to mRNA. The transcription rate may be regulated by proteins (incoming arrows). (B) mRNA transcripts are processed and the corresponding proteins are synthesized (translation). Some RNAs have a regulatory function and are not translated. (C) Regulatory proteins (transcription factors) bind to the DNA. Proteins also interact among each other. (D) In practice, a simplified representation is commonly used in gene network models when only mRNA levels are measured (as in microarray data). The real interactions are collapsed in this simplified representation.

Understanding TF-TG interactions has many potential far-reaching applications in biology and medicine, ranging from the *in silico* modeling and simulation of the Gene Regulatory Networks (GRN) to the identification of new potential drug targets. However, while many TF-TG interactions have been experimentally characterized in certain well-studied organisms, the systematic experimental characterization of the full GRN remains a difficult task due to the large number of potential regulations. The goal of the GRN inference methods is to identify the regulatory links of the network, and possibly to infer a quantitative model to be used.

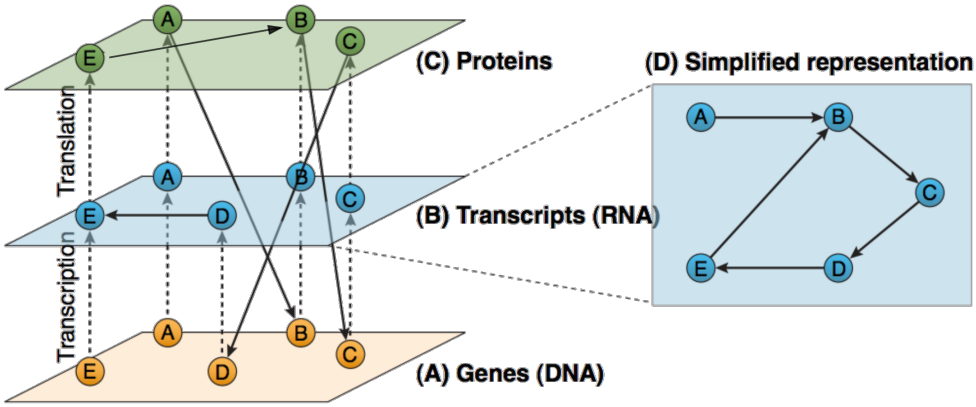


Fig. 2.1 Illustrative example of a gene regulatory network. Figure taken from (Marbach, 2009)

2.3 Basics of Graph theory

In this document, the following notations will be used: The expression measurement of gene i is associated with the random variable g_i and the set of expression measurements for all the genes with X. The gene expression profile g_i contains the measurements of a gene expression corresponding to several conditions, that could be natural conditions or resulting from an intervention, these experiments may vary and typically include knockdown or knockout experiments. So, the matrix X contains the expression of all genes under M different conditions or experiments, and can include replicates. If the system has N genes that are measured in M experiments, then $X \in \mathbb{R}^{M \times M}$.

We are going to work only with steady-state data set (X). Even though time-series data includes more information about the system dynamics, however the data is more difficult to obtain and the identification would be more difficult due to the high computational cost.

The final goal is to understand complex real biological systems, but real data suffers from several drawbacks. First, typically we only have access to partial knowledge of the underlying network (Roy et al., 2010), where a false positive could be a still undiscovered true positive. Therefore, the predicted gene network needs to be validated with additional experiments, to test if the predictions are correct.

Graph theory is the branch of mathematics that studies graphs. In this section we will present a brief review of the most important aspects that will be needed for understanding the following chapters.

Mathematically speaking a graph is an ordered pair $\Gamma = (V,E)$ comprising a set V of vertices or nodes together with a set E of edges, which are 2-element subsets of V (an edge is associated with two vertices, and that association takes the form of an ordered pair). Two vertices are adjacent if they are connected by an edge. A particular vertex can be connected with more than one vertex, the number of edges that end (or start) at this particular edge is called the degree of the vertex, and is denoted as $\deg(g_i)$. A connected sequence of edges in a graph is a path whose length is the number of traversed edges.

Two different graphs graphs Γ and Λ can be isomorphic. To be so, they should have the same number of vertices and be connected in the same way (even though that their vertices are different).

In our context the labeling of the vertices of the graph $\Gamma = (V,E)$, corresponds to the mapping $\alpha : V \to A$, where A is the set of genes $G = \{g_1, \ldots, g_N\}$. Similarly, the edges could also be labeled with a mapping $\beta : E \to B$. Often, these labels are numbers and we call them weights, obtaining a weighted graph. In the context of GRN inference these weights usually correspond to the confidence in the existence of the relationship.

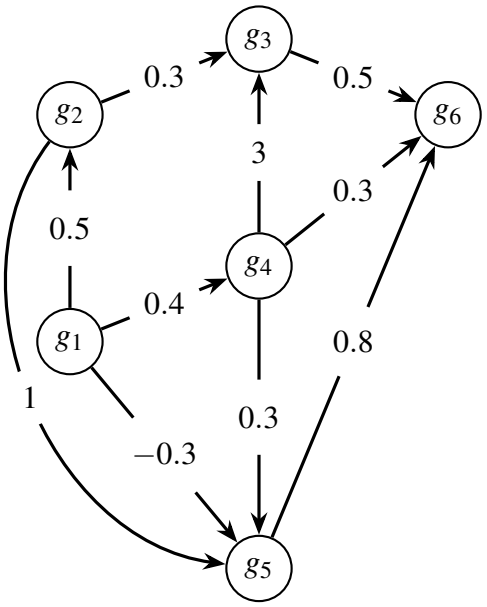
Matrix representation of Graphs The *adjacency matrix* of the graph $\Gamma = (V,E)$ is a matrix $A \in \mathbb{R}^{N\times N}$, where N is the number of vertices in $\Gamma, V = G = \{g_1, \ldots, g_N\}$. The adjacency matrix is defined such that its elements a_{ij} are the weights of the edge from vertex i to vertex j. A zero weight means that there is no edge. For example, Figure 2.2 shows a directed graph and its adjacency matrix.

Degree distribution In the study of graphs and networks, one of the main properties and characterisations is the degree distribution $P(k)$.

As already mentioned, the degree of a node in a network is the number of connections that it has to other nodes. As a particular case if the network is directed, then we should differentiate between the in-degree, which is the number of incoming edges, and the out-degree, which is the number of outgoing edges.

The degree distribution $P(k)$ is the probability distribution of these degrees over the whole network. It gives the probability that a selected node has exactly k links. $P(k)$ can be computed as the fraction of nodes in the network with degree k. For example, for the directed network of Figure 2.2, the out-degree distribution is $P(k)|_{k=0} = 1/6$, $P(k)|_{k=1} = 2/6$, $P(k)|_{k=2} = 1/6$ and $P(k)|_{k=3} = 2/6$.

This graph property allow us to characterize a graph into a feature vector (Xiao et al., 2008) and allows us to distinguish between different classes of networks. It has been observed



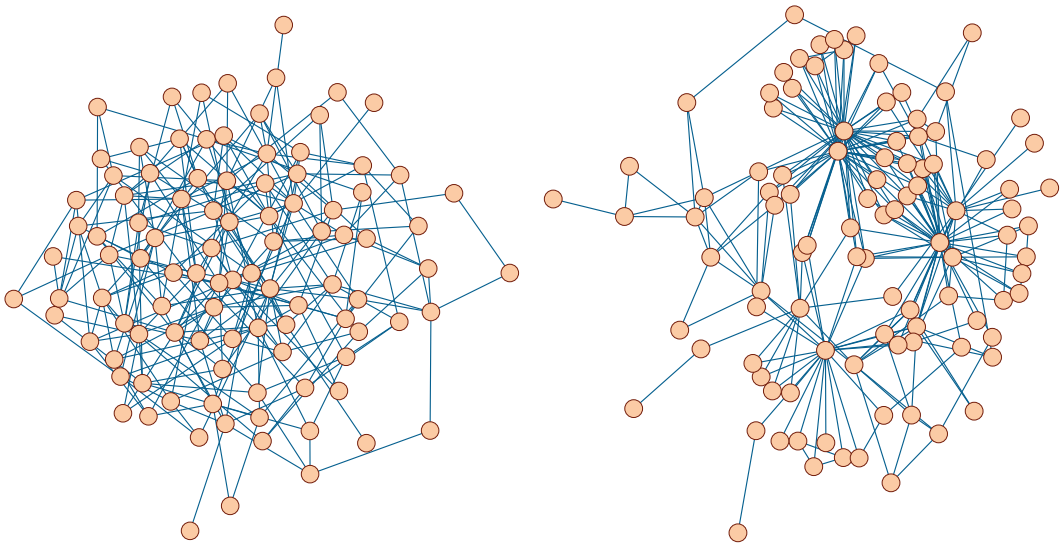
$$A = \begin{array}{c} \begin{matrix} g_1 & g_2 & g_3 & g_4 & g_5 & g_6 \end{matrix} \\ \begin{pmatrix} 0 & 0.5 & 0 & 0.4 & -0.3 & 0 \\ 0 & 0 & 0.3 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5 \\ 0 & 0 & 3 & 0 & 0.3 & 0.3 \\ 0 & 0 & 0 & 0 & 0 & 0.8 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \begin{matrix} g_1 \\ g_2 \\ g_3 \\ g_4 \\ g_5 \\ g_6 \end{matrix}$$

Fig. 2.2 A directed graph and its adjacency matrix.

that the degree distribution of many real networks decays as a power law $P(k) \sim k^{-\gamma}$, which is the defining property of scale-free networks (Barabási and Albert, 1999; Barabási and Bonabeau, 2003; Barabási and Oltvai, 2004; Jeong et al., 2000).

The topology of scale-free networks is dominated by a few highly connected nodes (hubs) which link the rest of the less connected nodes to the system. This property gives the network a great tolerance against errors, the nodes are able to communicate even with very high failure rates (Albert et al., 2000). Those properties appear in the network when $2 < \gamma < 3$, but when $\gamma > 3$ the network behaves like a random network (Erdös and Rényi, 1960), which is denoted as Erdös–Rényi (ER) model.

Figure 2.3 illustrates the difference between both models, the two networks have 100 nodes and around 250 edges. While in (a) each node has approximately the same number of links, in (b) a few nodes are very highly connected. Figure 2.4 shows the respective degree of both networks and the fitted power law distribution. We can observe that the degree of b) obtains a better fit to the distribution but not perfect, some authors propose to model these networks with a "partial" or "truncated" power law (Amaral et al., 2000).



(a) Erdös–Rényi (ER) model of a network with 100 nodes.

(b) Scale-free network with 100 nodes.

Fig. 2.3 Two different network models to understand biological networks.

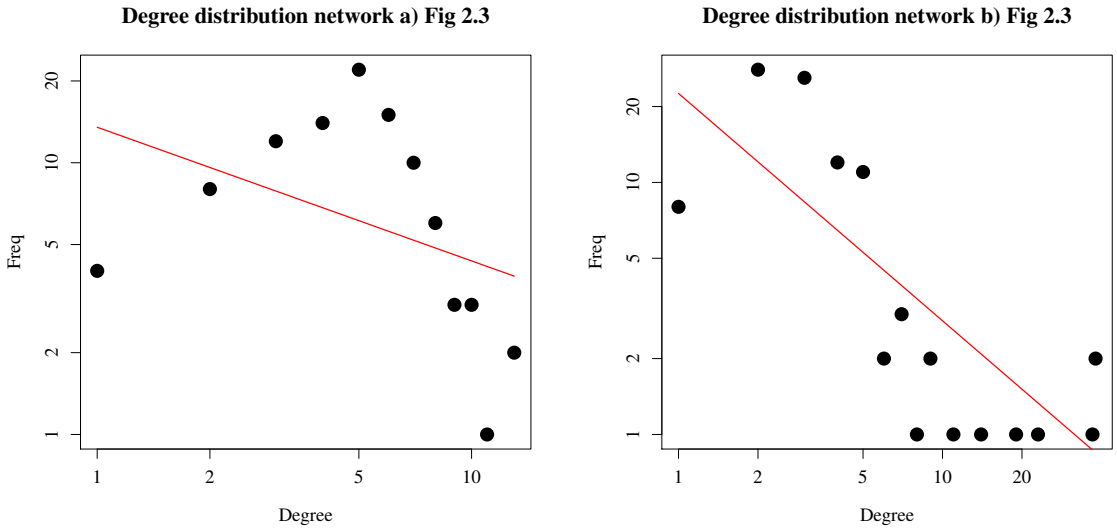


Fig. 2.4 Degree distribution of the networks from Figure 2.3

.

2.4 Network inference - State of the art

In general, GRN inference is performed by applying a learning algorithm that fits a mathematical model to the provided data. Therefore it pertains to a problem of machine learning. In this section we will first present briefly the three categories of machine learning that also

applies to GRN inference algorithms. The choice of an appropriate learning algorithm is mainly influenced by the model as well as by the quality and the quantity of the available data. Consequently, we will also review the different data that will be used in this thesis. In section 3.1 we will present the main state of the art of the GRN inference algorithms.

2.4.1 Categorization of network inference algorithms

Inference of the gene regulation network from expression data is a data mining and machine learning problem. Machine learning is the field that "gives computers the ability to learn without being explicitly programmed" (Simon, 2013). It involves the design of algorithms that are able to learn models in order to make and improve predictions or behaviors based on some data. It is said that these algorithms are data-driven. There are three main kind of learning algorithms:

Supervised Learning This kind of algorithms learn from examples. We have some data (X) where we already know the output (Y). The algorithm should learn the function describing the relationship between the input (X) and the output : $Y = f(X)$

Unsupervised Learning In this category of algorithms, we only have input data (X) but no corresponding output variables. Therefore these algorithms model the underlying structure or distribution of the data to discover and present the interesting structure in the data.

Semi-Supervised Learning Finally there are "intermediate" forms between the two previous classes of algorithms. Technically, in semi-supervised learning we have some input data (X) but only some of the data is labeled (Y). In semi-supervised learning, algorithms usually start with the labeled examples, and then try to guess the unlabeled data.

Inference of gene regulatory networks (Maetschke et al., 2014) categorized the existing GRN algorithms into these three groups. Unsupervised methods do not use any data to adjust their parameters while supervised ones use all the available data to optimize their parameters. In this thesis we will focus only on unsupervised GRN inference methods since there are few known regulatory networks and moreover the gene expression values present some discrepancies between them (Maetschke et al., 2014).

2.4.2 Data and experiments

As we already mentioned, the real data coming from few partially known networks present several drawbacks. First, they only provide a partial knowledge of the underlying network, where a false positive could be a still undiscovered true positive. Second, the intensity of noise is uncontrollable. Hence, assessing a method's robustness to varying intensities of noise cannot be done easily with real data. However, different noise intensities and distributions are observed from different measurement platforms (i.e. microarray vs RNAseq) as well as from different organisms. As a result, assessing the performance of any reverse-engineering algorithm on a few real datasets gives little information on its performance on other type of organisms or measurement platforms. This motivates the need for artificial generators that produce *in silico* data. The *in silico* datasets allow us to assess the performance of methods and can give insights into their strengths and weaknesses, as will be discussed in Chapter 4.

Although no artificial generator is really equivalent to real data, an in silico analysis gives reliable guidelines on algorithms' performance in line with the results obtained on real data sets (Bansal et al., 2007). Additionally, the use of multiple datasources coming from different simulators renders the subsequent analysis of methods more credible before any use on real data. Therefore, in order to provide a sound and fair comparison of the different methods, the use of various simulators becomes essential.

This fact raises the issue of how to generate realistic in silico data in particular with realistic underlying structures and data generation models. Several simulators have been proposed. They rely on different principles to generate a realistic underlying network as well as gene interaction kinetics models. In the context of this thesis we will use three of them:

GNW The GeneNetWeaver (GNW) simulator (Schaffter et al., 2011) generates network structures by extracting parts of known real GRN structures capturing several of their important structural properties. To produce gene expression data, the simulator relies on a system of non-linear ordinary differential equations (ODEs), and is able to generate gene knockouts (deletions of the gene setting the transcription rate of the genes to zero), gene knockdowns (the expression of one or more genes are reduced) and multifactorial data (involving or including several perturbations to a number of genes).

SynTReN The SynTReN simulator (Van den Bulcke et al., 2006) generates the underlying networks by selecting sub-networks from E. coli and Yeast organisms. Then the experiments are obtained by simulating equations based on Michaelis-Menten and Hill kinetics under different conditions.

Rogers The data generator described in (Rogers and Girolami, 2005) that will be referred as Rogers (as in (Olsen et al., 2009)) relies on a power-law distribution of the number of connections of the genes to generate the underlying network. The steady state of the system is obtained by integrating a system of differential equations simulating only knockout data.

DREAM challenges In order to understand the advantages and limitations of the different network inference methods, the DREAM project (Stolovitzky et al., 2007) that stands for Dialogue on Reverse Engineering Assessments and Methods has proposed several challenges on various biological problems to enable such an assessment through standardized performance metrics and common benchmarks.

There are two particular editions the DREAM4 and DREAM5 that proposed to recover gene regulation networks from data. Participants were challenged to infer the network structure from the given in silico and real gene expression datasets. (Marbach et al., 2012a, 2010, 2009; Prill et al., 2010)

2.4.3 GRN inference

The GRN inference methods aim to recover the topology of a GRN based on expression data only. Many computational methods have been developed to infer regulatory networks from gene expression data, predominately using unsupervised techniques. We will review them in detail in section 3.1.

Depending on the used inference algorithm, the resulting gene network can be either an undirected graph, in which the direction of the interaction is not specified ($a_{ij} = a_{ji}$), or a directed graph defining the direction of the interaction, that is gene j regulates gene i, and not vice versa ($a_{ij} \neq a_{ji}$).

2.5 GRN evaluation

The GRN inference methods aim to recover the topology of a GRN. Each one predicts a network that captures the relationships in the gene expression data. So, the natural question is how we compare them and know which one performs better in the task. In this context, the accuracy of a method is assessed by the extent to which the network it infers is similar to the true regulatory network. To do so, we need to evaluate and compare the inferred GRN with the Gold standard (GS). In this section, we explain how to perform this evaluation based on the framework proposed by the DREAM challenges (Marbach et al., 2012a).

2.5.1 Network evaluation as a binary classification task

The simplest way to compare networks is by means of edge comparison. For each pair of genes we can compare the status of an edge in both networks.

Therefore, a network reconstruction problem can be seen as a binary decision problem.

After thresholding the inferred adjacency matrix provided by the GRN algorithm, the final decision can be seen as a classification. For each possible pair of nodes, the algorithm either infers an edge or not. As a result, the algorithm will recover correct connections and misclassified connections.

Note that since the provided expression datasets do not contain temporal information, predicting self-interactions is irrelevant. Moreover, most of the state-of-the-art methods do not attempt to recover this kind of relationships. So, self-interactions are not considered to evaluate the networks.

For each target gene $i \in G$, the set of possible regulators would be $G_{-i} = \{G \setminus i\}$. The set of all candidate regulations is therefore $\xi = \{(t,g), g \in G, t \in G_{-i}\}$, and the GRN inference problem is to identify a subset of true regulations among ξ.

The GRN method computes a score $s : \xi \to \mathbb{R}$ to assess the evidence that each candidate regulation is true, and then predict as true regulation the pairs $(t,g) \in \xi$ for which the score $s(t,g)$ is larger than a threshold δ. The GRN method only focuses on finding a good ranking of the candidate regulations ξ such that true regulations tend to be at the top of the list. The threshold δ is left as a parameter, where large δ values correspond to less probable regulations.

As a result, a GRN method can be evaluated with standard performance metrics from machine learning, specifically precision vs. recall (PR) and receiver operating characteristic (ROC) curves (Davis and Goadrich, 2006). These scoring metrics are described in the next subsection.

2.5.1.1 PR and ROC curves

To score the ability of the different inference methods to predict the presence of regulatory interactions between genes, the GRN algorithm produces a ranked list of predicted edges for each network prediction. The list is ordered according to the confidence of the predictions, so that the first entry corresponds to the edge with the highest confidence.

The number of possible edges in a network of N genes, taking into account that autoregulatory interactions are not considered is $N^2 - N = N(N-1)$.

From a ranked edge-list of this kind, a particular concrete network with k edges is obtained by setting a cutoff k which considers the first k edges as present and the remaining edges as absent. Then, k is a parameter that controls the number of edges in a predicted network.

Recall is a measure of completeness,

$$\text{recall}(k) = \frac{\text{TP}(k)}{P} \tag{2.1}$$

where $TP(k)$ is the number of true positives at threshold k, and P is the number of positives (number of edges in the GS). Precision is a measure of fidelity,

$$\text{precision}(k) = \frac{\text{TP}(k)}{\text{TP}(k)+\text{FP}(k)} = \frac{\text{TP}(k)}{k} \tag{2.2}$$

where $FP(k)$ is the number of false positives at cutoff k in the edge list. The precision-recall curve graphically explores the trade-off between these complementary metrics as the k parameter varies. The Area under the precision-recall curve (AUPR) is a single number that summarises the precision vs. recall (PR) trade-off. As an alternative, the F-score (Powers, 2011) is also used. The traditional F-score (F1 score) can be interpreted as a weighted average of the precision and recall re-scaled between 0 and 1:

$$F_1 = 2\frac{1}{\frac{1}{\text{precision}}+\frac{1}{\text{recall}}} = 2\frac{\text{precision}\cdot\text{recall}}{\text{precision}+\text{recall}} \tag{2.3}$$

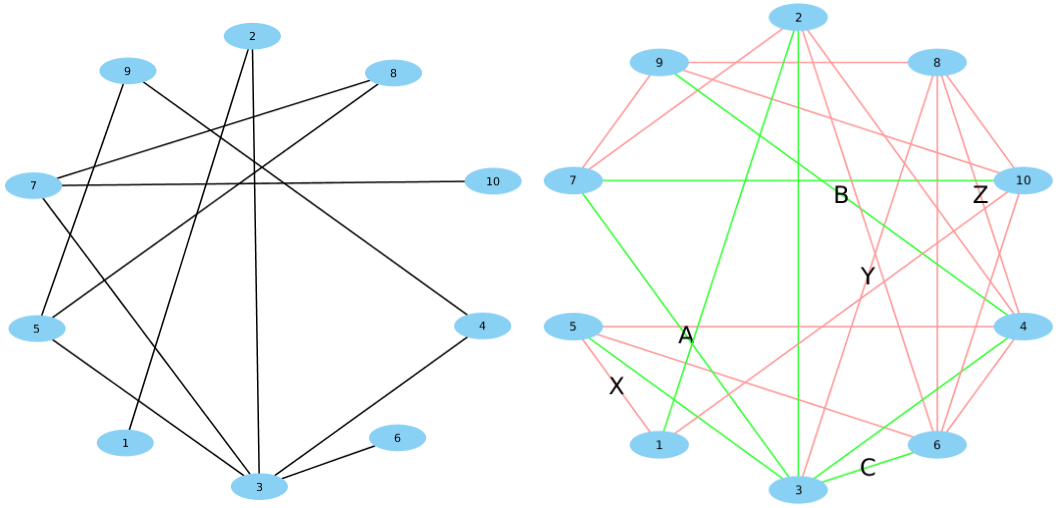
Similarly, the receiver operating characteristic (ROC) curve graphically explores the trade-off between the true positive rate (TPR) and the false positive rate (FPR). $\text{TPR}(k)$ is the fraction of positives that are correctly predicted at threshold k, and is equivalent to recall. $\text{FPR}(k)$ is the fraction of negatives that are incorrectly predicted at threshold k,

$$\text{FPR}(k) = \frac{\text{FP}(k)}{N} \tag{2.4}$$

Negative denotes the absence of an edge in the GS network. The area under the ROC curve (AUROC, also denoted AUC in the literature) is a single number that summarises the trade-off between $\text{TPR}(k)$ and $\text{FPR}(k)$ as the parameter k is varied.

A technical point is the issue of how to score a truncated prediction list. In this case, the list involves only a subset of the total number of possible edges. The missing edges may be added following a random order at the end of the list.

An example of a network prediction is illustrated in Figure 2.5 and expressed in form transcription factor-target gene interactions at Table 2.1. These interactions are denoted by the



(a) (b)

Fig. 2.5 Example of evaluation of a network as a binary classification task. (a) Gold standard (GS). (b) Inferred network, in green true positives (correct interacting gene pairs) and in pink the negatives or false positive (non-interacting pairs).

letters $A;B;C;\ldots;X;Y;Z$. In this example the method has highest confidence that interaction A is a true interaction. In Figure 2.5 and Table 2.1 green color denotes true positives (correct interacting gene pairs) and pink the negatives or false positive (non-interacting pairs). As we illustrated in this toy example, the topology of the inferred is not taken into account.

In Figure 2.6 the ROC and PR curves of the prediction of Table 2.1 are presented. The PR curves are more informative than the ROC curves for visualizing the performance at the top of the prediction lists. Moreover, precision vs. recall (PR) curves are recommended to use instead of ROC curves when the binary classification task presents a large class imbalance (Bunescu et al., 2005; Goadrich et al., 2004). This is the case of GRN evaluation as a binary classification task, biological networks follows a scale-free topology (see subsection 2.3), therefore their adjacency matrix is very sparse and as a consequence there is a high skew between both classes (link, no link).

An important difference between ROC space and PR space is the visual representation of the curves, looking at PR curves can expose differences between algorithms that are not apparent in ROC space (Davis and Goadrich, 2006). For example, the high precision of the most confident predictions of Table 2.1 is not apparent in the ROC curve. We address the interested reader to (Davis and Goadrich, 2006) for a discussion of the relation between PR and ROC curves.

Table 2.1 A hypothetical example of prediction of Figure 2.5, in green true positives (correct interacting gene pairs) and in pink the negatives or false positive (non-interacting pairs). Table is partially extracted from Supplementary information of (Marbach et al., 2012a).

Rank	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Prediction	A	B			C																	X	Y	Z

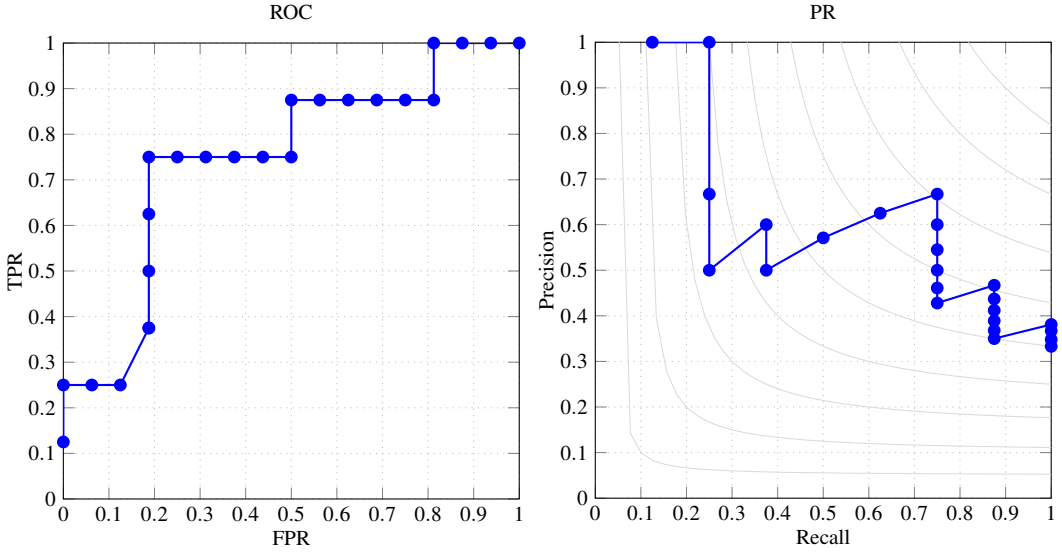


Fig. 2.6 ROC and PR curves for prediction of Table 2.1. ROC curve is shown at the left, and PR curve is shown at the right.

2.6 Conclusions

In this chapter, we have presented the foundations of modeling biological networks as well as some machine learning and graph theory concepts that will be needed along this thesis. We have also reviewed the kind of data used to do this task and its limitations. Finally, we have presented the process to evaluate a network and the classical performance measures that are used.

Chapter 3

GRN inference

Abstract

In this chapter, we present various types of network inference approaches currently used to recover GRNs. We present as well our proposal to solve the problem, we formulate the estimation and investigate the performance of several variable importance estimation methods. Finally, we examine different alternatives and their performance using the DREAM4 framework presented in the previous chapter.

3.1 GRN state of the art method

In this section, we are going to make a brief review of the different reverse engineering approaches: Gene identification algorithms based in co-expression, Information-theoretic approaches, Regression approaches and finally feature selection approaches.

Depending on the used inference algorithm, the resulting gene network can be either an undirected graph, in which the direction of the interaction is not specified ($a_{ij} = a_{ji}$), or a directed graph defining the direction of the interaction, that is gene j regulates gene i, and not vice versa ($a_{ij} \neq a_{ji}$).

We use the following notation: X_i denotes the expression levels of the ith gene in every experiment. It is a vector with M observations corresponding to the various experiments. Finally, the particular gene expression level of the kth experiment of the ith gene is denoted by x_{ik}.

3.1.1 Co-expression algorithms

These methods assume that similar patterns in expression profiles of genes that are under different conditions usually suggest relationships between them. Since the coordinated co-expression of genes encodes interacting proteins, studying co-expression patterns can provide insight into the underlying cellular processes.

Co-expression networks are reconstructed by computing a similarity score for each pair of genes. The most simple co-expression method uses the correlation between genes as similarity measure. If the correlation is greater than a threshold, then the genes are connected in the graph in an undirected way (because the correlation is symmetric).

GeneNet In (Opgen-Rhein and Strimmer, 2007), the authors propose a heuristic for statistically learning a "causal" network. It relies on the conversion of a network inferred through correlation into a partial correlation graph. Then, a partial ordering of the nodes is assigned by means of a multiple testing of the log-ratio of standardized partial variances. This allows identifying a directed acyclic causal network as a sub-graph of the partial correlation network.

MutRank MutRank (Obayashi and Kinoshita, 2009) ranks the correlation between every pair of genes and this rank is taken as the score that describes the similarity between genes. For every gene i, the Pearson's correlation (corr) with all other genes l is computed and ranked:

$$r_{ij} = \underset{j}{\text{rank}}(\,\text{corr}(X_i, X_l), \forall i \neq l). \tag{3.1}$$

As this expression is not symmetric, the final confidence score assigned between genes i and j is computed as the geometric mean of the scores obtained between gene i and j and vice versa:

$$s_{ij} = \frac{r_{ij} \cdot r_{ji}}{2}. \tag{3.2}$$

Weighted correlation network analysis Weighted correlation network analysis (WGCNA) (Langfelder and Horvath, 2008) is a correlation-based method that exaggerates high correlation values in a nonlinear way. This is achieved raising the absolute value of correlation to a power greater or equal to 1 ($\beta \geq 1$):

$$w_{ij} = |\,\text{corr}(X_i, X_j)|^{\beta} \tag{3.3}$$

Since this transformation is monotonic, the inferred network produces similar results to the correlation matrix (Maetschke et al., 2014).

Zscore Zscore (Prill et al., 2010) is a method that assumes interventional data, more concretely knockout experiments that lead to a change in other genes. The assumption is that the knocked-out gene i in the experiment k affects more strongly the genes that it regulates than the others. The effect of the gene i over gene j is captured with the Zscore z_{ij}:

$$z_{ij} = \left| \frac{x_{jk} - \mu_{X_j}}{\sigma_{X_j}} \right|, \tag{3.4}$$

assuming that the kth experiment is a knockout of gene i, μ_{X_j} and σ_{X_j} are respectively the mean and standard deviation of the empirical distribution of the gene j. To apply the original method, one needs to know which knockouts are done in each experiment. However, in practice, one can assume that the knocked-out gene is the one corresponding to the minimum value in the experiment k: $\arg\min_l(x_{lk}) = i$. With this generalization, the method can be applied to any type of data like multifactorial or knockdown data. If the same gene is detected to be knocked-out in various experiments, then the final Zscore is the mean of the individual Zscore values.

3.1.2 Information-theoretic approach

These approaches use a generalization of the pairwise correlation coefficient that is called mutual information (M_{ij}) (Cover and Thomas, 2005). It measures the degree of dependence between two genes X_i and X_j.

$$M_{ij} = \sum_{X_i} \sum_{X_j} p(X_i, X_j) \log_2 \frac{p(X_i, X_j)}{p(X_i)p(X_j)}, \tag{3.5}$$

where $p(X_i, X_j)$ is the joint probability distribution function of X_i and X_j, and $p(X_i)$ and $p(X_j)$ are the marginal probability distribution functions of X_i and X_j respectively (Cover and Thomas, 2005).

Relevance network The RELNET (Butte and Kohane, 2000) is the simplest method based on mutual information. For each pair of genes, the mutual information M_{ij} is estimated and the edge between genes i and j is created if the mutual information is above a threshold. Despite that mutual information is more general than the Pearson correlation coefficient, in

practice thresholding the M_{ij} or Pearson correlation produces similar results (Steuer et al., 2002).

CLR algorithm The Context Likelihood or Relatedness network (CLR) method (Faith et al., 2007) is an extension of the previous method. The method derives a score that is associated to the empirical distribution of the mutual information values. In practice, the score between gene i and gene j is defined as follows:

$$c_{ij} = \sqrt{c_i^2 + c_j^2}, \text{ with } c_i = \max\left(0, \frac{M_{ij} - \mu_{M_i}}{\sigma_{M_i}}\right) \text{ and } c_j = \max\left(0, \frac{M_{ji} - \mu_{M_j}}{\sigma_{M_j}}\right). \quad (3.6)$$

The mean and standard deviation of the empirical distribution of the mutual information of gene g_i are denoted by μ_{M_i} and σ_{M_i}, which are defined as:

$$\mu_{M_i} = \frac{1}{N}\sum_{l=1}^{N} M_{il}, \quad \sigma_{M_i} = \sqrt{\frac{1}{N-1}\sum_{l=1}^{N}(M_{il} - \mu_{M_i})^2}. \quad (3.7)$$

This process can be seen as a normalization of the mutual information (Bellot and Meyer, 2014).

ARACNE The motivation of the Algorithm for the Reconstruction of Accurate Cellular NEtworks (ARACNE) (Margolin et al., 2006) is that many similar measures between variables may be the result of indirect effects. In order to avoid the indirect effect, the algorithm relies on the "Data Processing Inequality" (DPI) which removes the weakest edge, that is the one with the lowest mutual information, in every triplet of genes.

MRNET MRNET (Meyer et al., 2007) method will be explained in the Feature Selection subsection.

PCIT The Partial Correlation coefficient with Information Theory (PCIT) (Reverter and Chan, 2008) algorithm combines the concept of partial correlation coefficient with information theory to identify significant gene-to-gene associations.

Similarly to ARACNE, PCIT extracts all possible interaction triangles and applies DPI to filter indirect connections, but instead of mutual information it uses first-order partial correlation as interaction weights. The partial correlation tries to eliminate the effect of a third gene l on the correlation of genes i and j.

C3NET The Conservative Causal Core NETwork (C3NET) (Altay and Emmert-Streib, 2010a) consists of two main steps. In the first step pairwise mutual information is computed. Then, non-significant connections are eliminated, according to a chosen significance level α, between gene pairs. This is recurrent to many GRN algorithms, like the previously presented algorithms (RELNET, ARACNE and CLR). But the main difference is the second step, where only the most significant edge for each gene is selected. This edge corresponds also to the highest mutual information value among the neighbouring connections for each gene.

The consequence of the second step is that the highest possible number of connections that can be reconstructed by C3NET is equal to the number of genes under consideration. C3NET does not aim at reconstructing the entire network underlying gene regulation but mainly tries to recover the core structure.

BC3NET In order to overcome the previous limitation of C3NET, in (Simoes and Emmert-Streib, 2012) a bagging procedure is introduced to improve C3NET. The basic idea of BC3NET is to generate from the dataset X that has M samples, B independent datasets. This is done sampling M samples with replacement B times, with a non-parametric bootstrap (Efron and Tibshirani, 1994).

For each of these B bootstraped datasets, a network (G_k^b) is inferred using C3NET (Altay and Emmert-Streib, 2010a), obtaining an ensemble of networks.

From this ensemble of networks, a weighted network G_w^b is constructed to determine the statistical significance of a connection between each pair if genes. The final result is a binary undirected network G.

3.1.3 Regression approach

If we had time-series data, these GRN inference algorithms would tackle the continuously varying quantities and their rates of change in time with the framework of differential equations.

The dynamics in a neighborhood of any given equilibrium point can be approximated by a set of linear differential equations. Therefore, the network can be described as a function of the concentrations of every other genes in the cell and the external perturbations:

$$\dot{g}_i = \sum_{j=1}^{N} a_{ij} g_j(t_k) + b_i u(t_k) \; \forall \; k = 1 \ldots K \tag{3.8}$$

The variable $g_i(t_k)$ is obtained by sampling from a continuous-time signal g_i, and then each k value corresponds to a sample. $b_i u$ stands for the external perturbation that can alter

the transcription rate of the genes in the cell. Since we are interested in the steady-state case, equation 3.8 should become independent of time, that is $\dot{g}_i = 0$. Therefore the equation is then simplified:

$$\sum_{j=1}^{N} a_{ij} g_j = -b_i u \tag{3.9}$$

Zavlanos algorithms Equation 3.9 was used by the Network Identification by multiple Regression (NIR) algorithm (Gardner et al., 2003), which solved it as a classic linear-regression problem (Hastie et al., 2009). The method only needs the information of the perturbed genes in each experiment and a sparseness assumption that determines a maximum number of genes that a single gene can be regulated by.

If we stack together equation 3.9 for every gene, the whole inference problem can be formulated by the following set of linear equations:

$$AX + BU = 0 \tag{3.10}$$

Where $A \in \mathbb{R}^{N \times N}$ encodes the interactions between the N individual genes at the given equilibrium, and the matrix $B \in \mathbb{R}^{R \times p}$ specifies the affected genes by each perturbation.

Then, we can find the linear model that best fits the experiments by minimizing an error in some norm. At the same time it is possible to incorporate any available a priory biological knowledge and a model of the matrix A as for example the sparsity of the network.

In (Zavlanos et al., 2011) the authors propose three different methods to solve equation 3.10. The methods rely on the belief that the matrix A should be stable and sparse. The problem is formulated as an optimization problem and solved with convex optimization techniques. Since the system is solved globally, it theoretically overcomes the NIR algorithm.

3.1.4 Feature selection approaches

A GRN reconstruction problem can also be seen as a feature selection problem. For every gene, the goal is to discover its true regulators among all other genes or candidate regulators. This approach can integrate knowledge about genes that are not TF's and therefore reduce the search space.

Typically, this approach only focuses on designing a significance score $s(i, j)$ that leads to a good ranking of the candidate regulations, such that true regulations tend to be at the

top of the list since an edge is assigned between i and j if the evidence $s(i,j)$ is larger than a threshold.

With the feature selection approach, the scores $s(i,j)$ for all the genes are jointly estimated with a method that is able to capture the fact that a large score for a link (i,j) is not needed if the apparent relationship between i and j is already explained by another and more likely regulation.

MRNET The Minimum Redundancy NETworks (MRNET) (Meyer et al., 2007) method reconstructs a network using the feature selection technique known as Minimum Redundancy Maximum Relevance (MRMR) (Ding and Peng, 2005), which is based on a mutual information measure. In order to get a network, the algorithm performs a feature selection for each gene ($i \in [1,G]$) on the set of remaining genes ($j \in [1,G] \setminus i$).

The MRMR procedure returns a ranked list of features that maximize the mutual information with the target gene (maximum relevance) and, at the same time, such that the selected genes are mutually dissimilar (minimum redundancy). For every gene, the MRMR feature selection provides a score of potential connections where the higher scores should correspond to direct interactions. The indirect interactions should have a lower scores because they are redundant with the direct ones. Then, a threshold is computed as in the RELNET method.

The MRNET reconstructs a network using a forward selection strategy, which leads to subset selection that is strongly conditioned by the first selected variables. The Minimum Redundancy NETworks using Backward elimination (MRNETB), uses instead a backward selection strategy followed by a sequential replacement (Meyer et al., 2010).

Tigress The Trustful Inference of Gene REgulation with Stability Selection (Tigress) algorithm (Haury et al., 2012) states a regression problem for each gene $g \in G$. For each regression problem, the algorithm tries to predict the expression level of g_i from the expression level of its candidate regulators. But the goal is not really to estimate the coefficients of regression function. It is the identification of a small set of transcription factors which are sufficient to provide a good model for g_i. Therefore, a score for each candidate transcription factor is defined to assess how likely it is to be involved in the regression model.

The Lasso (Tibshirani, 1996) algorithm is used as a feature selection technique, which leads to a sparse linear model for each gene. But since it is quite unstable and does not provide a confidence score of the transcription factors, a stability selection (Meinshausen and Bühlmann, 2010) procedure is used on top of it. For each gene, many lasso regression

are generated with randomly perturbed data. Then the score for each transcription factor is proportional to the number of times it was selected in the top five transcription factors.

Finally, the significance of each transcription factor in the target-gene-specific regression model is combined across all target genes and ranked.

Genie3 The GEne Network Inference with Ensemble of trees (Genie3) (Huynh-Thu et al., 2010) algorithm uses the random forests (Breiman, 2001) feature selection technique to solve a regression problem for each of the genes in the network. In each of the regression problems, the expression pattern of the target gene should be predicted from the expression patterns of all transcription factors.

The importance of each transcription factor in the prediction of the target gene is taken as an indication of an apparent regulatory edge. Then these candidate regulatory connections are aggregated over all genes to generate a ranking for the whole network.

3.2 Variable Importance on Gene Relationships

This section introduces our approach to network inference that is based on a feature selection technique with variable importance estimation: VIGR which stands for Variable Importance on Gene Relationships. Using variable selection strategies suggested for network inference has many advantages and has been previously used for network estimation in several other methods (Haury et al., 2012; Huynh-Thu et al., 2010; Meyer et al., 2007, 2010) (see Feature selection approaches in section 3.1).

For each target gene (i), a regression is formulated to estimate the relationships among variables. We try to capture how the expression of g_i changes when any one of the variables on G_{-i} varies. In other words, we should predict the expression g_i as a function of the expression level of its regulators $X_{G_{-i}}$:

$$g_i = f(X_{G_{-i}}) + \varepsilon \tag{3.11}$$

where g_i represents the expression level of the ith gene across different experiments (modelled as a random variable), $X_{G_{-i}}$ is the set of expression levels of the candidate transcription factors for gene i, and ε represents the noise. However, we are not interested in finding f, but in the identification of a small set of variables which are sufficient to provide a good model for g_i and rank them according to their likelihood of being involved in the regression model.

(Haury et al., 2012) proposed to solve this problem using Lasso algorithm (Tibshirani, 1996) with stability selection (Meinshausen and Bühlmann, 2010) to increase its robustness and to provide a score, while (Huynh-Thu et al., 2010) proposed to address this issue using random forest (Breiman, 2001) and score each predictor with a variable importance measure specific to this model.

We have adopted an approach inspired by the last strategy. The main idea is to obtain a ranking of the selected features based on the importance of each variable in the regression model. We have tested several alternatives to do so, but here we will show only the performance of two different ones: Partial Least Squares (PLS) Regression (Geladi and Kowalski, 1986) and Gradient Boosting Machine (GBM) (Friedman, 2002). Therefore, Variable Importance on Gene Relationships (VIGR) method will be named accordingly: VIGR.PLS and VIGR.GBM.

In the next subsections, we cover the basics of both methods as variable importance estimators.

3.2.1 Partial Least Squares

Partial least squares (PLS) (Geladi and Kowalski, 1986) regression is a technique that reduces the predictors ($X_{G_{-i}}$) to a smaller set of uncorrelated components and performs least squares regression on those transformed features, instead of on the original data.

PLS regression is especially useful when the feature set is highly collinear, or when the number of observations is smaller than the number of features. Both scenarios are usually present in our problem.

So, in a classical multiple linear regression:

$$g_i = b_o + \sum_{j \neq i} b_j g_j + \varepsilon \tag{3.12}$$

we would fit the different values of b_j to minimise the squared error. However, due to the problems mentioned before such simplistic approach usually fails.

Principal components regression was proposed to solve collinearity among predictors. It finds new uncorrelated predictors with a principal component analysis (Jolliffe, 2002) and then performs a multiple linear regression (see Equation 3.2.1), but instead of using raw predictors g_j it uses the principal component vectors .

PLS regression relies on the same principle, but it also finds hyperplanes of maximum variance between the response - in our context the target gene - and the independent variables -

the TF candidates. The linear regression model is found projecting the predicted variables and the observable variables to a new space with a particular dimension (number of components).

In this context, the variable importance measure is based on weighted sums of the absolute values of the regression coefficients. The weights are a function of the reduction of the sums of squares across the number of PLS components and are computed separately for each outcome. Therefore, the contribution of the coefficients are weighted proportionally to the reduction in the sums of squares.

Finally, in VIGR.PLS, we perform a PLS to solve N times the equation 3.2. For each $i \in G_{-i}$, we determine the optimal number of components with a cross-validation, in order to keep the computational load down the number of components that are explored lies in a fixed range. Then, the scaled estimated variable importance is set as the scores of potential links of regulators of the gene.

3.2.2 Gradient Boosting Machine

A lot of standard approaches for supervised machine learning are based on trees (Breiman et al., 1984). Gradient Boosting Machine (GBM) (Friedman, 2002) is also one of them.

A decision tree is a binary (not necessary balanced) tree, so it recursively partitions the input space (the features) with a simple thresholding of the features. In our case, we want to predict the expression of g_i based on the other set of expression genes ($X_{G_{-i}}$). For example, we can interpret the tree at Figure 3.1 asking a series of questions in the input space. For a particular point in the input space and based on the thresholds we will follow the branches until eventually, we get a leaf node. In our case of regression trees, those leaves contain constant prediction values and is usually the mean of all training examples of g_1 that ended up in a particular leaf. For example, in the root node of the tree of Figure 3.1 all the points in the input space for which the gene g_{42} have a smaller value than 0.55 will predict 0.19: $g_1|_{g_{42}<0.55} = 0.19$. If $g_{42} \geq 0.55$ then the tree is traversed with the splitting rules of the input space.

Since the leaves return a constant value, the kind of functions that trees learn are piecewise constant functions. Figure 3.2 shows the real function and different approximations with Regression Trees with different depth levels. As we increase the depth of the tree, it would have more leaves and therefore more constants segments to model the data. For example, the regression tree with $d = 1$ only has 2 leaves and consequently approximates the real value of g_1 with only two values.

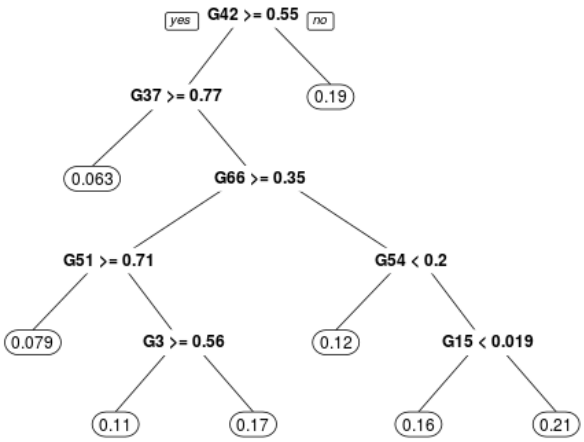


Fig. 3.1 Example of regression Tree for g_1

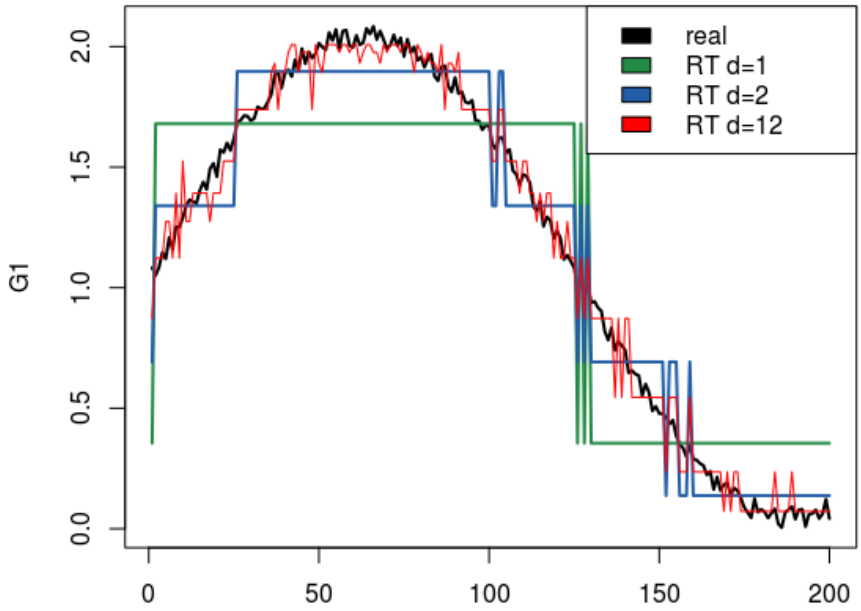


Fig. 3.2 Function approximation with Regression Trees

Trees are very easy to interpret and flexible and tend to ignore redundant variables, but they have poor predictive performance by themselves. However, they are heavily used with different ensemble techniques that aggregate a large number of trees, so averaging the models. There are thee main types: Random Forest, Bagging, or Boosting.

On the one hand, Bagging (Breiman, 1996) and Random Forest (RF) (Breiman, 2001) are based on the idea of fitting many large trees to a bootstrapped resampled versions of the training data so the trees will be different, and of averaging the outputs to reduce the variance of the model. On the other hand, Boosting (Freund and Schapire, 1995) fits many large or

small trees to reweighed versions of the training data in a way that gives more weight to areas where the error of the previous trees is larger.

So, Boosting learns the parameters of the model in a stagewise manner to decrease the overfitting rate. Understanding Boosting as minimising an arbitrary loss function in this kind of stagewise fashion permits to extend the paradigm (Friedman, 2002). Since GBM learns from the errors it could end up overfitting, so the tree size becomes an important parameter since it determines the order of interaction (Hastie et al., 2009).

Both methods belongs to the so-called "meta-algorithms": approaches to combine several machine learning techniques into one predictive model in order to decrease the variance (bagging) or bias (boosting). Bagging generates additional data for training from your original dataset using combinations with repetitions. Boosting is a two-step approach, that uses subsets of the original data to produce a series of averagely performing models, this subset creation is not random and depends upon the performance of the previous models.

RF and GBM's are capable of providing a variable importance measure.

In RF for each tree, the MSE is computed on the out-of-bag (OOB) data for each tree. OOB is a method of measuring the prediction error defined as the mean prediction error on each training sample, using only the trees that did not have the training sample in their bootstrap sample. The same is computed after permuting a variable, and the differences are averaged and normalised by the standard error. We refer to the interested reader to (Louppe et al., 2013) for a detailed interpretation.

The relative influence in GBM described in (Friedman, 2001) is similar to the variable importance measures Breiman uses for random forests (Breiman, 2001), but it is computed using the entire training dataset (not the out-of-bag observations). In the words of (Elith et al., 2008) "the measures are based on the number of times a variable is selected for splitting, weighted by the squared improvement to the model as a result of each split, and averaged over all trees".

Finally, to perform VIGR.GBM, GBM is used to solve N times the equation 3.2. For each $i \in G$, the scaled estimated variable importance is set as the scores of potential links of regulators of the gene. We determine the optimal number of iterations with a cross-validation. In order to keep the computational load down, the maximum number of iterations is set to 100. Then, the scaled estimated variable importance is set as the scores of potential links of regulators of the gene.

3.3 Benchmark

To evaluate the proposed method against the state of the art GRN inference methods we used the framework and the data of DREAM4 *in silico* network challenge. DREAM4 offers a benchmark suite for performance evaluation of methods for gene network inference (reverse engineering).

All the data have been generated with GeneNetWeaver (GNW) (Schaffter et al., 2011) version 2. The GNW simulator creates network structures by extracting parts of known real GRN structures capturing several of their important structural properties. To produce gene expression data, the simulator relies on a system of non-linear ordinary differential equations (ODEs). It can generate gene knockouts (deletions of the gene setting the transcription rate of the genes to zero), gene knockdowns (the expression of one or more genes are reduced) and multifactorial data.

In particular, we have used data of the third subchallenge. In this challenge, only the multifactorial perturbation dataset was provided. This data consists of different variations or observations of the network that simulates samples from different patients. The goal is the prediction of the network structure. This challenge consisted of five networks of 100 genes each one.

Network topologies were obtained by extracting subnetworks from the transcriptional regulatory networks of model organisms. In particular, two networks were used:

- Escherichia coli transcriptional regulatory network: 1502 nodes, 3587 edges. Corresponds to the TF-gene interactions of RegulonDB release 6.2 (Gama-Castro et al., 2008).
- Saccharomyces cerevisiae (yeast) transcriptional regulatory network: 4441 nodes, 12873 edges (Balaji et al., 2006).

The results that we show in this section have been obtained with the R versions of the available algorithms. The complete list of package versions is shown in section A.1 of supplemental material. The evaluation has been done with a porting of the Matlab code of DREAM framework (Marbach et al., 2012a) and can be accessed at https://github.com/paubellot/DREAM_Evaluation.

Table 3.1 presents the Area under the precision-recall curve (AUPR) obtained within the DREAM framework while Table 3.2 shows the AUROC. Both tables show the performance measures as well as the mean and variance across the five different networks of the challenge of the particular measure for various GRN inference methods. The best result (or results) is

Table 3.1 AUROC values of different methods evaluated on DREAM4 multifactorial network

Method	ARACNE	C3Net	CLR	Genie3	MRNET	MRNETB	PCIT	Zscore	VIGR.PLS	VIGR.GBM
Net1	0.609	0.585	0.640	0.682	0.645	0.643	0.653	0.491	0.660	0.639
Net2	0.570	0.547	0.693	0.723	0.689	0.696	0.658	0.481	0.709	0.660
Net3	0.647	0.625	0.709	0.744	0.718	0.714	0.704	0.509	0.697	0.693
Net4	0.623	0.605	0.728	0.749	0.728	0.727	0.695	0.486	0.701	0.683
Net5	0.660	0.603	0.739	0.762	0.741	0.740	0.683	0.506	0.727	0.688
μ	0.621	0.593	0.702	**0.732**	0.704	0.704	0.679	0.495	0.699	0.673
σ	0.035	0.029	0.039	0.031	0.038	0.038	0.022	0.012	0.024	0.023

(are) highlighted in bold. These results are presented in a graphical manner in Figure 3.3 and Figure 3.4. Both Figures present in the bars the mean AUROC / AUPR, respectively, and the ticks represent the standard deviation of the metrics obtained for each method.

Observing these tables and figures we can conclude that Genie3 obtains the best performance within this evaluation. Even though, all methods present a similar behaviour across different networks of DREAM4 challenge, with the noticeable exception of the Zscore that obtains a very poor performance. We believe that this is caused by the kind of data since Zscore was originally designed for knockout data that is not the case in this challenge. We can conclude that our presented approach based on Variable Importance estimated with different methods is competitive compared with the other methods. Furthermore, let us notice that MRNET, CLR also exhibit competitive performances.

Therefore, we have selected these five methods to perform a deeper analysis of their results. We present at Figure 3.5, some of the ROC and PR curves to illustrate the main findings. Observing the subfigure 3.5b we can realise that the different performances between methods that Table 3.2 reveals are not that different. Also, if we observe subfigure 3.5d we could question if some of these methods obtain better networks than others. Even though having similar AUPR performances, if we had computed the accumulated area under the curve we would have observed that for different algorithms the accumulated area would have been obtained at different recall levels. For example, our proposed method VIGR.GBM obtains the great part of his area under the curve between 0.1 and 0.3 values of recall while CLR or MRNET obtain their respective area with very small values of recall. This fact implies that CLR and MRNET lists have a greater precision in the top predicted links.

Here, we have tested the different algorithms on the DREAM4 framework. DREAM challenges perform an evaluation on partial knowledge of the underlying network where a false positive could be a still undiscovered real positive. Therefore, a predicted link that has its origin outside the TF list is eliminated. This could result in an overestimation of the precision and underestimation of the FPR.

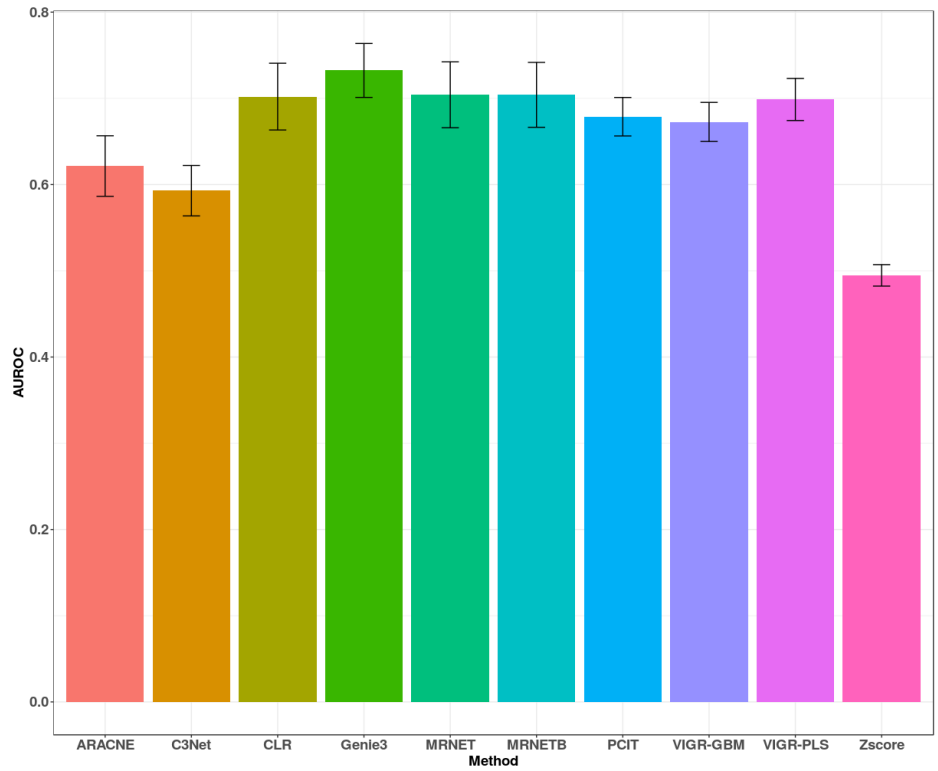


Fig. 3.3 Mean AUROC values for each GRN inference method on the different datasources of the DREAM4 multifactorial networks. The best results are highlighted.

We can notice that this presented benchmark is not able to discriminate the performances of the different methods since they obtain very similar metrics. Moreover, it does not allow to evaluate the changes of performances of the methods as a function of the number of genes, of the number of experiments or of the intensity of noise for multiple topologies. Hence, we conclude that the benchmarking framework should be improved, provide more precise assessment and designed in such a way that the experimental setting can be easily modified. Moreover, it would be interesting to develop a tool that is available to the research community and that provides reproducible assessment of current or future GRN inference techniques.

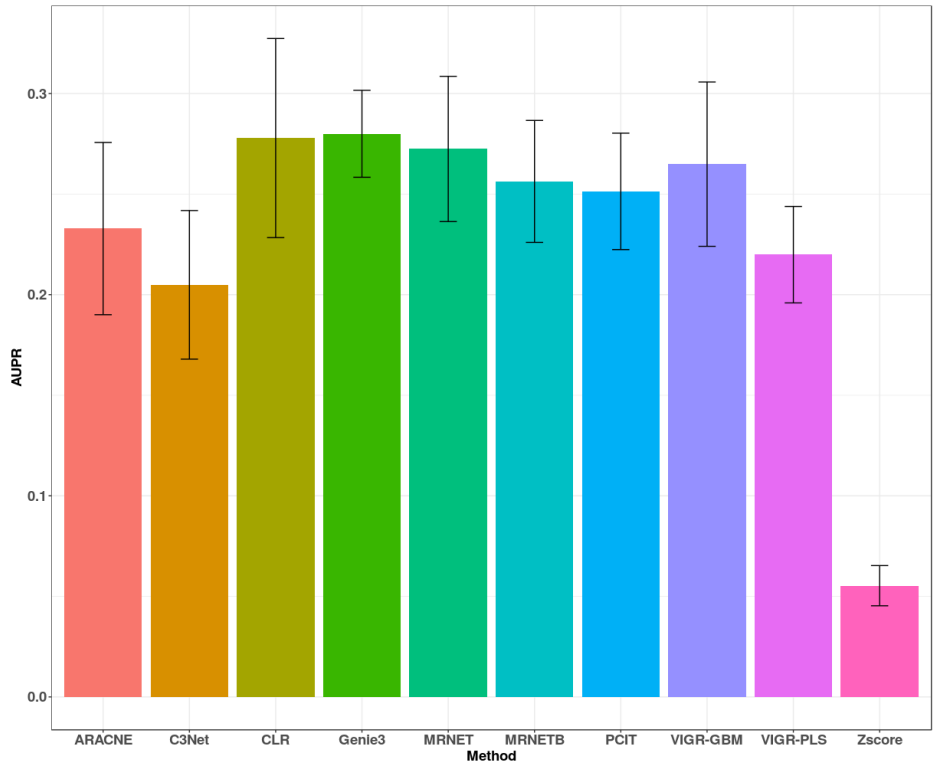
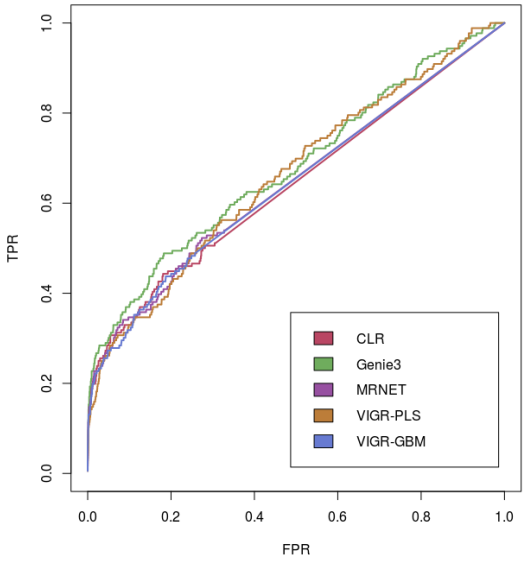


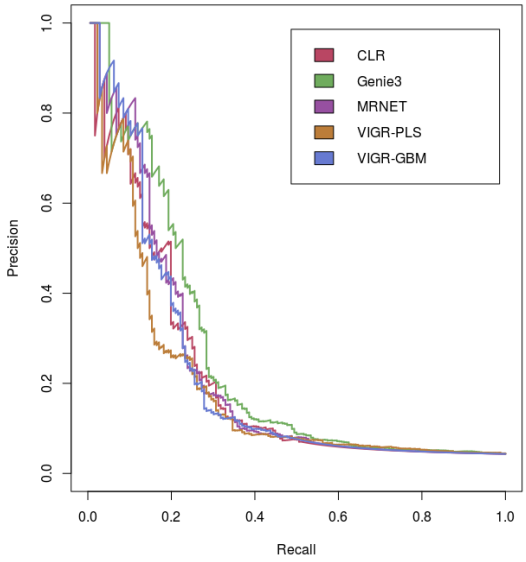
Fig. 3.4 Mean AUPR values for each GRN inference method on the different datasources of the DREAM4 multifactorial networks. The best competitive results are highlighted.

Table 3.2 AUPR values of different methods evaluated on DREAM4 multifactorial networks

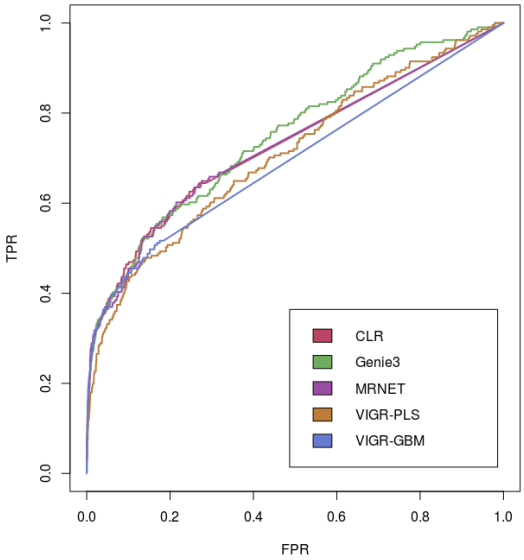
Method	ARACNE	C3Net	CLR	Genie3	MRNET	MRNETB	PCIT	Zscore	VIGR.PLS	VIGR.GBM
Net1	0.202	0.187	0.208	0.247	0.218	0.220	0.246	0.043	0.190	0.205
Net2	0.176	0.151	0.244	0.270	0.253	0.280	0.277	0.065	0.199	0.241
Net3	0.266	0.248	0.308	0.293	0.293	0.246	0.246	0.050	0.228	0.287
Net4	0.244	0.221	0.312	0.299	0.304	0.295	0.279	0.053	0.239	0.300
Net5	0.276	0.218	0.318	0.292	0.294	0.241	0.208	0.066	0.243	0.292
μ	0.233	0.205	**0.278**	**0.280**	**0.272**	0.256	0.251	0.055	0.220	**0.265**
σ	0.043	0.037	0.049	0.022	0.036	0.030	0.029	0.010	0.024	0.041



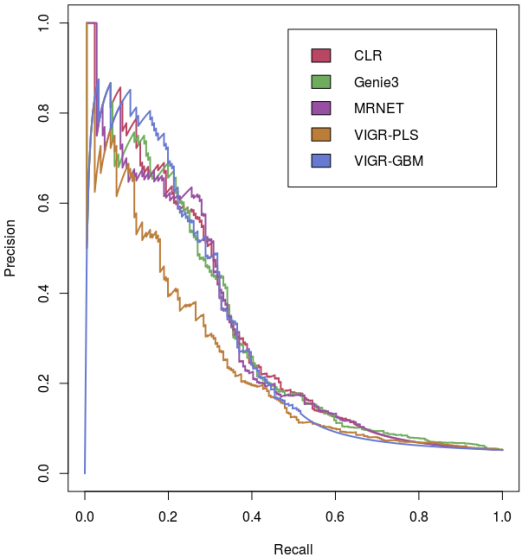
(a) ROC curve network 1



(b) PR curve network 1



(c) ROC curve network 4



(d) PR curve network 4

Fig. 3.5 ROC and PR curves of selected GRN algorithms in DREAM4

3.4 Conclusions

We have presented several methods for reconstructing gene regulatory networks from expression data that have been proposed during the last decades. Using DREAM4 framework we have assessed their performance and compared them with our proposal. We have concluded that Genie3 obtains the best performance within this evaluation, but also that our proposed approach is competitive compared with the other methods.

We have also highlighted the need for a new benchmark framework able to discriminate the performances of the different methods and allowing to evaluate the impact on the performances of the number of genes, of the number of experiments or of the noise intensity.

Chapter 4

GRN performance evaluation: NetBenchmark

Abstract

Predicting regulatory interactions from gene expression data have become a key challenge in recent years. Many network inference algorithms have been proposed to tackle this problem. In this chapter, we introduce a new benchmarking framework to evaluate GRN inference methods. We will compare it against previously proposed ones and justify its need. The proposed benchmark is designed to allow reproducible research and its provided in a software package that has been published in Biocondutor. A part of this chapter was published in Bellot, P., Olsen, C., Salembier, P., Oliveras-Vergés, A., and Meyer, P. E. (2015). NetBenchmark: a bioconductor package for reproducible benchmarks of gene regulatory network inference. BMC bioinformatics, 16(1):312 (Bellot et al., 2015a)

4.1 Introduction and motivation

In the previous chapter we have seen the importance of GRN inference and some of their proposals. To understand the advantages and limitations of the different network inference methods we need a fair and sound framework to compare networks. In the previous chapter we make use of a framework to perform an evaluation of GRN methods.

But several papers have compared and evaluated different network reconstruction methods (Altay and Emmert-Streib, 2010b; Maetschke et al., 2014; Marbach et al., 2012a; Schaffter et al., 2011; Van den Bulcke et al., 2006). However, in each state-of-the-art study, only one synthetic data generator has been used: either the GeneNetWeaver (GNW) simulator (Schaffter et al., 2011) in (Marbach et al., 2012a) and (Maetschke et al., 2014) or the SynTReN simulator (Van den Bulcke et al., 2006) in (Altay and Emmert-Streib, 2010b). As a result, different conclusions about the best performing methods have been obtained in each study. Finally, most reviews do not study the robustness of the methods and do not evaluate for example the changes of performances of the methods as a function of the number of genes, of the number of experiments or the intensity of noise for multiple simulators and topologies (SynTReN, GNW, E.coli, S. cerevisae, etc.).

Some reviews such as (De Smet and Marchal, 2010) and (Madhamshettiwar et al., 2012) evaluate the behavior of different GRN reconstruction methods in real data corresponding to well-known microbes in (De Smet and Marchal, 2010) and to ovarian cancer cells in (Madhamshettiwar et al., 2012). Although real data represents a theoretically more interesting challenge than artificial data, they suffer from several drawbacks. First, the different algorithms are tested based on only partial knowledge of the underlying network (Roy et al., 2010). In particular, a false positive could be a still undiscovered true positive. Second, the intensity of noise is uncontrollable. Hence, assessing a method's robustness to varying intensities of noise cannot be done easily with real data. However, different noise intensities and distributions are observed from different measurement platforms (i.e. microarray vs. RNAseq) as well as from various organisms. As a result, assessing the performance of any reverse-engineering algorithm on a few real datasets gives little information on its performance on other types of organisms and measurement platforms.

4.2 Benchmarking framework

In this section, we explain with detail a benchmarking frame of reference originally presented at (Bellot et al., 2015a).

As we previously stated, to perform a systematic evaluation of transcriptional network inference methods, and provide a sound and fair comparison of the different GRN inference methods, the need to use *i*n silico data arises and therefore the use of various simulators is essential.

In the design of the benchmarking framework, we have in mind to provide a software tool to the community so it can be fully reproducible and easily modified to change the

experimental setting or introduce a new inference algorithm. Although often overlooked, reproducibility is an important issue in the field of benchmarking. Hence, in order to provide the scientific community with tools allowing the full reproduction of the tests as well as their extension or modification, we provide our benchmarking framework in a tool that was lacking in the community. It is provided in the form of a Bioconductor package (https://www.bioconductor.org/packages/release/bioc/html/netbenchmark.html). Table 4.1 summarizes the most important aspects concerning benchmarking and compares the features included in previously published reviews and the one described here.

Table 4.1 Reviews of GRN reconstruction methods and their characteristics. (Altay2010b): (Altay and Emmert-Streib, 2010b), (Marbach2012): (Marbach et al., 2012a), and (Maetschke2014): (Maetschke et al., 2014).

Review	(Altay2010b)	(Marbach2012)	(Maetschke2014)	This study
Number of variables	100	$\in [1643, 5950]$	$\in [10, 100]$	$\in [300, 2000]$
Topologies	Yeast	E. coli & S. cerevisiae & S. aureus	Yeast & E. coli	Synthetic & Yeast & E. coli
Simulators	SynTReN	GNW	GNW	Rogers & GNW & SynTReN
Number of experiments	$\in [20, 200]$	$\in [160, 805]$	$\in [10, 100]$	$\sim$150
Impact of number of experiments	–	–	✓	✓
Impact of noise	–	–	–	✓
Dataset availability	–	✓	–	✓
Benchmark extension	–	✓	–	✓
Possibility to change parameters	–	–	–	✓

4.2.1 Data generation process

First, we have produced a large set of gene expressions generated by three different simulators and collected them in what we call "Datasource" (see Figure 4.1). This allows having a faster and easiest process as no ODEs have to be computed. Moreover, this makes the reproducibility of the tests much easier, as it is not necessary to interact and parametrise the various simulators (some of them being quite complex). In Figure 4.1, a flowchart illustrates the process applied independently for each one of the five datasources.

At this stage, the data generated by the simulators is free of noise. The noise is added later so that it is possible to control its properties independently of the simulators and also to provide fully reproducible tests. If we observe Table 4.1, this is not possible in any of the other benchmarks.

As we have previously justified, different GRN simulators should be used to perform a proper benchmarking. Therefore, we have generated data with the three different GRN simulators presented in subsection 2.4.2. Using these simulators, five large datasources involving many noise-free experiments have been generated.

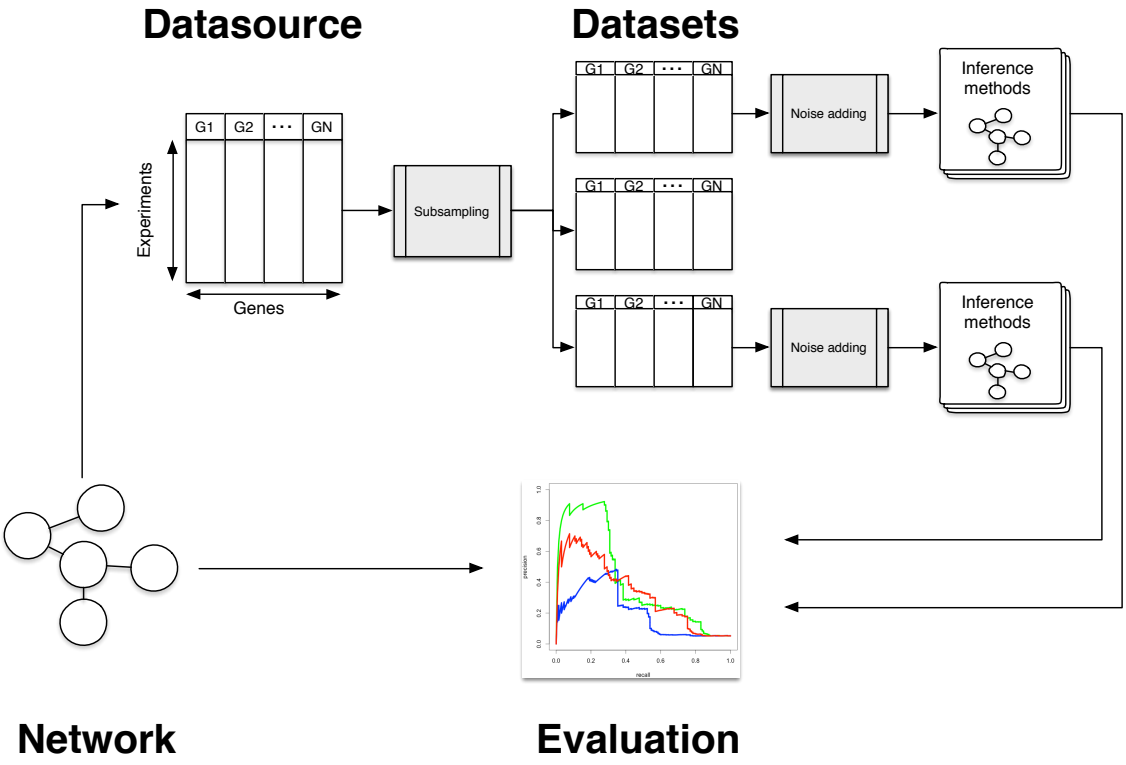


Fig. 4.1 Workflow of the network evaluation process.

The characteristics of these datasources are detailed in Table 4.2. To produce these datasources we have simulated multifactorial data with SynTReN and GNW , which provides less information than extensive knockout, knockdown or time series experiments (Marbach et al., 2010). However, multifactorial data are the most common type of expression datasets because of experimental constraints. Rogers only offers the possibility to simulate knockout data, so in this datasource, we had no choice.

From each one of those datasources we want to generate a large set of different but homogeneous datasets. These datasets are meant to reproduce for example a real microarray

Table 4.2 Datasources used in this study and their characteristics.

Datasource	Name	Topology	Experiments	Genes	Edges
$Rogers_{1000}$	R1	Power-law tail topology	1000	1000	1350
$SynTReN_{300}$	S1	E. coli	800	300	468
$SynTReN_{1000}$	S2	E. coli	1000	1000	4695
GNW_{1565}	G1	E. coli	1565	1565	7264
GNW_{2000}	G2	Yeast	2000	2000	10392

data, each one of these have a smaller number of experiments (or samples) than genes and have different noises and batch effects. To do so, we propose to subsample without replacement the experiments of the datasource randomly and then add the noise.

Each dataset has a different number of experiments extracted from the datasources. In the design we prevent the same experiment to be used several times in the same dataset, but it can appear in different datasets. It is worth noting that because of the high number of samples provided in the datasource, the probability of many identical samples in several datasets is very low in all our tested setups. Each dataset is then contaminated with noise with a slightly different signal-to-noise ratio; this aims to reproduce the variability in the real microarray generation process within the same laboratory or between different ones. In the present study, we have chosen to add a sum of Gaussian noise and lognormal noise to resemble characteristics of the experimental noise observed in microarrays (Stolovitzky et al., 2005). The first noise, called “local" noise is an additive Gaussian noise with zero mean and a standard deviation ($\sigma_{Local(g)}$) that is around a percentage ($\kappa\%$) of the gene standard deviation (σ_g). Therefore, the Signal-to-Noise-Ratio (SNR) of each gene is similar. The local noise standard deviation can be formulated as follows:

$$\sigma_{Local(g);\kappa\%} = \sigma_g \frac{\mathscr{U}(0.8\kappa, 1.2\kappa)}{100}, \tag{4.1}$$

where $\mathscr{U}(a,b)$ denotes the uniform distribution between a and b. This kind of noise will be referred to as local noise.

Additionally, we add an independent lognormal noise called “global” noise in the sequel. The standard deviation of this noise (σ_{Global}) is the same for the whole dataset and is a percentage ($\kappa_g\%$) of the mean variance of all the genes in the dataset ($\overline{\sigma_g}$). It is defined as follows:

$$\sigma_{Global;\kappa_g\%} = \overline{\sigma_g} \frac{\mathscr{U}(0.8\kappa_g, 1.2\kappa_g)}{100}. \tag{4.2}$$

We have chosen to add a range of 40% around κ and κ_g in order to add some variability to the different generated datasets. This range allows the various datasets to have some heterogeneity in noise but ensures at the same time that they are not too different from the originally specified values κ and κ_g. We have chosen this value to reflect our experience with real data. Nevertheless, in addition to this range (40%), we also tested bigger and smaller ranges (60%, 20% and 10%) around κ and κ_g and the conclusions reached by the benchmark are equivalent.

Although no artificial generator is equivalent to real data, an in silico analysis gives reliable guidelines on algorithms’ performance in line with the results obtained from real

data sets (Bansal et al., 2007). Additionally, the use of several different datasources coming from different simulators renders the subsequent analysis of methods more credible before any use on real data.

4.2.2 Network evaluation metrics

The DREAM5 challenge (Marbach et al., 2012a) and its previous editions (Marbach et al., 2010)s have established a de-facto protocol to evaluate an inferred network. The protocol consists of computing the PR or ROC curves, and in measuring the Area under the precision-recall curve (AUPR) or Area under receiver operating characteristic curve (AUROC). This approach gives an estimation of the global behavior of the method.

The DREAM protocol proposes not to take into account predicted interactions that are not part of the Gold standard (GS) (and outside of the set of transcription factors), these interactions are not considered incorrect. The rationale behind this proposal is the use of the same protocol in partially known networks. In this scenario, some real interactions could be missing in the Gold Standard, and therefore it makes sense not to penalise them and to be considered as a potential yet unknown interaction. However, as long as we use *in silico* data with perfectly known Gold Standards we should consider predicted interactions that are not in the Gold Standard as a false positive.

Other papers have evaluated the inferred networks using only the most reliable inferred connections (Marbach et al., 2012a; Roy et al., 2010). The reationale behind all this is that most biologists are mainly interested in a ducked set of interactions that should be tested individually with other expermients. We have adopted this last approach, evaluating the inferred networks using only the top best $x\%$ of the total number of possible connections (if the network has G genes, then the total number of possible connections is $G^2 - G$). This leads to a total of t evaluated connections that will be different for each datasource. We propose to use as performance measures the mean precision, the AUPR and the AUROC in the top best t inferred connections. These measures could be obtained from a directed or undirected evaluation. The former evaluates the existence of an edge and its direction while the latter only evaluates the existence of an edge. In the next section, we will study how a particular measure (undirected AUPR) evolves as the $x\%$ increases. It is worth noting that if a network provides less predicted links than the t evaluated ones, its list is extended with the needed links that are taken randomly from the links that the method puts to zero. This is a de-facto procedure in network evaluation (Marbach, 2009; Marbach et al., 2012a; Prill et al., 2010).

For each datasource of Table 4.2, we generate ten datasets with around 150 experiments. We aim to reproduce typical real microarray datasets that are typically constituted of much fewer experiments than genes. As previously explained, we add two different types of noise: local and global. We propose to add local Gaussian noise around 20 % of the standard deviation ($\sigma_{Local(g);20\%}$, see equation 4.1) and global lognormal noise around 10 % ($\sigma_{Global;10\%}$, see equation 4.2).

Since we generate several datasets for each datasource and use them to evaluate the different methods, for each evaluated method we obtain different metrics for the same datasource, and therefore we can assess the statistical significance of the results. In order to do so, we perform a Wilcoxon Rank sum test with Bonferroni correction (Conover, 1971) for the measures obtained at each datasource.

4.2.3 Additional benchmark

Additionally to the main benchmark, we also analyse the influence of the number of experiments included in the datasets on the different algorithms. This study aims to measure the robustness of the various reconstruction methods regarding the number of available experiments. In a real world scenario, one has budgetary limitations and therefore there is a restriction on the number of different experiments that can be done. Here, we want to address this issue by identifying the best methods in several scenarios with a different number of experiments. To do so, we propose to subsample the experiments of the datasources of Table 4.2 with a different number of experiments and then add local noise of 20 % of intensity. As in the noise sensitivity study, this process is repeated ten times and the performance metrics ($\text{AUPR}_{x\%}$) and is proposed to be average over the different trials.

4.3 Results

In this section, we present the results obtained with of our proposed benchmark that have been obtained with version 1.6 of the Netbenchmark package.

We have proposed to evaluate only the best $x\%$ of the total predictions, so first we will analyse how the $\text{AUPR}_{x\%}$ behaves as a function of x. As an illustration, we show here the curves of only two of the five datasources that come from a network of similar dimensions (S2 and G1). Figure 4.2 and 4.3 shows respectively $\text{AUPR}_{x\%}$ as a function of $x\%$ in datasource S2 and G1.

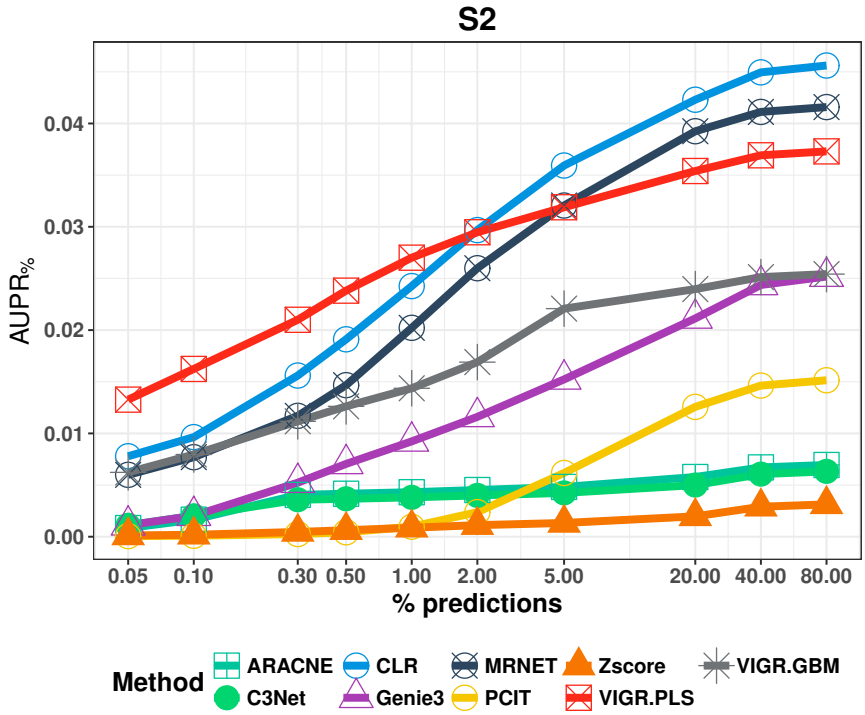


Fig. 4.2 Evolution of $AUPR_{\%}$ of GRN algorithms in S2 datasource.

Observing both figures, we notice that $AUPR_{5\%}$ and $AUPR_{20\%}$ obtains very similar values, so the majority of the accumulated area is achieved with a very little percentage of total links. Therefore, we propose to use $AUPR_{5\%}$. Figure 4.2 and 4.3 also shows how some algorithms behave better than others when we consider a tiny regime of $x\%$. For example, VIGR.PLS is the best performing method in Figure 4.2 when $x < 1\%$. And contrarily, it is difficult to distinguish the best performing method on Figure 4.3 when $x < 1\%$.

As an illustration of the advantage of using $AUPR_{x\%}$ figures instead of actual Precision-Recall curves we show the respective graphs of Figure 4.2 and Figure 4.3 at Figure 4.4 and Figure 4.5. Since the unsupervised GRN inference is a burdensome task, the obtained performances are limited and the different lines of precision-recall are overlapped. Therefore, we propose to use figures of AUPR.

Once we have studied and decided to use the value of $AUPR_{5\%}$ as our metric we generate ten datasets for each datasource of Table 4.2 to perform the benchmarking of the GRN methods listed in Table A.1. Table 4.3 presents $AUPR_{5\%}$ obtained in an undirected evaluation for each datasource. The table also gives the mean and variance across the ten different datasets.

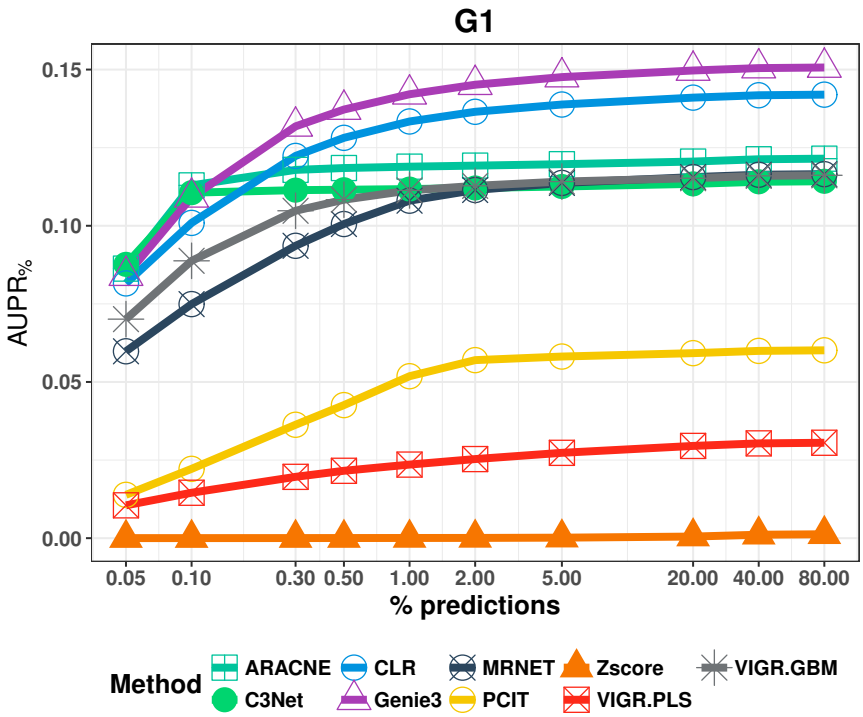


Fig. 4.3 Evolution of AUPR$_{\%}$ of GRN algorithms in G1 datasource.

The results reveal that the studied methods exhibit different behavior across different simulators (and datasources), and none of the methods is the best one for all datasources. We also find significant variations regarding AUPR$_{5\%}$ across datasources: Better results can be expected for smaller networks and simpler simulators such as Rogers. It is worth noting that PCIT, Zscore and VIGR.PLS already start with a high AUPR$_{x\%}$ with tiny values of $x\%$, meaning that they reach a 100 % precision over their most confident connections in the Rogers datasets (see evolution of AUPR$_{x\%}$ in Figure 4.6). The good performances obtained by PCIT and Zscore could be explained because they assume knockout experiments and normally distributed samples, in phase with how the data have been generated (by the Rogers simulator). On the other hand, we think that VIGR.PLS also obtains a good reconstruction because it can decorrelate the irrelevant genes.

None of the methods achieves the best results across the different datasources. But, as a general overview, we can observe that CLR is the best on the majority of the datasets. It is also one of the fastest methods concerning computation time (see Table 4.4).

Differently from (Maetschke et al., 2014), we do not find the Zscore method as the best-performing method. However, there are several aspects to take into account. Our analysis evaluates only the most confident connections returned by the different methods whereas

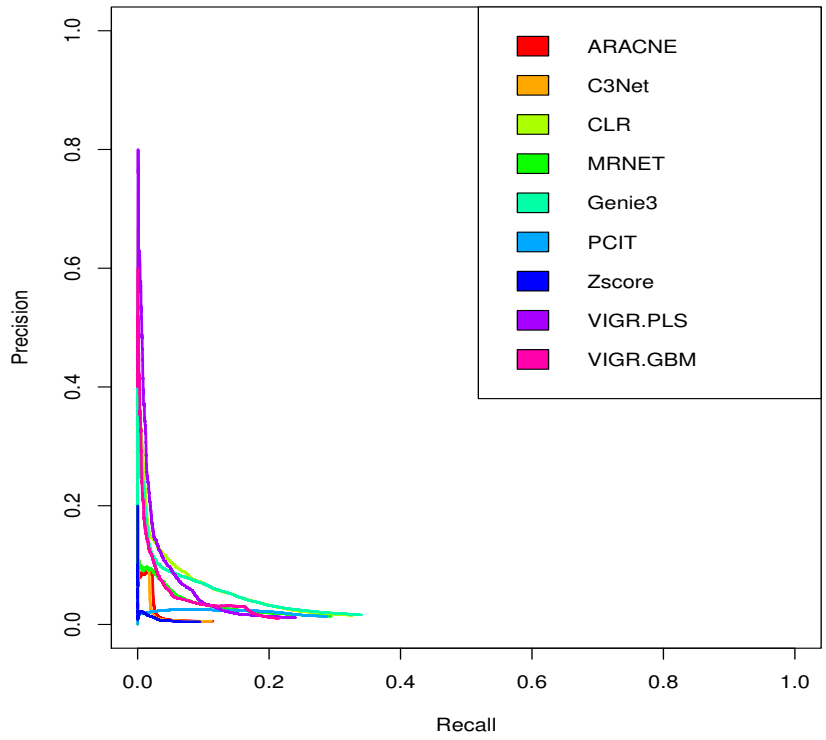


Fig. 4.4 PR curves of selected GRN algorithms in S1.

the study reported in (Maetschke et al., 2014) evaluates all the links. The authors use the AUROC measure that could benefit the sparse recovered networks (Davis and Goadrich, 2006), as is the case of Zscore method. Furthermore, the analysis of (Maetschke et al., 2014) is based on simulation of the fully interventional data, knockouts and knockdowns, of the DREAM4 (Marbach et al., 2010), and only involves the GNW simulator. Nevertheless, we also have evaluated the different reconstruction methods with the same setup as in (Maetschke et al., 2014) and also found that the Zscore is one of the best-performing methods when using knockout data. Therefore, Zscore should only be used to infer networks from datasets obtained with knockout data.

To assess the overall behavior of each method, we propose to aggregate the different performances obtained on the different datasources. But as can be seen in Table 4.3, the $\text{AUPR}_{5\%}$ values have different ranges for each datasource. Therefore, instead of aggregating those values, we aggregate the rank of each method, the smaller the rank, the better the algorithm. Figure 4.7 presents the evolution of the mean value of the rank of the different algorithms across all datasources for different values of $x\%$.

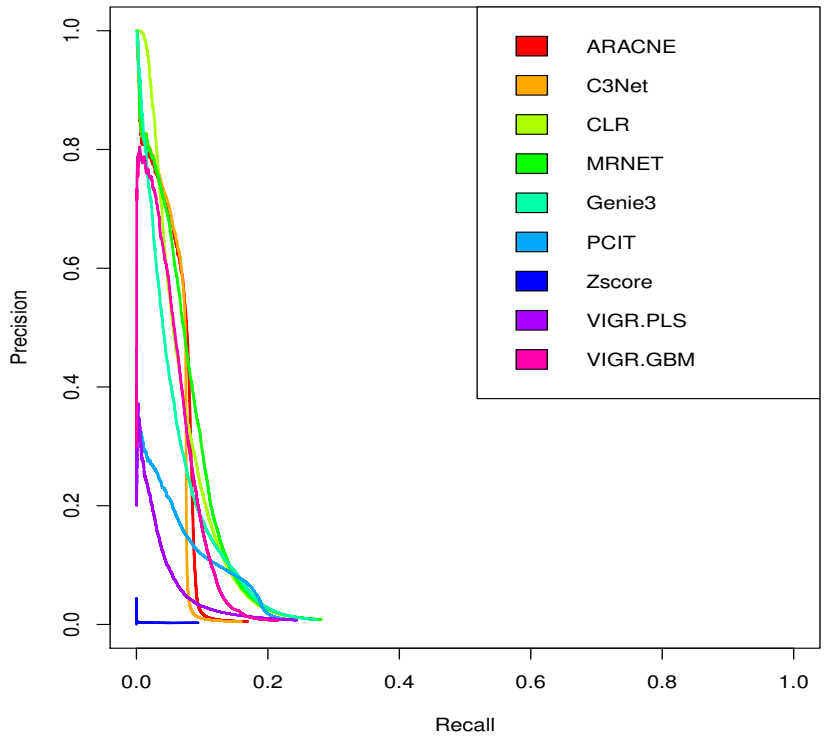


Fig. 4.5 PR curves of selected GRN algorithms in G1.

From Figure 4.7 we can conclude that CLR is the best method for all studied values of $\text{AUPR}_{x\%}$. It is also worth noting that C3Net starts as one of the best performing methods when we only consider the most confident links and becomes the second worst when $x > 5\%$. This behavior can be expected, since the method recovers a very sparse network (or core network) and if we evaluate more and more links they will be randomly chosen edges from the links that C3Net puts to zero. Figure 4.8 presents a boxplot of the rank of $\text{AUPR}_{5\%}$ of the different algorithms across all datasources. For more information on the boxplot, we refer the reader to (Chambers, 1983).

Table 4.3 Performances of the various GRN inference methods on the datasources. AUPR in the top 5 % of the possible connections with a undirected evaluation. The different datasets are contaminated with a 20 % local Gaussian noise and 10 % of global lognormal noise. The best statistically significant results tested with a Wilcoxon test are highlighted for each datasource. Results obtained with current version (1.6) of the package.

		ARACNE	C3Net	CLR	MRNET	Genie3	PCIT	Zscore	VIGR.PLS	VIGR.GBM	Random
S1	Mean	0.044	0.038	**0.137**	**0.115**	**0.137**	0.059	0.033	**0.130**	0.092	0.002
	std	0.012	0.010	0.004	0.015	0.005	0.002	0.023	0.006	0.006	0.000
S2	Mean	0.006	0.005	**0.042**	0.021	0.039	0.013	0.002	0.035	0.024	0.001
	std	0.001	0.001	0.001	0.002	0.002	0.001	0.001	0.002	0.002	0.000
R1	Mean	0.003	0.001	0.005	0.005	0.020	**0.156**	0.123	0.111	0.003	0.001
	std	0.001	0.000	0.001	0.001	0.003	0.016	0.012	0.014	0.001	0.000
G1	Mean	0.121	0.113	**0.141**	**0.150**	0.116	0.059	0.001	0.030	0.115	0.001
	std	0.005	0.005	0.004	0.005	0.001	0.004	0.000	0.001	0.004	0.000
G2	Mean	0.110	0.104	0.099	**0.135**	0.070	0.043	0.000	0.017	0.069	0.001
	std	0.006	0.006	0.003	0.005	0.003	0.003	0.000	0.001	0.003	0.000

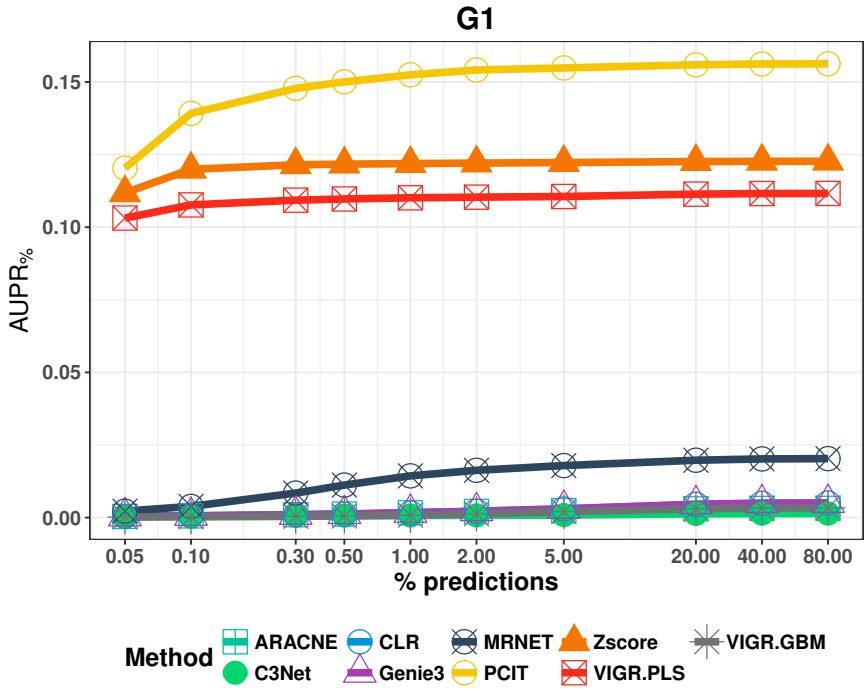


Fig. 4.6 Evolution of AUPR$_{\%}$ of GRN algorithms in R1 datasource.

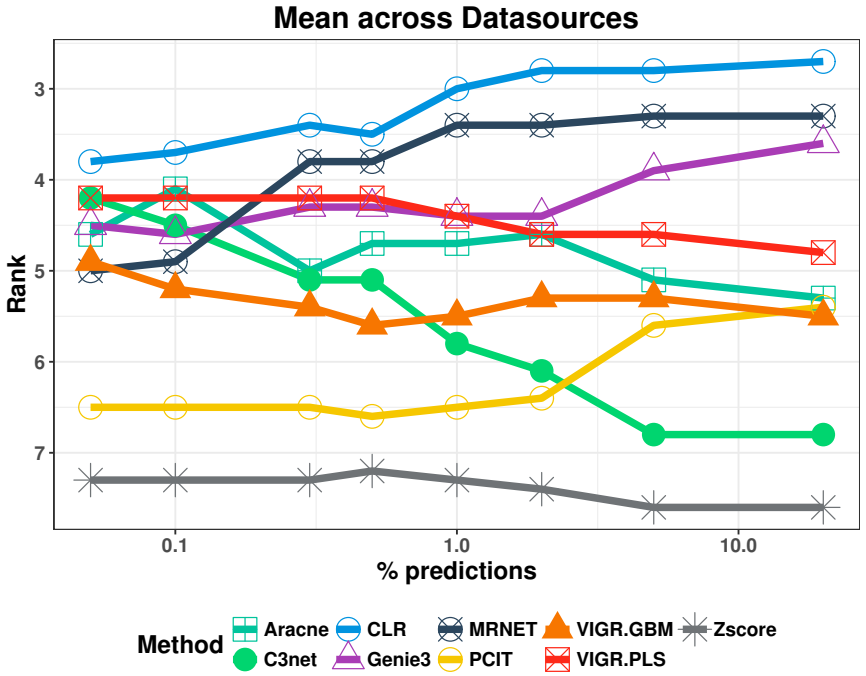


Fig. 4.7 Evolution of Ranking of AUPR$_{\%}$ of GRN algorithms across all datasources.

Table 4.4 Evaluation of the computational complexity. Mean CPU time in seconds of each reconstruction method on the different datasources.

		ARACNE	C3Net	CLR	MRNET	Genie3	PCIT	Zscore	VIGR.PLS	VIGR.GBM
S1	Mean	0.150	0.254	0.055	0.125	263.396	0.368	0.062	217.972	23.496
	std	0.058	0.063	0.003	0.006	53.799	0.005	0.064	48.176	0.927
S2	Mean	2.227	2.078	0.492	7.249	1258.074	12.374	0.176	2221.411	238.015
	std	0.170	0.068	0.061	0.267	206.664	0.217	0.048	432.161	11.163
R1	Mean	2.153	1.720	0.396	5.458	868.189	7.798	0.206	1772.694	200.703
	std	0.129	0.030	0.045	0.126	150.958	0.127	0.010	377.904	5.414
G1	Mean	7.282	4.215	0.932	20.481	2256.310	29.542	0.341	5712.300	553.241
	std	0.381	0.049	0.042	0.317	449.378	0.327	0.067	1342.584	20.595
G2	Mean	19.644	7.627	1.729	53.980	3992.898	95.804	0.611	8485.604	1126.891
	std	0.332	0.250	0.083	0.713	846.123	1.097	0.035	2191.461	43.222

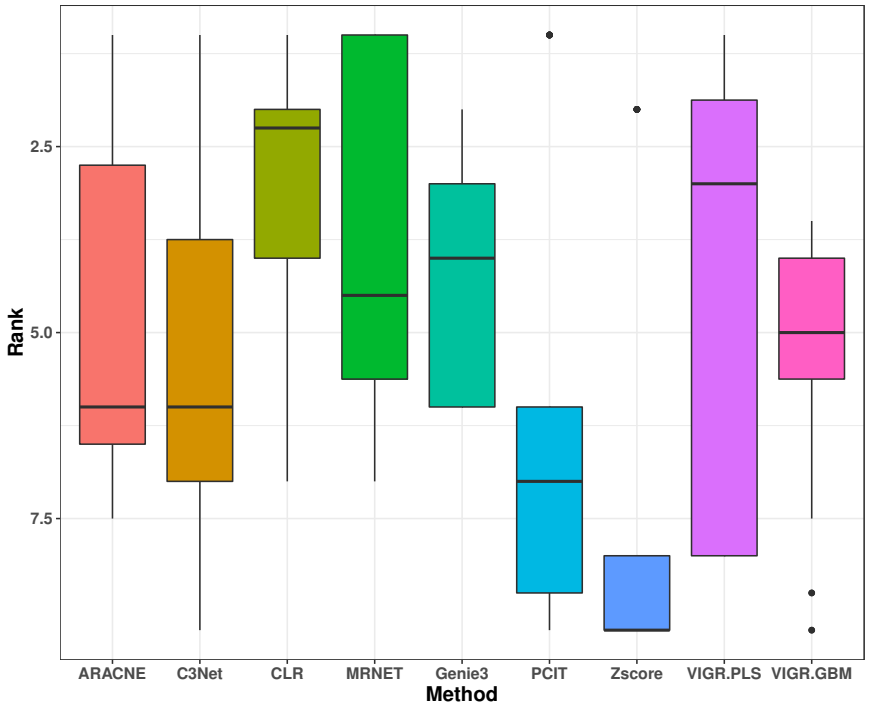


Fig. 4.8 Boxplots of performance. Each box represents the statistics of a method with the ranking performance across all datasources, the smaller the rank, the better. The black line represents the median of the distribution, the box goes from the first to third quartile, while whiskers are lines drawn from the ends of the box to the maximum and minimum of the data excluding outliers that are represented by a point outside the whiskers.

4.3.1 Sensitivity to number of experiments

The aim of this procedure is to measure the robustness of the different reconstruction methods regarding number of available experiments. In a real world scenario, one has budgetary limitations and therefore there is a restriction on the number of different experiments that can be done. Here, we want to address this issue by identifying the best methods in several scenarios with a different number of experiments. To do so, we subsample the experiments of the datasources of Table 4.2 with a different number of experiments and then add local noise of 20 % of intensity. This process is repeated five times and the performance metrics ($AUPR_{5\%}$) are averaged over the different trials.

The results are presented in Table 4.5. Figure 4.9 to Figure 4.10 presents the results for one datasource of each simulator; to have a realistic setting we have chosen datasources that have more than 800 genes and one datasource for each simulator. This study includes different setups, starting with a configuration where the number of experiments is around 1 % of the number of genes and finishing with the same number of experiments of the previous study. We found that increasing the number of samples seems beneficial in most of the methods; it is worth noting that on datasource R1 the performance is outstanding for the Zscore and PCIT methods (see Table 4.5). These results are coherent with a similar study presented at (Maetschke et al., 2014). Note that C3NET and ARACNE methods are the methods that suffer more the effects of a low number of experiments scenario. When few experiments are available, the mutual information values between genes is more difficult to be estimated. The C3NET extracts the maximum value of mutual information per gene, while ARACNE eliminates the edge with the minimum value of MI at every triangle which can explain this result.

Table 4.5 Results of the study on the sensitivity on the number of experiments. Mean AUPR in the top 5% of the possible connections with a undirected evaluation concerning the number of experiments (# exp). The best results are highlighted. Results obtained with current version (1.6) of the package.

Datasource	# exp	Aracne	C3Net	CLR	MRNET	Genie3	PCIT	Zscore	VIGR.PLS	VIGR.GBM
syntren300	20	0.0184	0.0148	**0.0873**	0.0493	**0.0768**	0.0368	0.0128	0.064	0.001
	50	0.0229	0.0170	**0.1025**	0.0623	**0.0935**	0.0349	0.0156	0.091	0.112
	100	0.0289	0.0242	**0.1066**	0.0738	**0.1043**	0.0361	0.0151	0.108	0.096
	150	0.0345	0.0274	**0.1078**	0.0763	**0.1065**	0.0356	0.0450	0.111	0.084
syntren1000	20	0.002	0.002	**0.027**	0.008	0.021	0.007	0.000	0.014	0.000
	50	0.003	0.003	**0.032**	0.011	0.027	0.006	0.001	0.021	0.029
	100	0.005	0.005	**0.036**	0.017	0.032	0.007	0.001	0.028	0.027
	150	0.004	0.004	**0.036**	0.014	0.031	0.006	0.004	0.031	0.021
rogers1000	20	0.000	0.000	0.000	0.000	0.000	**0.018**	**0.018**	**0.011**	0.000
	50	0.000	0.000	0.001	0.001	0.003	**0.051**	**0.043**	0.034	0.000
	100	0.002	0.001	0.002	0.002	0.012	**0.114**	0.091	0.078	0.001
	150	0.004	0.002	0.004	0.004	0.021	**0.169**	0.133	0.121	0.002
gnw1565	20	0.0130	0.0101	0.0184	0.0149	0.0134	**0.0321**	0.0001	0.0028	0.0001
	50	0.0474	0.0433	**0.0711**	0.0596	**0.0630**	0.0486	0.0001	0.0087	0.0411
	100	0.0888	0.0839	**0.1163**	**0.1149**	0.0950	0.0553	0.0002	0.0173	0.0881
	150	0.1136	0.1064	**0.1352**	**0.1407**	0.1101	0.0599	0.0001	0.0257	0.1079
gnw2000	20	0.0106	0.0083	0.0196	0.0141	0.0111	**0.0227**	0.0001	0.0024	0.0001
	50	0.0373	0.0352	**0.0572**	**0.0545**	0.0394	0.0337	0.0001	0.0063	0.0288
	100	0.0835	0.0796	0.0842	**0.1089**	0.0609	0.0399	0.0001	0.0109	0.0536
	150	0.1089	0.1038	0.0966	**0.1335**	0.0696	0.0431	0.0001	0.0157	0.0684

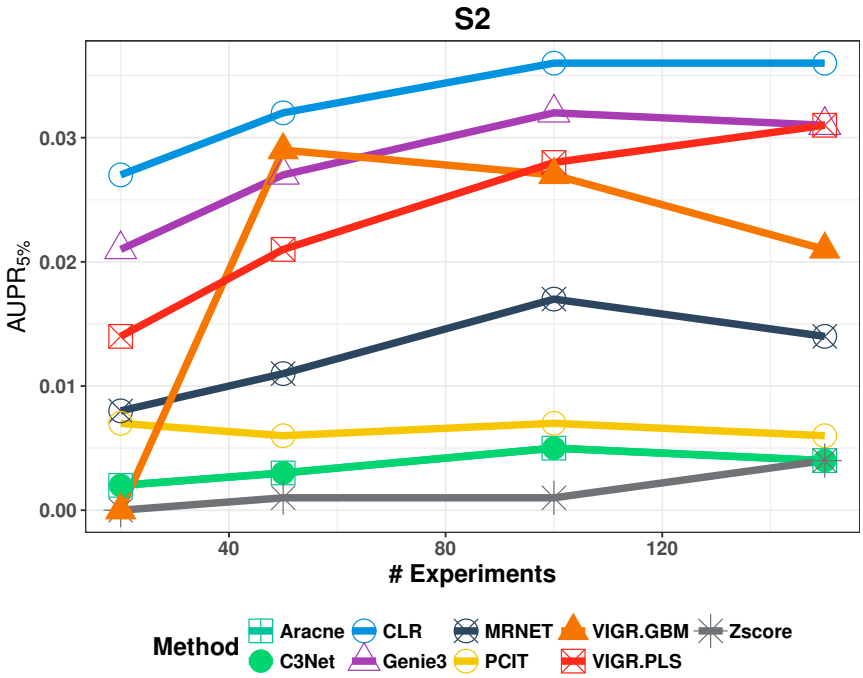


Fig. 4.9 Plots of mean performance of selected GRN algorithms with different number of experiments in S2.

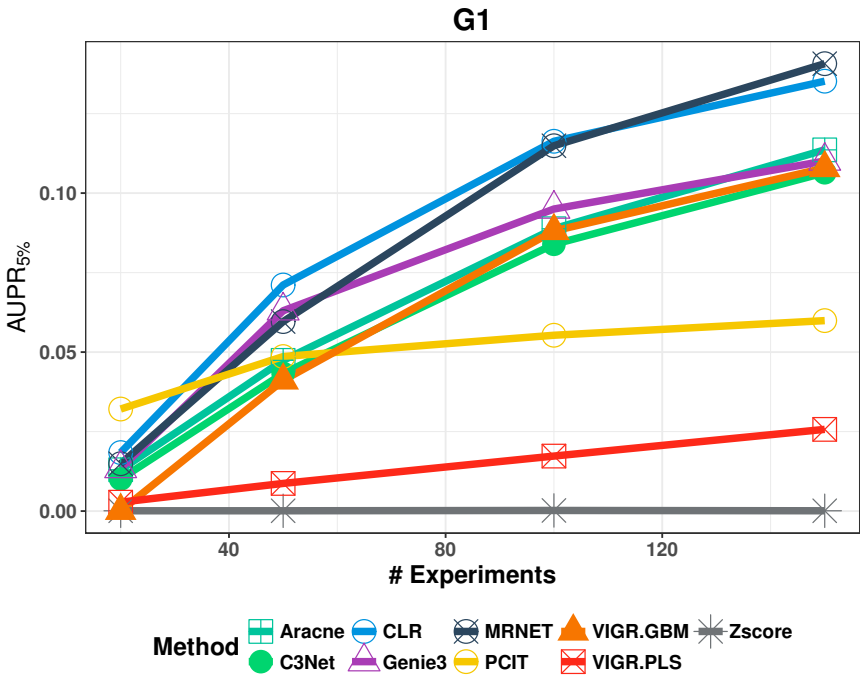


Fig. 4.10 Plots of mean performance of selected GRN algorithms with different number of experiments in G1.

4.4 Network evaluation with topological features

The previous analysis represents the first step for a network inference method to demonstrate its ability to recover regulatory networks from a broad set of realistic artificial datasets, where the truth is known and where the noise is controlled. A second step can address some topological evaluation of the networks. This previous evaluation based on classification of links explained in subsection 2.5.1 is very important but it may be interesting to go beyond. In this section, we will present another measure that consider the topology to perform network comparison.

Other authors have studied how to perform this network comparison based on a measurable and quantitative description of the main properties of a graph. The first approach to resolve if two graphs have similar topologies consists determining whether the two graphs are isomorphic, this is known as graph isomorphism problem which is believed to be in the class NP (Schöning, 1988) and therefore impractical. As a major alternative strategy, one can define a suitable similarity measure on the topology of the underlying (possibly directed and/or weighted) graphs. This line of study dates back to the 70s with the theory of graph distances, regarding both inter- and intra-graphs metrics (Entringer et al., 1976).

Here, we mention some of the most important metrics coming from different approaches: the family of edit distances, evaluating the minimum cost of the transformation of one graph into another inserting and deleting links (edit operations); the family of common network subgraphs, which looks for shared structures between the graphs, and the family of spectral measures, which relies on functions of the eigenvalues of the graph matrices. However, in general, the most efficient techniques use a kernel machine, using an implicit mapping of the graph into a high dimensional feature space defined by a kernel function. In practice, this methods compare substructures of graphs that are computable in polynomial time (Schölkopf et al., 2004; Vert and Kanehisa, 2002; Vishwanathan et al., 2010).

4.4.1 Graphlet Degree Distribution Agreement (GDDA)

Graphlets were introduced in (Pržulj et al., 2004) as small connected non-isomorphic induced subgraphs of a larger network. They have been used to define several other graphlet-based network properties to perform Biological network comparison (Pržulj, 2007). To perform this comparison the author define a generalisation of degree distribution. If the degree distribution measures the number of nodes with k edges, the Graphlet Degree Distribution (GDD) measures the number of nodes that are present k times within a particular graphlet.

A graphlet is a small connected non-isomorphic subgraphs of a large network. Figure 4.11 presents the graphlets up to 5 nodes.

An edge is the only graphlet with two nodes denoted as G_0 in Figure 4.11. Therefore, with the graphlet G_0 we can obtain the degree distribution: If we count how many nodes are involved in only one G_0, how many nodes are involved in only two G_0, and continue to count how many nodes are involved in only k G_0s and then divide those numbers by the total number of nodes we get the degree distribution, $P(k)$ (defined in subsection 2.3).

So, the rationale of GDD is to extend this procedure to all the other graphlets. But as we start to take into account bigger subgraphs the need to take care of the topology arises. For example, for graphlet G_1, to count the nodes that are involved in k G_1 we should distinguish between nodes at an end of G_1 or at the central node. This happens because G_1 admits an automorphism. The nodes of the ends belong to the same *automorphism orbit* (or just *orbit*) and the middle one belongs to another orbit. Such nodes in the same orbit can be permuted, and we would obtain the same graph, being, therefore, an automorphism. Figure 4.11 presents the different *orbits* groups for the graphlets up to five nodes with 73 different orbits.

Finally, we can obtain the GDD counting the number of nodes involved in a particular orbit. Therefore, the GDD is the distribution of an orbit. Hence, if we consider graphlets up to 5 nodes, we will get 73 GDD's. In this way, we obtain 73 distributions analogous to the degree distribution. The degree distribution is the first of the 73 GDD's. Each GDD measure different local structural properties of the network and characterise the graph.

In order to compare two networks Γ_1 and Γ_2, (Pržulj, 2007) propose a way to compare these 73 distributions of both networks an a way to "reduce it" into a single number. The $d^o_\Gamma(k)$ counts how many nodes are involved k times in the orbit o in graph Γ. In order to compute the distribution it is scaled as: $S^o_\Gamma(k) = d^o_\Gamma(k)/k$.

This distrubution is further normalised to to decrease the contribution of larger degrees with respect to its total area giving the "normalized distribution":

$$\nu^o_{\Gamma_1}(k) = \frac{d^o_\Gamma(k)/k}{\sum_{l=1}^{\infty} \Gamma^o(l)/l} \tag{4.3}$$

Finally, it is possible to define a "distance" $D^o(\Gamma_1,\Gamma_2)$ between their normalized distributions as:

$$D^o(\Gamma_1,\Gamma_2) = \frac{1}{\sqrt{2}} \left(\sum_{k=1}^{\infty} \left[\nu^o_{\Gamma_1}(k) - \nu^o_{\Gamma_2}(k) \right]^2 \right)^{1/2} \tag{4.4}$$

where the $\frac{1}{\sqrt{2}}$ ensures the distance is always < 1 (Pržulj, 2010).

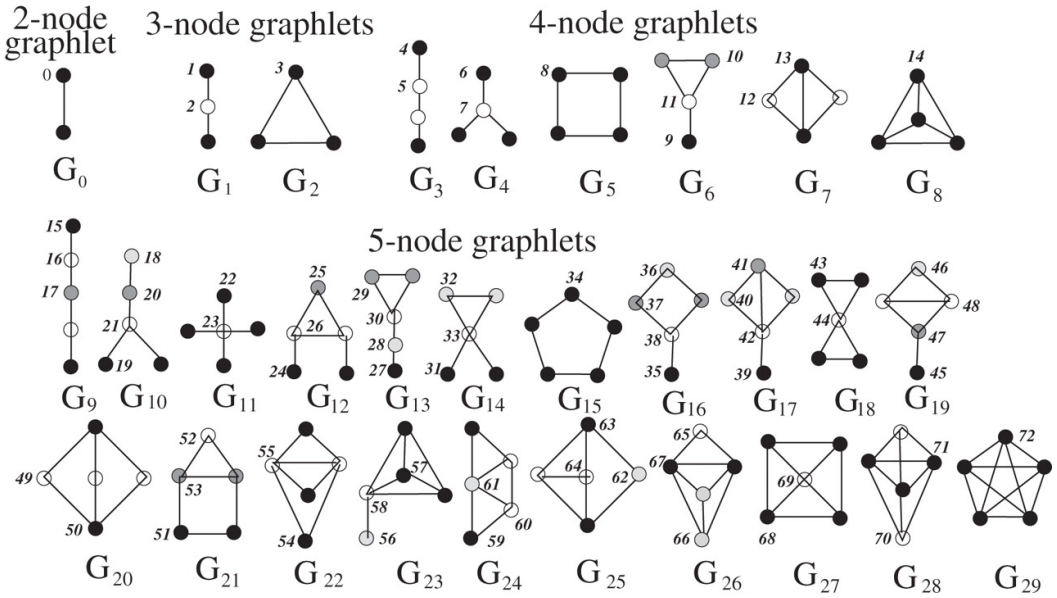


Fig. 4.11 The 73 automorphism orbits for the graphlets up to 5 nodes. In a particular graphlet G_i, $i \in \{0,1,\ldots 29\}$, nodes that belongs to the same orbit have the same gray color. Figure taken from (Pržulj, 2007).

Finally, to get an "agreement" instead of a distance a complement in positive space is computed $(1-D^o(\Gamma_1,\Gamma_2))$, and the Graphlet Degree Distribution Agreement (GDDA) is obtained as the mean of these 73 values:

$$GDDA(\Gamma_1,\Gamma_2) = \frac{1}{73}\sum_{o=0}^{72}(1-D^o(\Gamma_1,\Gamma_2)) \tag{4.5}$$

With this definition, if two networks have a GDDA close to 1 then they have a similar topology in the sense that their GDDs are similar in the proportion of their network size.

4.4.2 Results of topological similarity evaluation

To provide this second step we evaluate the networks using the topological measure introduced at section 4.4. We also have used the predicted networks with up to 5% of the maximum predictions and compared them with the Gold standard (GS). Figure 4.12 presents the distribution of the GDDA measure in the form of boxplots for all different datasources. GDDA measures the similarity of the degree distribution of graphlets and provides a tool for biological network comparison (Hayes et al., 2013). We can see that the models recovered in the datasources generated by GNW (G1 an G2) are better than the ones obtained at other datasources. This can happen because data generated by GNW is easier than SynTReN.

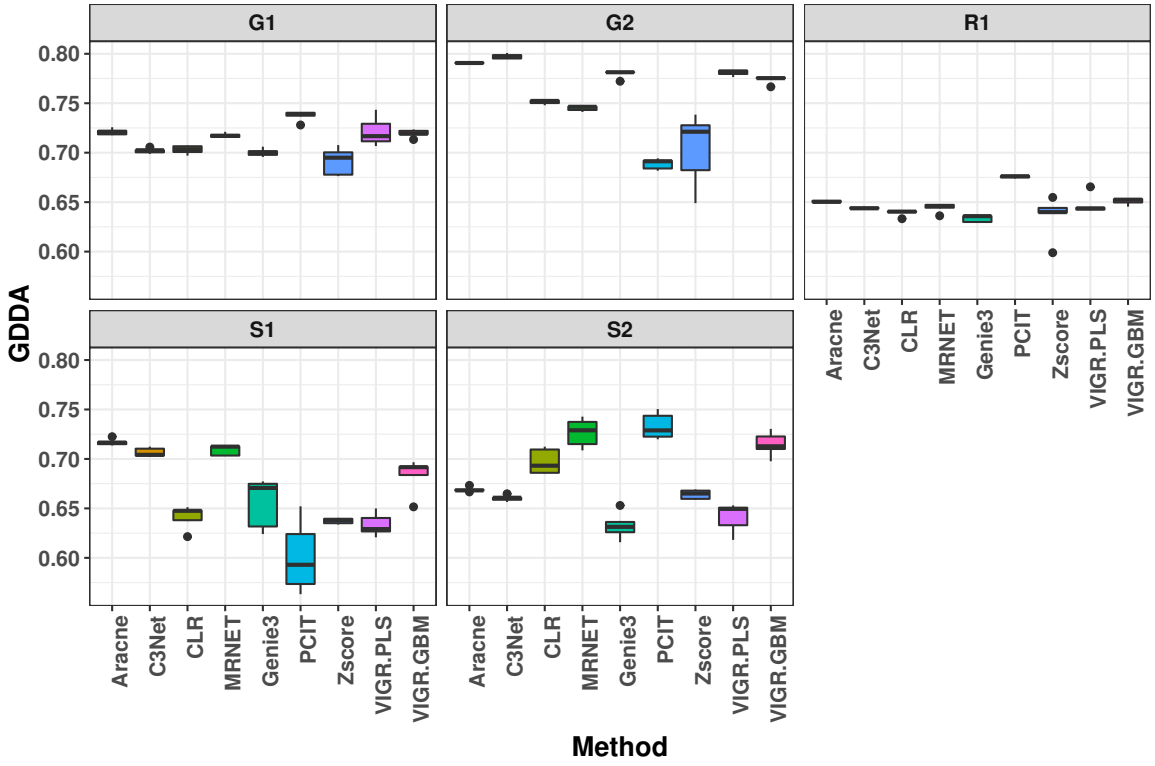


Fig. 4.12 Boxplots of GDDA measure per datasource.

Mean values of $AUPR_{5\%}$ in Table 4.3 are smaller for SynTReN than for GNW datasources which could have influence on the topological quality of the networks. It is worth noting that the networks recovered at datasource R1 are the worst ones even for good performance methods in this datasource (PCIT and Zscore). This can be caused by the pure Power-law topology of the network.

Figure 4.13 presents the distribution of GDDA for all datasources. We can see that the various algorithms recover similar models, the median of the GDDA that all the analysed method obtains lie between 0.65 and 0.725. However, we can observe that Zscore and VIGR.PLS seems to recover a network with less similar topological properties. We also highlight that MRNET seems to obtain a more similar topology to the Gold standard (GS). We think that this is caused by the MRMR feature selection algorithm. Since it allows the natural appearance of some gene with many connections and therefore permitting a scale-free topology.

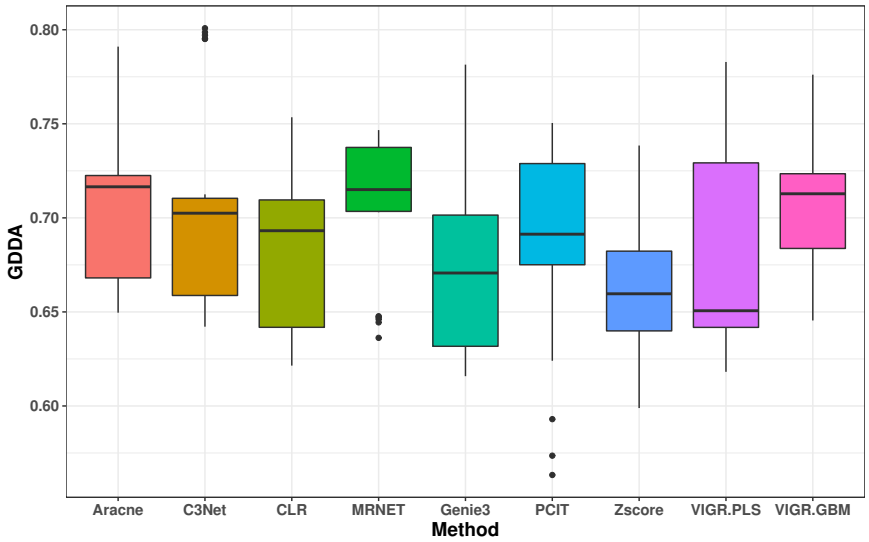


Fig. 4.13 Boxplots of GDDA measure for all datasources.

4.5 Implementation

NetBenchmark is a Bioconductor (Gentleman et al., 2004) package. As a result, the code is written primarily in R[1]. This allows the present review to be fully reproducible, with one function call. The package imports several CRAN and Bioconductor packages. Most of those provide competitive network inference methods that are used in our benchmark. The pipeline starts with a set of noise-free datasources coming from different GRN simulators that have been previously generated for this package. The datasources are stored in grndata package[2] and are loaded automatically as input. These datasources are subsampled and contaminated with noise to generate datasets with enough variability to provide an informative and thorough comparison of GRN inference methods.

The package allows the user to reproduce and even to modify the experiments reported in this document. However, a significant additional functionality is that it also allows new methods to be evaluated. In the current version of the netbenchmark package (1.6), it is possible to evaluate new unsupervised network inference methods. The method should infer the network from steady-state expression data and should be able to perform this task with a fewer number of experiments than the number of genes. The last requirement is that the provided method is able to infer networks with thousands of genes. To benchmark a new

[1]R Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria (2015).

[2]Bellot, P., Olsen, C., Meyer, P.E.: grndata: Synthetic Expression Data for Gene Regulatory Network Inference. (2014). R package version 1.6.0.

Listing 4.1 Roadmap to compare a new method with the state-of-the-art ones with netbenchmark

```
# Define a wrapper function (as an illustrative example)
fun ← function(data){cor(data)}
# Compare it with the state-of-the-art:
top5.aupr ← netbenchmark(methods=c("all","fun"), no.topedges = 5)
print(top5.aupr[[1]])
```

method, a new wrapper has to be defined: `fun(data)`. This function receives a numeric `data.frame` with the gene expression data in the argument data where the columns contain the genes and the rows the experiments. The function should return a matrix which is the weighted adjacency matrix of the network inferred by the algorithm. To benchmark this method against all the other algorithms of the package the procedure is shown at Listing 4.1.

For more information about the use of NetBenchmark package we provide a quick tutorial in section A.2. This tutorial allows unlocking the full set of functionalities of the package including the ability to add new methods in the benchmark.

4.6 Conclusions

We have presented a benchmark process for network reconstruction algorithms that relies on data produced by several in silico generators and a subsampling strategy to generate an environment for evaluating the different methods, in a fast and robust way. This benchmark is focused on (but not limited to) a GRN reconstruction task and therefore we have taken into account the goals of the community such as the evaluation of the most confident connections.

This framework is provided as a tool that was lacking in the community in the form of a R/Bioconductor package that allows full reproducibility, with one function call of the Bioconductor package NetBenchmark. With this package, the different datasources are automatically loaded and some of the most used GRN inference methods are implemented or imported with the package.

Our proposed setup, described in the previous, compares ten variations of five datasources having more than 100 expression-measurements each. In other words, it will compare methods on 50 datasets, each with very different samples and even different amounts of noise. Using realistic artificial data allows for large number of samples that in turn, allows for reliable statistical measures indicative of performances and robustness. So far, no consortium nor database focusing on real data has assembled several thousands of homogeneous

expression samples (coming from the same experimental platform) that would allow for a similar benchmark.

We have argued that the first step to support a new network inference method is to demonstrate its ability to recover regulatory networks from a broad set of realistic artificial datasets, where the truth is known and where the noise is controlled. Then, of course, a second step would be the analysis of the algorithms on real data (for example, coming from model organisms), and possibly the addition of some topological evaluation of the model.

As the main conclusion in terms of specific algorithm, we can highlight that none of the methods obtains the best results across the different datasources. In the following chapter we will present different alternatives to improve the quality of the inferred networks and analyze how to aggregate various network inferences through a consensus mechanism. We will propose a common framework to standardise already proposed consensus methods.

Chapter 5

Meta-networks and post-processing methods

Abstract

Inferring gene regulatory networks from expression data is a tough problem that has been tackled in different ways. The different algorithms have some particular biases and strengths, and none of them is the best across all types of data and datasets. In this chapter, we will present different approaches to attempt to improve the inferred network. First, we will present some post-processing methods. Second, we will introduce the idea of aggregating various network inferences through a consensus mechanism. We present a framework to standardise already proposed consensus methods, and based on this framework different proposals are introduced and analysed in different scenarios. A part of this chapter was published in Bellot, P., Salembier, P., Oliveras-Vergés, A., and Meyer, P. E. (2015). Study of Normalisation and Aggregation Approaches for Consensus Network Estimation. In 2015 IEEE Symposium Series on Computational Intelligence, number 1, pages 1081-1086 (Bellot et al., 2015b). Moreover, also: Pham, N. C., Haibe-Kains, B., Bellot, P., Bontempi, G., and Meyer, P. E. (2016). Study of Meta-Analysis Strategies for Network Inference using Information-Theoretic Approaches. Biological Knowledge Discovery and Data Mining. (Pham et al., 2016)

5.1 Introduction and motivation

To infer a GRN from data is a challenging task. All the methods presented in Chapter 3 face some challenges, that includes among others the non-linearity of regulatory relationships among genes, the incompleteness and noisiness characterising genomic data, the presence of a relatively small number of samples compared with the number of genes (the "large p small n problem" (Hastie et al., 2009)).

GRN inference methods tend to recover indirect regulatory relationships. For example, if gene A regulates gene B and this last one regulates gene C. Many algorithms will find a relationship between gene A and gene C even though it is an indirect effect.

Moreover, mixed regulatory interactions represent also a very difficult task. As for another example, if two genes D and E regulates gene F but with opposite effect (one activates the gene, and the other represses it). It is very likely to find any other gene that have a greater similarity to F's gene expression rather than D and E. Therefore, many methods will infer a false relationship instead of true ones. In this sense, (Schaffter et al., 2011) made an analysis of different network topologies (motifs) that are commonly inferred incorrectly. Furthermore, (Krishnan et al., 2007) pointed that some particular topologies are impossible to be inferred from gene expression data without any further information (knockdowns, existing interactions, etc).

5.2 GRN post-processing methods

To deal with these false recovered relationships, different post-processing methods have been proposed.

Silencing indirect relationships

In (Barzel and Barabási, 2013), the authors propose a method that tries to eliminate indirect relations. This method is an application of an experimental perturbation model proposed at (Barzel and Biham, 2009). Starting from the premise that the system of genes is in equilibrium, a model for the change in expression of one gene in response to a small perturbation in another is proposed.

The authors define the local response matrix as $S \in \mathbb{R}^{N \times N}$. While A is inferred adjacency matrix that contains indirect relations, the local response matrix measures the existence of direct links. Therefore, $s_{ij} \geq 0$ implies a direct link between gene i and j, while $a_{ij} \geq 0$

can be caused by an indirect relationship. The goal of the method is to recover S from A, obtaining the following approximated relationship:

$$S \approx (A - I + \mathscr{D}((A-I)A))^{-1} \tag{5.1}$$

where I is the identity matrix, A the estimated adjacency matrix that contains all (including indirect ones) and $\mathscr{D}(M)$ sets the off-diagonal terms of M to zero.

Network deconvolution

In (Feizi et al., 2013) a method called network deconvolution is presented, in their words, the method provides "... a systematic method for inferring the direct dependencies in a network, corresponding to true interactions, and removing the effects of transitive relationships that result from indirect effects."

In their model, an unknown matrix A_{dir} (that contains only direct relationships) that have its eigenvalues between -1 and 1 is obtained from an observed matrix A_{obs}. Both matrices are related though the following expression:

$$A_{\text{obs}} = A_{\text{dir}} + A_{\text{dir}}^2 + A_{\text{dir}}^3 + \ldots = A_{\text{dir}}(I - A_{\text{dir}})^{-1} \tag{5.2}$$

Note that A_{dir}^l reflects the number of walks of length l in A_{dir} from g_i to g_j in the position (i, j) of the matrix A_{dir}^l. Finally, the network deconvolution is obtained manipulating the previous equation to obtain:

$$A_{\text{dir}} = A_{\text{obs}}(I + A_{\text{obs}})^{-1} \tag{5.3}$$

Unfortunately, many matrices cannot be decomposed as an infinite sum of powers of some matrix as in equation 5.2. So equation 5.3 cannot be applied directly to arbitrary data matrices, but "can be achieved for any matrix by scaling the observed input network by a function of the magnitude of its eigenvalues."

The authors propose to use a scaling factor close to one, to consider a higher order of indirect interactions and as a result obtain the best performance.

Re-ranking predictions: Netter

In (Ruyssinck et al., 2016) a post-processing algorithm called Netter is proposed. It changes the rank of the predicted edges of the inferred network. It tries to improve the structural

properties (based on graphlets (Pržulj, 2007)) of the final network to resemble those typically found in a gene regulatory network.

The algorithm reorders only the top x links. Each ranking has an assigned cost and using simulated annealing (Hwang, 1988) this cost is minimised several times, obtaining different re-ranked lists. These lists are averaged to get the final output ranking. The cost function penalises the modification of the original ranking and rewards better structural properties.

Comparison of post-processing methods

According to the experiments provided at (Ruyssinck et al., 2016), the Netter algorithm is the only post-processing method that can systematically improve networks. The authors found that Network Deconvolution results in a decrease in AUROC and AUPR in all but a few cases. On the other hand, Silencer can increase AUROC score in some cases but does not have a positive effect in all cases. This finding is in line to some controversy of the performance of the Silencer and Network Deconvolution methods (Bastiaens et al., 2015; Pachter, 2014).

As another advantage of Netter compared to other methods, it is not limited to correlation-like measures, and it can be applied to rankings or ranking ensembles of different algorithms.

5.3 Meta-network algorithms

Many GRN inference methods have been proposed to try to recover the regulatory network from gene expression data. We presented some of the most important ones in subsection 3.1. Several network inference methods have been compared in Chapter 4. The main conclusion of the chapter and other studies (Marbach et al., 2012a) is that no single inference method performs optimally across all datasets and each GRN inference method returns a model that is different and has some strengths and biases.

From those observations comes the idea of combining multiple network inference algorithms. This could be a good strategy to infer an accurate and comprehensive GRN thanks to the meta-network algorithms that combine several methods. This approach is receiving more and more attention from the scientific community.

To this aim, different alternatives have been proposed. They belong to three broad categories: data merging, pairwise ensemble and network ensemble.

In the data merging approach, datasets are integrated at the expression level into a unique dataset. This process tries to improve the quality and increment number of experiments in the data. From this unique dataset, a GRN is inferred (Adler et al., 2009; Belcastro et al.,

2011; Huber et al., 2002). However, this simple strategy presents some problems such as the removal of unwanted biases from data (batch effects) that can lead to incorrect conclusions. Some of the classical methods to remove this batch effects are BMC (Johnson et al., 2007) and COMBAT (Sims et al., 2008). We will not deal with this strategy since it is more a data-merging problem.

Instead of mixing the raw data, the pairwise ensemble matrices approach aggregates a comparable feature for each dataset in the form of pairwise measures between variables. This pairwise measures usually consists of mutual information or correlation matrices.

Finally, networks ensemble combines different inferred networks to produce a so-called community or consensus network. In the following subsections, we will explain these two families in more detail.

5.3.1 Assembling pairwise matrices

In (Pham et al., 2016) a meta-analysis approach for inferring GRNs from multiple studies is presented. The method is adapted to methods based on pairwise measures such as correlation or mutual information and consists of two steps: aggregating matrices of the pairwise measures from every dataset followed by extracting the network from the meta-matrix.

The proposed method aggregates mutual information matrices rather than data or networks. The idea behind assembling pairwise matrices is that, although expression data typically shows high variability due to differences in technology, samples, labels, etc., pairwise dependency measures between genes should be much less variant (i.e. dependent variables, such as a regulating variable and its regulated counterpart, should remain dependent in every platform/experiment/dataset even if ranges of values differ significantly). Thus, to infer a network from various expression data, the approach consists in combining mutual information matrices (MIMs) estimated independently from each dataset. Then, a GRN network is inferred from the aggregated MIM using one of the Information-theoretic based GRN inference methods (see subsection 3.1).

5.3.2 Network consensus

In this category, every single network is constructed independently, and we aim to combine them. In general, combining networks consists of two distinct steps: transformation and aggregation, which will be covered later in this document (see section 5.5). In the next subsections, we present two state-of-the-art consensus methods.

5.3.2.1 Wisdom of crowds

In (Marbach et al., 2012a), the authors propose an algorithm to integrate the predictions of different network inference methods to construct a community network by re-scoring interactions according to their average rank across all methods.

Various applications of the same network reconstruction algorithm inferring a given network can create lists of predictions in which the same interaction I can be placed in various rank positions. The position will depend on the set of experiments used by the algorithm, random aspects of the method itself, biological variability, etc.

The question remains as to how to aggregate the K lists into one final ranked list. This issue is discussed in (Haury et al., 2011; Schimek et al., 2015).

In (Marbach et al., 2012a), a method based on rank averaging is proposed: The individual ranks for each link are added together to compute the final rank. The final list is calculated decreasingly sorting these scores. This method is also known as *Borda* count or RankSum and will be referred as RankSum in this thesis.

5.3.2.2 Top*k*Net

In (Hase et al., 2013), the Top*k*Net algorithm is presented. Its underlying idea is that there exist some optimal algorithms for a given expression data and the integration of these optimal algorithms may be a more robust strategy to reconstruct accurately GRNs than using all the algorithms.

The algorithm generates a new prediction list through the combination of the individual network inference algorithms. Top*k*Net applies a bagging method, introduced by (Breiman, 1996), to combine confidence scores between each gene pair from multiple network inference algorithms.

The parameter k controls the number of algorithms that are taken into account, for example, if we have a pool of 20 individual network inference algorithms, the maximum value of k is 20. Setting $k = 1$ the integration prediction will define an edge between two genes if at least one ($k = 1$) of the individual algorithms assigns a high confidence level to the link between them. By contrast, the integrated list obtained by Top20Net will have an edge between two genes only if all the 20 individual algorithms assign a high confidence level to the link between them.

5.4 Data heterogeneity

It is possible to reconstruct GRN from different kind data. As an example a physical, regulatory network is one where edges represent a physical interaction between a TF and a target as detected in Chromatin Immunoprecipitation (ChIP) assays or predicted using sequence-based DNA binding models (regulatory motifs). However such edges may not necessarily lead to functional changes in gene expression. In contrast, a functional regulatory network is one where edges between TF's and their targets are supported by functional changes in the gene expression (Capaldi et al., 2008), but as we already stated these relationships might be indirect.

From those observations comes the idea of combining both physical and functional evidence among others to further improve the inferred network. However, the nature of such networks are very different, and the recovered links are very different for each data. We will refer to this scenario as *Heterogeneous* scenario. This situation reflects the case where the individual networks have been inferred from very different kind of data, and the individual networks are very different, hardly having any edge in common.

In contrast, the *Homogeneous* scenario reflects the case where the individual networks have been inferred with different algorithms but using the same type of data like gene expression which leads to rather more similar individual networks (which is the case of subsection 5.3).

In (Marbach et al., 2012b) the authors tackle this problem using both supervised and unsupervised methods to predict regulatory edges by integrating datasets as input features. The unsupervised method consists on averaging of the links across different dataset networks. The final list is computed sorting these score decreasingly. This method will be referred as IdSum in this thesis.

5.5 Consensus network as Normalisation and Aggregation

Hereafter, we will restrict to network ensemble methods. We argue that this strategy is the most general one, since the networks may come from different kind of data. Furthermore, this strategy can be even applied when the networks are constructed from Chromatin Immunoprecipitation (ChIP) data or even from the literature, see section 5.4. In this section, we will describe a common framework to standardise already proposed consensus methods, moreover, based on this framework different proposals are introduced.

The consensus network estimation can be seen as involving two distinct steps. The first one transforms the different network scores, $s_n(e_{ij})$, in order to have a common scale or distribution. This process is referred to as “Normalisation". For example, (Hase et al., 2013; Marbach et al., 2012a) use *Rank*, whereas (Marbach et al., 2012b) does not perform any normalisation which can be seen as the *Identity* normalisation.

Then, the second step is the “Aggregation" of the N different edge scores into one consensus score for every edge. In (Marbach et al., 2012a) and (Marbach et al., 2012b) this process is done through the *Sum* process, while (Hase et al., 2013) uses a rank order filter.

We now extend this idea to propose new algorithms. First, different Normalisation options are presented, and then the distinct Aggregation proposals are discussed. Finally, their combination will lead to different consensus network algorithm proposals.

5.5.1 Normalisation

Four different normalisation techniques will be analyzed. Let us call $t_n(e_{ij})$ the normalised value assigned to edge e_{ij} for the network n.

Identity

The *Identity* does not apply any transformation to the original scores of the inferred network:

$$t_n(e_{ij}) = s_n(e_{ij}) \tag{5.4}$$

Rank

The *Rank* replaces the numerical score $s_n(e_{ij})$ by its rank $r_n(e_{ij})$ such as the most confident edge receives the highest score. The *Rank* method preserves the ordering of the scores of inferred links but the differences between them are lost:

$$t_n(e_{ij}) = r_n(e_{ij}) \tag{5.5}$$

Scale

A classical normalisation of a random variable involves a transformation to remove the effect of the mean value and to scale it accordingly to its standard deviation. The differences between scores are preserved:

$$t_n(e_{ij}) = \frac{s_n(e_{ij}) - \mu_n}{\sigma_n} \tag{5.6}$$

where μ_n and σ_n are respectively the mean and standard deviation of the empirical distribution of the inferred scores $s_n(e_{ij})$ for network n. This normalisation does not assure a limited range of values.

ScaleL

The previous proposals normalise the network only taking into consideration the score values. *ScaleL* is an extension of the last normalising method. This method takes into account the local context of the scores of gene i and j for computing the normalised score of interaction $t_n(e_{ij})$. In the *ScaleL* method (L stands for *Local*), two local scaled values are initially computed (ζ_i and ζ_j):

$$t_n(e_{ij}) = \sqrt{\zeta_i^2 + \zeta_j^2}, \text{ with } \zeta_i = \frac{s_n(e_{ij}) - \mu_{s_i}}{\sigma_{s_i}}, \text{ and } \zeta_j = \frac{s_n(e_{ij}) - \mu_{s_j}}{\sigma_{s_j}} \tag{5.7}$$

where μ_{s_i} and σ_{s_i} denote the mean and standard deviation of the empirical distribution of the scores of all edges connected to gene i. They are defined as:

$$\mu_{s_i} = \frac{1}{G}\sum_{l=1}^{G} s_n(e_{il}), \tag{5.8}$$

$$\sigma_{s_i} = \sqrt{\frac{1}{G-1}\sum_{l=1}^{G}(s_n(e_{il}) - \mu_{s_i})^2} \tag{5.9}$$

Note that this rule is related to the CLR network inference method (Faith et al., 2007) which can be interpreted as a normalisation strategy as pointed out in (Bellot and Meyer, 2014). This normalisation step highlights a few links per node that stand out among all other scores of the gene. In this way, a "core" network with the most relevant (and presumably true) links is obtained.

5.5.2 Aggregation

Three different aggregation techniques will be studied. Assume $a(e_{ij})$ denotes the aggregated value of the normalised scores.

Sum

The *Sum* is a simple summation process that is equivalent to the average of the N values of

each link:

$$a(e_{ij}) = \sum_{n=1}^{N} t_n(e_{ij}). \tag{5.10}$$

Median

Finally, the *Median* method assigns the median of the N values:

$$a(e_{ij}) = \text{Median}\left\{t_1(e_{ij}), \ldots, t_N(e_{ij})\right\} \tag{5.11}$$

This method could be seen as a particularization of the (RankOrderFilter$_k$) with a fixed value of $k = {}^{N}/_{2}$.

Weighted Sum

It has been noticed that some algorithms perform better than others (see section 4.3). So, it may have sense to integrate only in the consensus the best-performing methods. The *WeightedSum* is based on this idea and implements it giving methods with a better performance a higher associated weight (w_n).

$$a(e_{ij}) = \sum_{n=1}^{N} w_n \cdot t_n(e_{ij}). \tag{5.12}$$

These weights can be learned in a supervised manner as in (Marbach et al., 2012b). The authors propose a logistic regression-based binary classifier, where the class label represents the presence or absence of an edge.

In the sequel, we propose to estimate these weights in an unsupervised manner since the networks are very different and the knowledge is very sparse. We will define a strategy where weights are proportional to the "topological quality" of the networks. The "topological quality" is measured by the relative frequency of some graphlets as compared to other graphlets. (Ruyssinck et al., 2016) proposes to use the frequency of graphlet G4 among the graphlets of four nodes (see Figure 4.11).

The details of how the weights of the Weighted Sum are estimated are presented in section 5.7.

5.5.3 Consensus network algorithms

The combination of four possible normalisation strategies with three possible aggregation

rules gives rise to 12 different consensus network algorithms. Regarding nomenclature, the algorithms will be referred to by the two names of the two steps, such that each word or abbreviation of the two steps begins with a capital letter.

Note that some of the combinations give a consensus method that has already been published. These methods are listed in Table 5.1. The other methods have not been reported in the literature and, to our knowledge, they are studied here for the first time.

Table 5.1 State-of-the-art consensus network algorithms.

Normalisation	Aggregation	Name	Reference
Identity	*Sum*	*IdSum*	(Marbach et al., 2012b)
Rank	*Sum*	*RankSum*	(Marbach et al., 2012a)
Rank	*Median*	*RankMed*	(Hase et al., 2013)

5.6 Data

In this section, we present the data that will be used to test the methods presented in the previous section. They belong to the two different scenarios. The *Homogeneous* situation reflects the case where the individual networks have been inferred with various algorithms but with the same type of data like gene expression as in (Marbach et al., 2012a). In order to get an estimation of the degree of homogeneity that might be expected in this kind of scenario, we have downloaded the networks from the supplemental information of (Marbach et al., 2012a). To measure the uniformity between networks, we have converted them to vectors and computed the correlation between the networks obtaining a mean correlation of 0.6.

On the other hand, the *Heterogeneous* scenario reflects the case where the individual networks have been inferred from very different kind of data as in (Marbach et al., 2012b). The N individual networks are very different, hardly having any edge in common. To get an estimation of the expected correlation in this scenario, we have downloaded the networks from supplemental material of (Marbach et al., 2012b) and converted them to vectors. As before we have computed the correlation between the networks obtaining a mean correlation of 0.06.

We will use data of three different origins, going from a synthetic scenario to a real one. In the first place, the networks will be generated in a synthetic manner to reproduce individual networks. In the second frame, we will rely on real GRNs inference algorithms

applied on a simulated data as individual networks. Moreover, finally, we will use network inference algorithms on real data.

5.6.1 Synthetic network generation

The different consensus methods are meant to be used for the integration of networks obtained through the use of real inference algorithms. However, first, we use synthetically generated networks. This allows providing a first approach and estimation of the performance of the different consensus network alternatives. Creating the individual networks in a synthetic manner allows us to control the degree of homogeneity between networks. Moreover, more important, we do not depend on any specific network inference algorithm. So, instead of relying on real network inference algorithms, we rely on a subsampling strategy applied on a real network (True_{Net}), coming from Table 4.2.

Homogeneous scenario

The N individual networks are very similar and have many edges in common. To create the dataset corresponding to this scenario, a unique network is first created by a subsampling of True_{Net}. Then, this network is altered N different times by introducing hard errors (false positives and negatives) and by adding a Gaussian noise to the scores associated to all edges. The alteration parameters are chosen so that the homogeneity of the resulting networks is similar to the one obtained when various inference algorithms are applied to the same data.

Heteregeneous scenario

In this case, the N individual networks are very different and hardly have any edge in common. To reflect this situation, N different networks are generated through various independent subsampling of True_{Net}. As previously, these networks are then altered N different times with the introduction of hard errors (false positives and false negatives), and then with soft errors through the addition of noise.

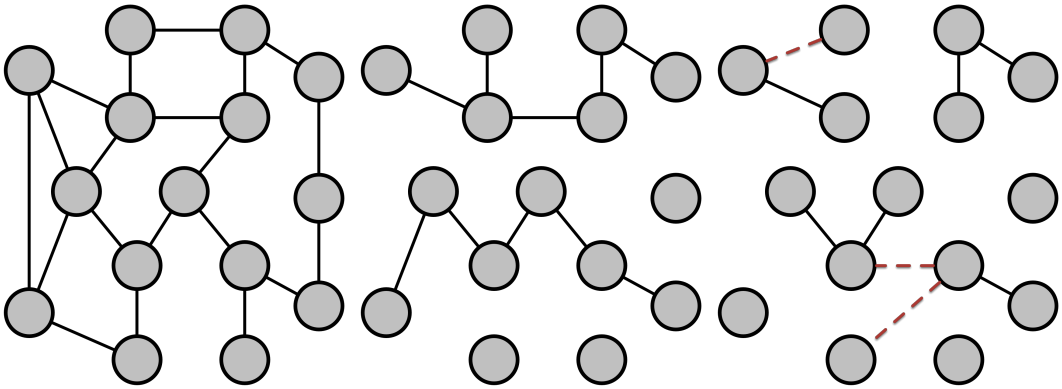
5.6.1.1 Subsampling strategy

The networks of the two scenarios are generated with the Algorithm 1, which is illustrated with a toy example in Figure 5.1. A toy True_{Net} is shown in Figure 5.1a. It has 15 genes (illustrated with circles) and 20 edges (illustrated with lines). The subsampling step of Algorithm 1 selects randomly $\tau\%$ edges of the True_{Net} to get ν_i. In Figure 5.1b a particular

case of ν_i is shown, in this case $\tau = 50$ so ν_i has 10 edges. Then, m% of errors (both false positives and false negatives) are introduced. Following the example and with $m = 30$, this will introduce 3 false negatives and 3 false positives to obtain η_i. The resulting network is presented in Figure 5.1c, where the dashed lines represent the false positives. The following steps of Algorithm 1 are meant to generate realistic networks. First a noise is added to the network, after this process a constant value is added to shift the scores values in order to ensure non-negative scores in the networks. This step introduces more false positives with a low confidence (if δ is small) and also introduces variability of the scores in the true "recovered" edges.

Input: True_{Net}, *Heterogeneous*, N
Output: N individual synthetic networks $\left(\{\eta_n\}_{n=1}^{N}\right)$
$n \leftarrow 1$;
while $n \leq N$ **do**
  if *Heterogeneous* **then**
    $\nu_n \leftarrow$ Subsample edges from the True_{Net};
  else
    if *n==1* **then**
      $\nu_1 \leftarrow$ Subsample edges from the True_{Net};
    else
      $\nu_n \leftarrow \nu_1$;
    end
  end
  $\eta_n \leftarrow$ Introduce errors in the network ν_n;
  Add noise to η_n;
  $n \leftarrow n + 1$
end

Algorithm 1: Generate N individual synthetic networks.



(a) True_{Net} (b) Subsampled network ν_n (c) ν_n with errors

Fig. 5.1 Toy example illustration of the first steps of Algorithm 1.

5.6.2 DREAM5 Data

In a second step, we will use the predictions of 35 real GRN inference algorithms on the first simulated dataset of DREAM5 (Marbach et al., 2012a). We use the inferred networks by the participants of the challenge. These networks are obtained from the same insilico data with different approaches. So we can consider it in the *Homogeneous* scenario. However, these networks present a particularity, the different algorithms presents a high variability in their performances (see Figure 5.6).

5.6.3 Real Data

Finally, the proposed consensus procedures are tested on public data sets of well-studied model organisms. With this study, we will improve our understanding of the algorithms seeing their performances when dealing with real data.

In this case, the metrics are computed according to a partial Gold standard (GS). This standard is generated by collecting all curated interactions for a particular organism. Therefore, this leads to a partial knowledge of the network.

All the collected interactions are treated as true positives, moreover, all predicted interactions between genes that are not documented in the curated database are treated as false positives. Such evaluation tends to overestimate the false positive rate (FPR), as most genes probably interact with much more TF's than currently documented ones. Moreover, predictions for transcription factors and genes that are not part of the GS, i.e., for which no experimentally supported interactions exist, are ignored in the evaluation. This evaluation

reward methods that tend to reproduce current knowledge and penalises those that could find new results (De Smet and Marchal, 2010).

In particular, we use two real data. The first one is chosen to represent the *Homogeneous* scenario and the second one the *Heteregeneous* scenario.

5.6.3.1 Escherichia coli

We have selected Escherichia coli (E. coli), a very well known bacteria, since predictions can be validated against the GS from RegulonDB database. The RegulonDB database (Gama-Castro et al., 2011; Salgado et al., 2012) contains the largest and best-known information on transcriptional regulation in E. coli. Thus, it has been used as a Gold standard (GS) to evaluate the accuracy of the variously constructed networks.

We use different E. coli datasets:

1. *Ecoli1*: Many Microbe Microarrays Database ("M3D") (Faith et al., 2008) which contains 907 microarrays measured under 466 experimental conditions using Affymetrix GeneChip E. coli Genome arrays.

2. *Ecoli2*: Expression data from laboratory evolution of Escherichia coli on lactate or glycerol (Fong et al., 2005) (GSE33147), which contains 96 microarrays of laboratory adaptive evolution experiments using Affymetrix E. coli Antisense Genome Arrays.

3. *Ecoli3*: Expression data from (Sangurdekar and Srienc, 2006; Xiao et al., 2011) which contains 217 arrays that measures the transcriptional response of E. coli to different perturbations and stresses, such as drug treatments, UV treatments and heat shock.

Note that this data will be used to infer the networks with the GRN methods that we have evaluated in chapters 3 and 4 (ARACNE, C3Net, CLR, Genie3, MRNET, Zscore, VIGR.PLS and VIGR.GBM). Then, we use the different consensus strategies to integrate them.

5.6.3.2 Drosophila melanogaster

The fruit fly Drosophila melanogaster provides an ideal model organism for the inference and study of functional regulatory networks in multicellular organisms. There exists a rich literature about regulatory relationships, which have resulted in small, but high-quality networks of known regulatory interactions such as REDfly (Halfon et al., 2008).

We have selected the data used in (Marbach et al., 2012b), since it provides a Heterogeneous scenario with networks of the same organism that comes from different kind of

data. There is a total of six networks that comes from both functional and physical regulatory interactions.

5.7 Weighted Sum

In this section we detail the strategy to estimate the weights of the weighted sum, formulated as following:

$$a(e_{ij}) = \sum_{n=1}^{N} w_n \cdot t_n(e_{ij}), \tag{5.13}$$

where the weights w_n should ponderate the N individual networks, according to a "topological quality" criterion.

In the previous chapter, we have introduced the graphlets as a local topology's descriptor, see section 4.4. So, we will base our "topological quality" measurement onto those descriptors. To derive it into the next subsection, we study the relative frequency of any graphlet compared to all other graphlets.

5.7.1 Graphlet histogram study

Here, we study the distribution of different graphlets and see if any trend arises in the network topologies. Figure 5.2 shows the probabilites of the different graphlets up to 5 nodes for the synthetic networks of Table 4.2 discussed in subsections 5.6.1 and 5.6.2. While Figure 5.3 shows the graphlet probability of the real networks presented at section 5.6.3. Observing Figure 5.2, we see that G4 node seems a little more frequent compared to all other 4-node graphlets as (Ruyssinck et al., 2016) suggests. However, if we observe Figure 5.3, G3 and G4 have similar proportions. However, it seems that G10 and G11 presents a high relative frequency for both figures (all the different graphlets can be found in Figure 4.11). Therefore, we propose that the topological criterion should reward a higher frequency of these specific graphlets.

We denote $|G_i(A)|$ as the number of graphlet-i that are present in network A, then the "topological criterion" $(\vartheta(A))$ may be defined as:

$$\vartheta(A) = \frac{|G_{10}(A)| + |G_{11}(A)|}{\sum_{i=0}^{29} |G_i(A)|} \tag{5.14}$$

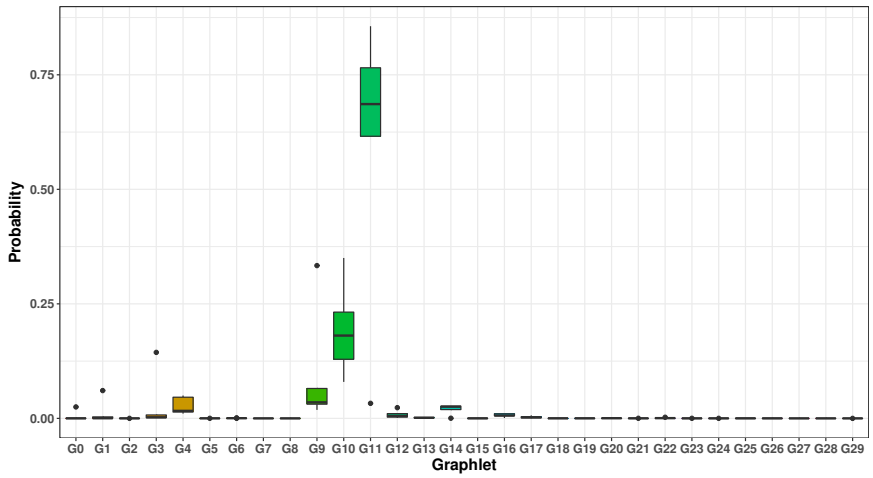


Fig. 5.2 Boxplots with frequency of different graphlets in synthetic networks.

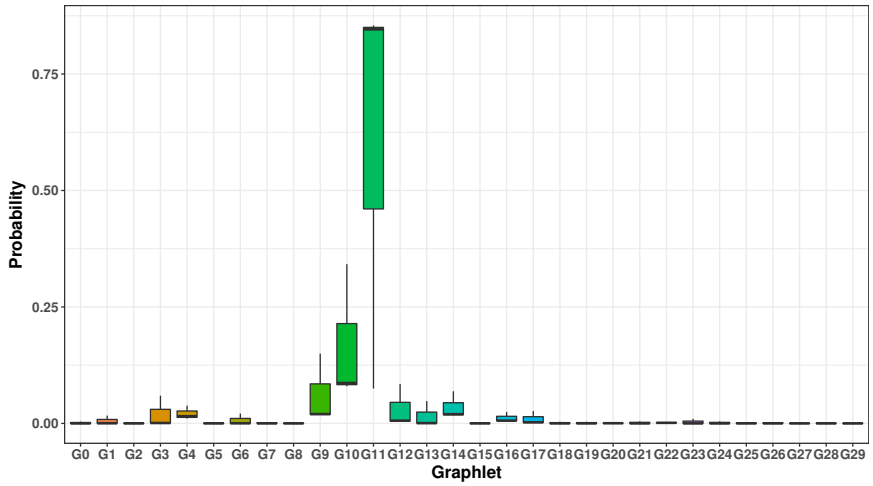


Fig. 5.3 Boxplots with frequency of different graphlets in real networks.

5.7.2 Formulation as an optimization problem

The frist idea to handle the topological criterion is to state the problem as an optimisation. The optimization procedure will find the weights that maximise the "topological quality" of the consensus network $\vartheta(A)$:

$$\begin{aligned} \underset{w_n}{\text{maximise}} \quad & \vartheta(A) \\ \text{subject to} \quad & a(e_{ij}) = \sum_{n=1}^{N} w_n \cdot t_n(e_{ij}), \\ & 0 \leq w_n \leq 1. \end{aligned} \tag{5.15}$$

This cost function to be optimised is non-convex and presents many local maxima which does not allow to use a convex optimization solver.

So, in order to find the weights in such formulation, one may use a metaheuristic to explore the solution space. We opted for genetic algorithm (GA) (Mitchell, 1998), which is inspired by the process of natural selection that belongs to the larger class of evolutionary algorithms (EA). Genetic algorithms are commonly used to find a solution to optimisation and search problems using bio-inspired operations such as mutation, crossover and selection in order to find the best fitting on the search space.

5.7.3 Approximation of the optimisation problem

The previous formulation and solving of the problem with GA have a high computational load and does not scale well. If the number of individual networks to integrate or if the size of the network increases the optimisation problem becomes more and more difficult to solve.

So, we propose a sub-obtimal estimation of the weights. Instead of estimating the weights that gives a better $\vartheta(A)$, a second strategy is to calculate this "topological criterion" $\vartheta(\cdot)$ for each of the individual networks to integrate (T_n). Finally the weights could be obtained rescaling the $\vartheta(\cdot)$ measures:

$$w_n = \frac{\vartheta(T_n)}{\sum_{k=1}^{N} \vartheta(T_k)} \tag{5.16}$$

The intuition behind this formulation is to give higher weights to networks that have better topological characteristics.

5.7.4 Comparison of both options

In this subsection, we test if the previously described strategies To test them, we have done a preliminary study using the synthetically generated networks described in subsection 5.6.1. As in chapter 4, we use AUPR_5 of the consensus network. Figure 5.4 shows the AUPR_5 obtained by the different options to estimate the weights of equation 5.13 after the scaling normalisation.

We can observethat with the weights obtained with the optimisation formulation in equation 5.15, the consensus network consistently obtains better AUPR_5. However, the optimisation alternative is not garanteed to converge in a certain number of iterations. On the other hand, the estimated weights with the approximation scheme are very fast to obtain and it seems that the loss on performance is acceptable.

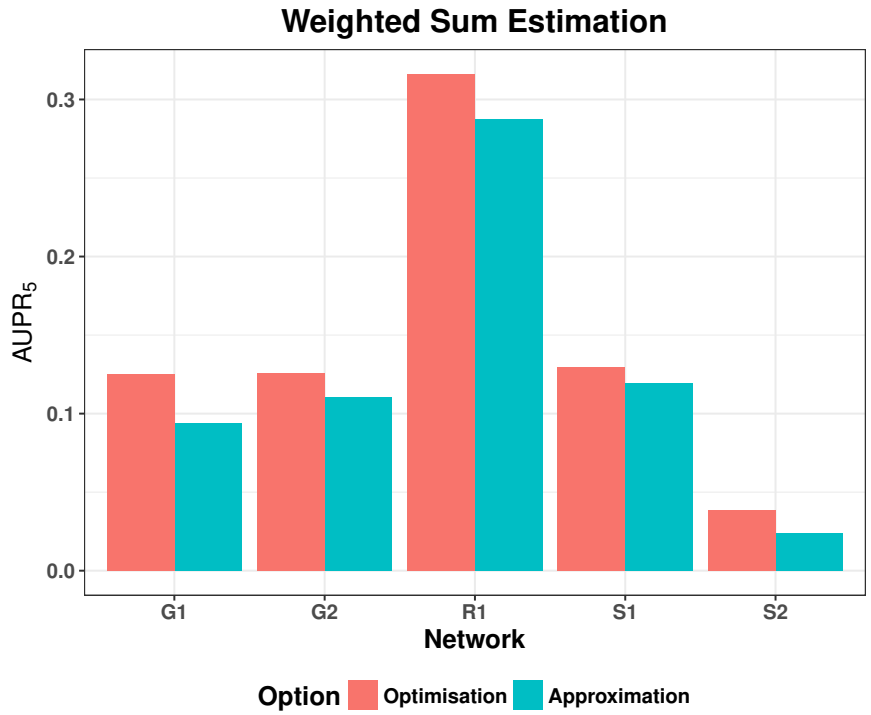


Fig. 5.4 $AUPR_5$ obtained with different options of *ScaleLSum* on the synthetically generated networks.

5.8 Results

As mentioned before, the performances of the various algorithms are benchmarked with both real and synthetic networks. As previously, we use $AUPR_5$ of the consensus network. But as we can observe in Figure 5.4, we are comparing very different data and situations, presenting different sizes and topological properties. Moreover, using $AUPR_5$ it is not possible to know if the consensus method is improving or not from the individual networks. Therefore, we propose to normalise the $AUPR_5$ with respect to the mean $AUPR_5$ of the individual networks ($\mu_{AUPR_{5;ind}}$):

$$AUPR_{5;norm} = \frac{AUPR_5}{\mu_{AUPR_{5;ind}}} \tag{5.17}$$

With this approach, we can compare the different networks. And now, it is possible to know if the consensus method is improving on the average network (if $AUPR_{5;norm}$ is greater than 1).

5.8.1 Results on Synthetic networks

In the case of synthetic networks, the individual networks are generated with Algorithm 1 with the parameters specified in Table 5.2. Using these values, the mean correlation between the networks in the *Homogeneous* case is 0.66, and the average correlation of the *Heterogeneous* case is 0.003. On this experimental setup, we generate the individual networks for each one of the networks of Table 4.2, and this procedure is repeated ten times in order to have different runs of Algorithm 1 and therefore different pools of individual networks.

Table 5.2 Algorithm 1 parameters to generate the experimental setup for synthetic networks.

Parameter	Value
Number of individual networks (N)	10
Subsampling (t) %	15
Introduced errors (m) %	20

The results reported in Figure 5.5 present the boxplots of $\text{AUPR}_{5;\text{norm}}$ of different consensus algorithms across all networks. Each box represents the statistics of a method, at the *Homogeneous* scenario and *Heterogeneous* scenario.

In the *Heterogeneous* case, we observe bigger differences between different consensus proposals, where some of them reach worst results compared to the average individual network. Observing the figure we can confirm that the *IdSum* method that was used in (Marbach et al., 2012b) in a *Heterogeneous* scenario is a good choice for this case. However, using the *Rank* as normalisation step and *Sum* as aggregation step provides even better results.

On the other hand, the *homogeneous* scenario, it can be concluded that consensus network algorithms allow improving the inference compared to the average individual networks as the $\text{AUPR}_{5;norm}$ is around 4 for all consensus methods. This conclusion is in line with previous publications such as (Hase et al., 2013; Marbach et al., 2012a). However, there are few differences between various algorithms. This case almost shows no significant differences between methods. Therefore, we think that the best option is to use the *RankSum* (Marbach et al., 2012a) or *IdSum* (Marbach et al., 2012b), as they are already part of the state-of-the-art and have a simpler normalisation and aggregation methods.

This study shows how in both scenarios combining the results of multiple inference methods is a good strategy for improving the individual results. However, in the *Heterogeneous* case, it seems that there is more potential for improvement. In average, the results improve by 8.5 outperforming the *Homogeneous*' results, which has an average $\text{AUPR}_{5;\text{norm}}$ of 3.7.

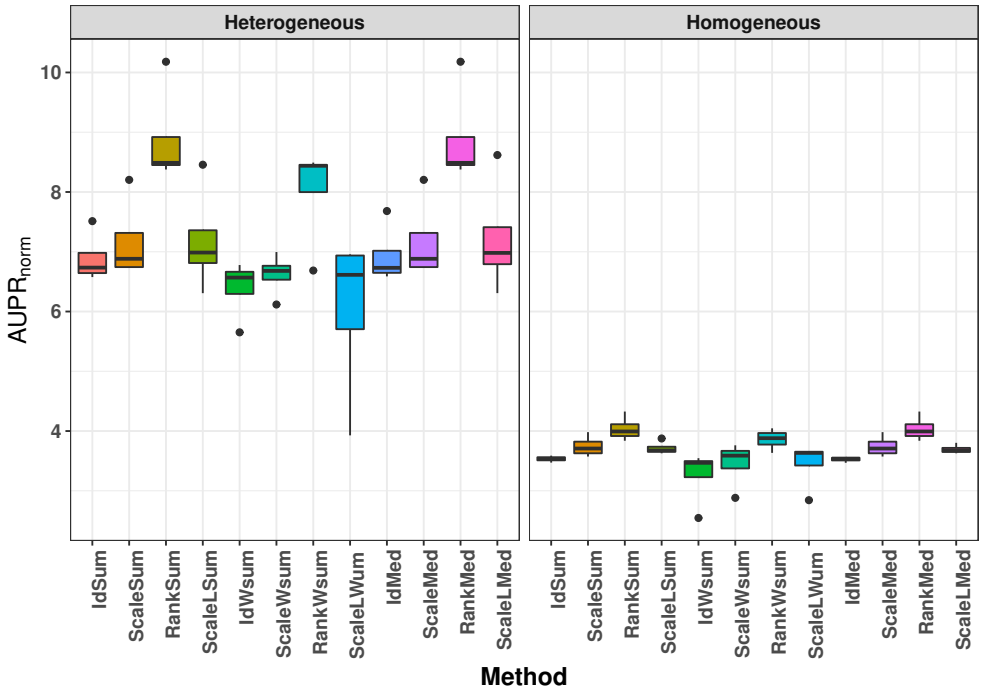


Fig. 5.5 Boxplots of $AUPR_{5;norm}$ performance of consensus methods on synthetic generated networks.

Since the individual results are synthetically generated, the obtained results should be interpreted as an estimation of the potential of consensus methods in the two different scenarios. Therefore, in the following subsections, we will study the different consensus methods on more real datasets.

5.8.2 Results on DREAM5

In this subsection, we have integrated the predictions of all 35 GRN predictions on DREAM5 to construct community networks with the different approaches.

The different consensus networks obtain in average better performances than the 35 applied inference methods, which shows that the community network is consistently as good or better than the top individual methods (Figure 5.6). Some of the top-performing methods are competitive with some of the consensus methods. However, as we have seen in the previous chapter the performance of individual methods does not generalise across different networks. Moreover, it is difficult to know in a real situation which one of the algorithms has the best performance. Furthermore, in Figure 5.6, we can see how the mean value of $AUPR_5$ of individual networks is 0.17, while of the consensus network is 0.31. Obtaining

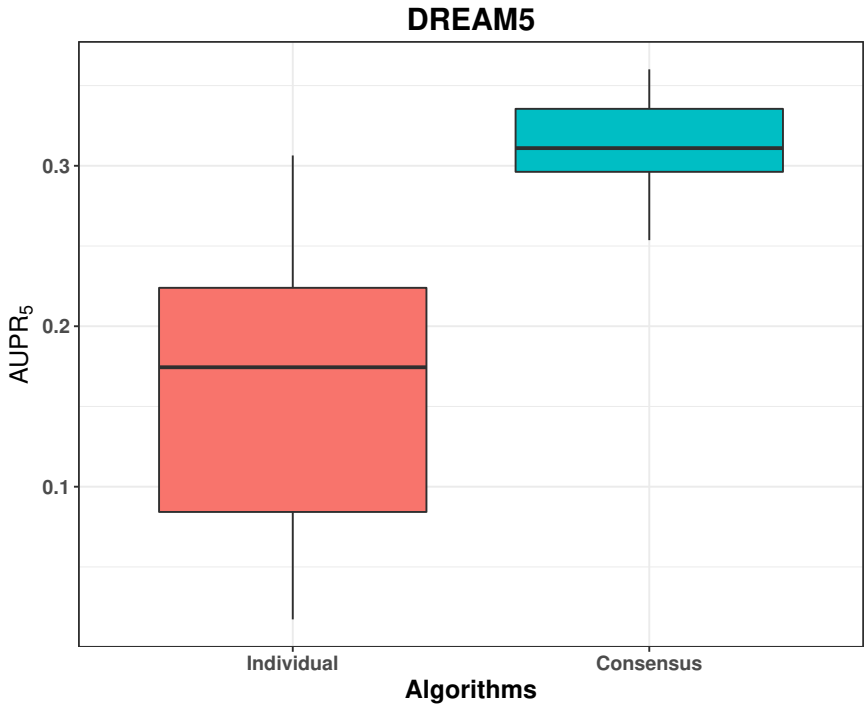


Fig. 5.6 Boxplots of $AUPR_5$ performance of individual networks and consensus methods on DREAM5.

a $AUPR_{5;norm} = 1.78$, which is a value more modest than the ones obtained on Synthetic generated networks.

Finally, the values of $AUPR_{5;norm}$ for each individual consensus method are shown in Figure 5.7. In this case, our proposal *ScaleLWsum* achieves the best consensus with a value of $AUPR_{5;norm} = 2.24$. We think that a big part of this good integration is obtained thanks to the normalisation step since *ScaleLSum* also almost obtain the same performance. However, comparing methods that use *Sum* with those that uses *WSum* we see that the weighted approach is systematically above. So both aspects seem important.

It is worth noting that the estimated weights are the same ones that are applied in the weighted sum after different normalisation methods. Despite that, we can observe how *ScaleLWsum* and *RankWsum* obtain very different performances. This fact points that the normalisation step has a huge impact in this scenario. To corroborate this, Figure 5.8 shows the statistics of the distribution of the original scores from 35 individual networks on DREAM5. It can be observed that there exist large differences in the scores distributions of different GRN methods.

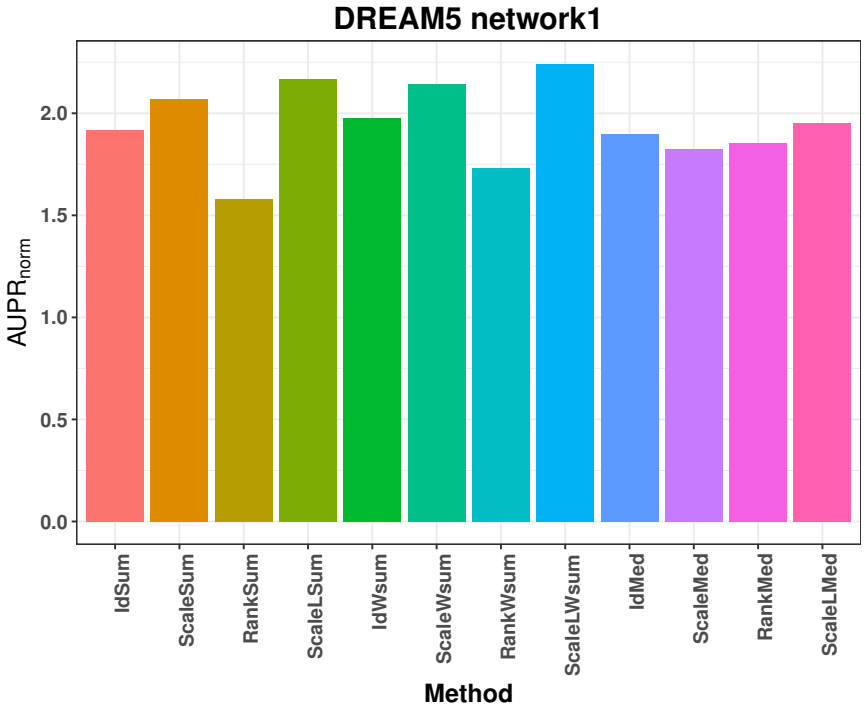


Fig. 5.7 Boxplots of $AUPR_5$ performance of individual networks and consensus methods on DREAM5.

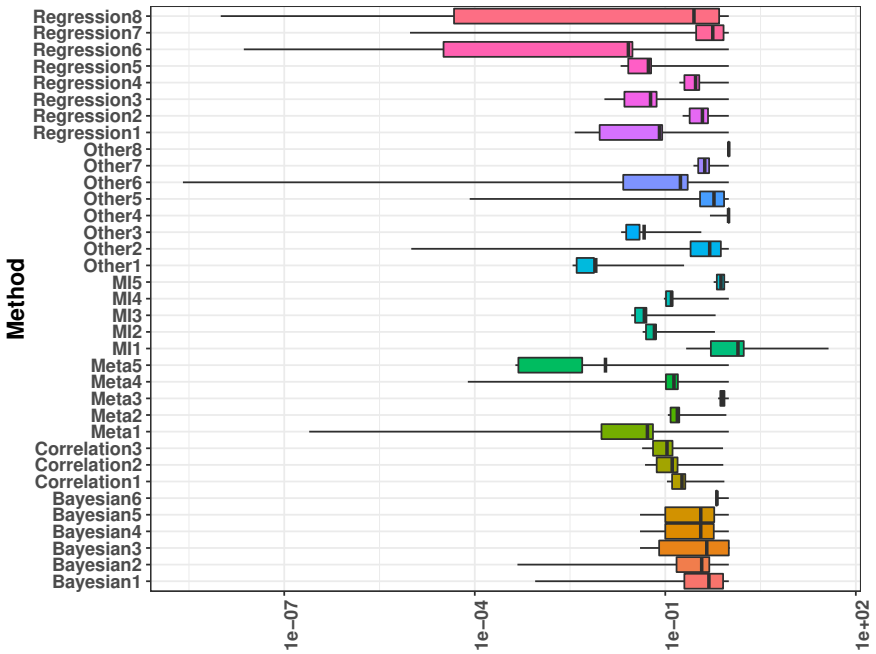


Fig. 5.8 Boxplots of score values of 35 individual networks on DREAM5.

5.8.3 Results on Real Data

To finish the analysis of consensus methods and their limits, we will evaluate them on the real data described in subsection 5.6.3.

With the three different E.coli microarray data, we have inferred GRN with the different methods described in chapter 3. Afterwards, we have integrated these networks with the different consensus proposals and evaluated them. Figure 5.9 shows the $AUPR_{5;norm}$ obtained by the different consensus proposals for each one of the datasets. The mean $AUPR_{5;norm}$ of the three values is marked with a black diamond.

Figure 5.9 confirms the conclusions reached in the previous subsection. *ScaleLWsum* is the best consesus method while *RankWsum* even obtains a worst result than the average individual method, $AUPR_{5;norm} < 1$, in Ecoli3 dataset.

The previous results reflect that in a *Homogeneous* scenario *ScaleLWsum* is the best alternative to the studied consensus methods. In order to have a larger picture, we evaluate the performance of consensus methods in a real *Heteregeneous* scenario with FlyNet data shown in Figure 5.10. Observing the figure we can confirm that the *IdSum* method that was used in the original work (Marbach et al., 2012b) is the best choice for this case. In this case, using the *ScaleL* as normalisation step and *Wsum* as aggregation step does not provide the best results, but it does not reach worst results compared to the average individual network.

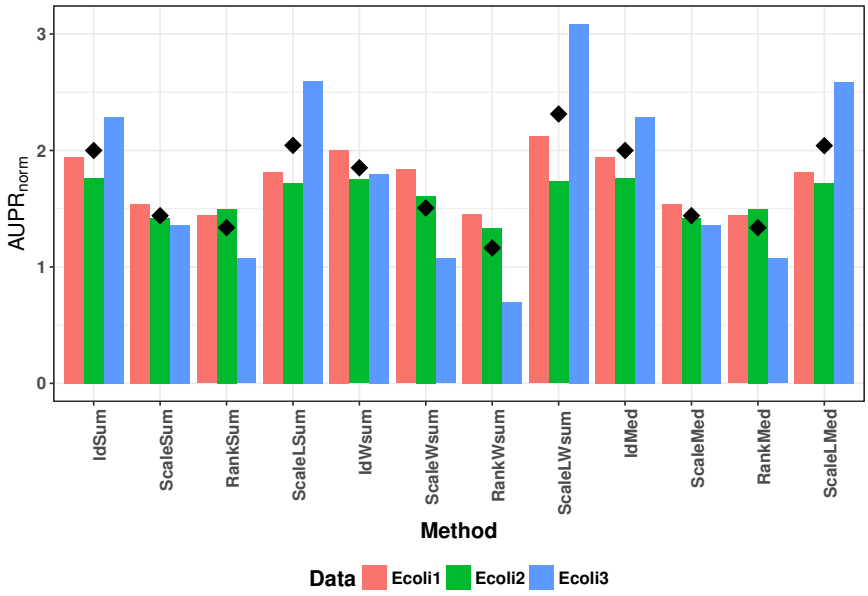


Fig. 5.9 $AUPR_{5;norm}$ performance of consensus methods on E.coli datasets.

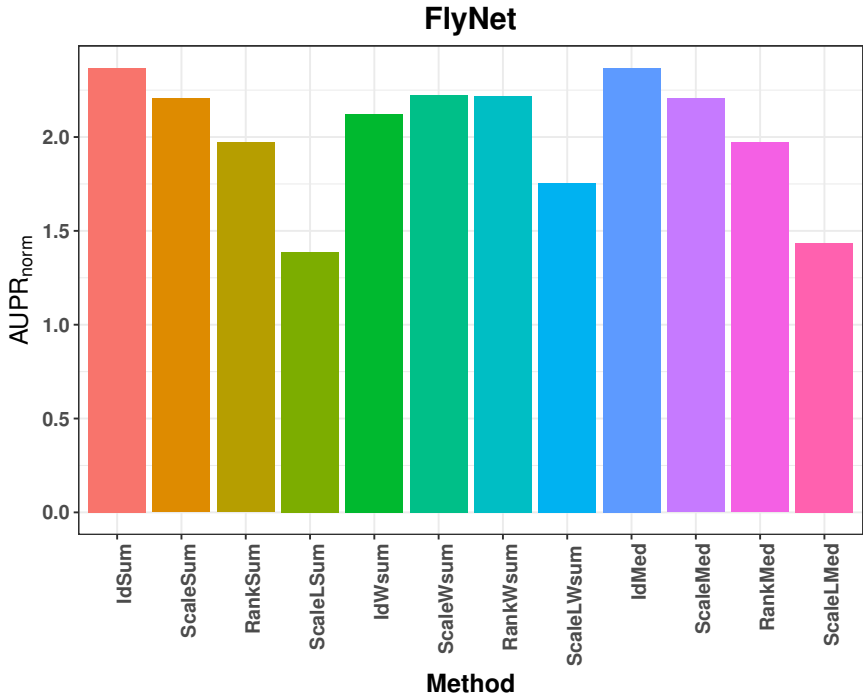


Fig. 5.10 $AUPR_{5;norm}$ performance of consensus methods on FlyNet.

5.9 Conclusions

In this chapter, we have reviewed different alternatives for post-processing GRN predictions and the means to combine them. In particular, we have proposed a framework for combining and integrating different inferred networks. It has been defined as a two-step process, consisting in a normalisation strategy followed by an aggregation technique. We studied two different scenarios of practical interest: *Homogeneous* (a situation where various network inference algorithms are used on the same data) and *Heterogeneous* (a situation where various sources of data are used to generate the individual networks), with a controlled synthetic experimental setup. The results show how in a *Homogeneous* scenario combining individual networks generally outperforms the mean individual network, and that the different analysed algorithms do not present significant differences. However, in a *Heterogeneous* scenario the differences are very significant, and the potential win is much bigger. The choice of the proper normalisation and aggregation steps allows very large improvements to be obtained.

Finally, we have studied how those results compare in different kinds of data when dealing with *Homogeneous* networks with very different performances and moreover in a real *Homogeneous* and *Heteregeneous* scenario. We have concluded that in this case, the improvement over the average individual network is smaller than the synthetic study. However, the increase in performance seems to be still attractive. We have concluded that *ScaleLWsum* is interesting in some cases but maybe not all them.

Chapter 6

Conclusions and Future Work

This thesis has been devoted to the analysis of network inference methods from gene expression datasets. These datasets have thousands of variables and only tens of noisy samples. Moreover, there exist non-linear multivariate dependencies between variables and little a priori knowledge and labelled data.

The motivation is to be able to model a cell allowing us to design new therapies, even to create or change organisms for new purposes. For example, we could manipulate microbes or algae to generate biofuel or produce other useful goods. From a biomedical perspective, it opens the understanding of a disease's causes and not only its symptoms. It enables to design a suitable drug targeting a particular disorder and avoiding side effects of nowadays's drugs.

6.1 Accomplished work

6.1.1 Network Inference

We have presented a methodology to infer a network by selecting the important variables for each variable of the dataset. Gene expression datasets present a large number of variables and few experiments or samples, and avoiding the selection of redundant variables is a challenging task. In this thesis, we have introduced VIGR. This method relies on a variable importance estimation that is finally used with two classical methods: Partial Least Squares (PLS) Regression (Geladi and Kowalski, 1986) and Gradient Boosting Machine (GBM) (Friedman, 2002). Using DREAM4 data we have concluded that our proposed approach is competitive compared with the other methods.

6.1.2 Performance assessment of inference methods

In the process of designing VIGR, we realised that many methods used their own data to evaluate themselves and the only public data that was used as a standard reference was the DREAM challenges. This led to a possibility to overtrain methods to this synthetic data. This motivates the development of a new benchmark for inference methods. This thesis introduces a framework for critical performance assessment of such methods, which consists of:

- the inclusion of several datasources generated with different simulators.
- tools for generating realistic datasets from these datasources.
- tools for evaluating the performance of inference methods.
- the possibility to test the robustness of the algorithms to noise and to number of experiments.

This benchmark of realistic in silico datasets is provided in a Bioconductor package. This package allows the reproducibility of the performance assessment in an easy way. The NetBenchmark package is gaining attention as a benchmark tool in the field, as it has already been downloaded over 3600 times in these past two years.

6.1.3 Consensus of inference methods

Using this tool we have seen that inferring gene regulatory networks from expression data is a tough problem. The different algorithms have some particular biases and strengths, and none of them is the best across all types of data and datasets.

We have introduced the idea of aggregating various network inferences through a consensus mechanism. In particular, we have presented a framework to standardise already proposed consensus methods, and based on this framework different proposals are introduced and analysed in different scenarios. We have concluded that by performing a consensus we can obtain an improvement over the average individual network. Also, we have found that our proposal *ScaleLWsum* seems to be a valuable choice and the best in some of the analysed real data.

6.2 Future Direction

This section aims to sketch some future directions to improve this work. We mention four main research axis:

1. Algorithmic improvements: We have seen that VIGR approach is competitive with state-of-the-art approaches. However, its main drawback is the computational load. Some improvements can be made in order to make its use feasible for datasets with thousands of genes. The problem is easily parallelisable, so a first step should explore this direction. If this is not sufficient, a closer look at the parameters of PLS and GBM may be considered.

2. Biological validation: our work has mainly used synthetic data to validate the proposed methods experimentally. However, applications of these techniques to biological data should be performed to discover new gene interactions or to identify potential drug targets to demonstrate its practical purpose.

3. Extension of techniques:

   (a) In order to compute the consensus network we used as a normalisation step among other the scale. The scale computes the standard score also called z-values. They are most frequently used to compare an observation to a standard normal deviate, though they can be defined without assumptions of normality. A generalisation of such score for the real distribution of raw scores could result in an improvement of consensus methods based on either *Scale* or *ScaleL*.

   (b) The weights used in the weighted sum are estimated to the topological quality of the individual networks, so they are the same regardless of the normalisation that is used. We have formulated an optimisation problem to find the weights that improves the topological properties of the normalised networks' consensus. However, this formulation does not scale well. Therefore, another formulation of this optimisation problem might be investigated so that an efficient solution could be obtained.

4. Comparing meta-networks: in order to extend this work, a full study of comparison between pairwise ensemble methods, data-merging based methods and post-processing alternatives should be performed onto several scenarios and different kind of data. The aim of such study, should reveal the strengths and biases of these methods.

References

Adler, P., Kolde, R., and Kull, M. (2009). Mining for coexpression across hundreds of datasets using novel rank aggregation and visualization methods. *Genome*.

Albert, R., Jeong, H., and Barabási, A.-L. (2000). Error and attack tolerance of complex networks. *nature*, 406:378–382.

Altay, G. and Emmert-Streib, F. (2010a). Inferring the conservative causal core of gene regulatory networks. *BMC Syst Biol*, 4.

Altay, G. and Emmert-Streib, F. (2010b). Revealing differences in gene network inference algorithms on the network-level by ensemble methods. *Bioinformatics (Oxford, England)*, 26(14):1738–1744.

Amaral, L. A. N., Scala, A., Barthelemy, M., and Stanley, H. E. (2000). Classes of small-world networks. *Proceedings of the National Academy of Sciences of the United States of America*, 97(21):11149–52.

Balaji, S., Babu, M., Iyer, L., and Luscombe, N. (2006). Comprehensive analysis of combinatorial regulation using the transcriptional regulatory network of yeast. *Journal of molecular biology*, 360:213 – 227.

Bansal, M., Belcastro, V., Ambesi-Impiombato, A., and di Bernardo, D. (2007). How to infer gene networks from expression profiles. *Mol Syst Biol*, 3.

Barabási, A.-L. and Albert, R. (1999). Emergence of scaling in random networks. *Science*, 286(October):509–512.

Barabási, A.-L. and Bonabeau, E. (2003). Scale-free networks. *Scientific American*, 3(1):50–59.

Barabási, A.-L. and Oltvai, Z. N. (2004). Network biology: understanding the cell's functional organization. *Nature reviews genetics*.

Barzel, B. and Barabási, A.-L. (2013). Network link prediction by global silencing of indirect correlations. *Nature biotechnology*, 31(8):720–5.

Barzel, B. and Biham, O. (2009). Quantifying the connectivity of a network: The network correlation function method. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 80(4).

Bastiaens, P., Birtwistle, M., and Blüthgen, N. (2015). Silence on the relevant literature and errors in implementation. *Nature*.

Belcastro, V., Siciliano, V., and Gregoretti, F. (2011). Transcriptional gene network inference from a massive dataset elucidates transcriptome organization and gene function. *Nucleic acids*.

Bellot, P. and Meyer, P. E. (2014). Efficient combination of pairwise feature networks. *JMLR: Workshop and Conference Proceedings,*.

Bellot, P., Olsen, C., Salembier, P., Oliveras-Vergés, A., and Meyer, P. E. (2015a). NetBenchmark: a bioconductor package for reproducible benchmarks of gene regulatory network inference. *BMC bioinformatics*, 16(1):312.

Bellot, P., Salembier, P., Oliveras-Vergés, A., and Meyer, P. E. (2015b). Study of Normalization and Aggregation Approaches for Consensus Network Estimation. In *2015 IEEE Symposium Series on Computational Intelligence*, number 1, pages 1081–1086.

Bosio, M., Bellot, P., Salembier, P., and Oliveras-Vergés, A. (2011). Feature set enhancement via hierarchical clustering for microarray classification. In *Genomic Signal Processing and Statistics (GENSIPS)*.

Bosio, M., Bellot, P., Salembier, P., and Oliveras-Vergés, A. (2012). Gene expression data classification combining hierarchical representation and efficient feature selection. *Journal of Biological Systems*, 20(04):349–375.

Breiman, L. (1996). Bagging Predictors. *Machine Learning*, 24(421):123–140.

Breiman, L. (2001). Random forests. *Mach Learn*, 45(1):5–32.

Breiman, L., Friedman, J. H., Stone, C. J., and Olshen, R. A. (1984). *Classification and Regression Trees*, volume 19. CRC press.

Bunescu, R., Ge, R., Kate, R. J., Marcotte, E. M., Mooney, R. J., Ramani, A. K., and Wong, Y. W. (2005). Comparative experiments on learning information extractors for proteins and their interactions. *Artificial Intelligence in Medicine*, 33(2):139–155.

Butte, A. J. and Kohane, I. S. (2000). Mutual information relevance networks: functional genomic clustering using pairwise entropy measurements. *Pac Symp Biocomput*, 5:418–429.

Capaldi, A. P., Kaplan, T., Liu, Y., Habib, N., Regev, A., Friedman, N., and O'Shea, E. K. (2008). Structure and function of a transcriptional network activated by the MAPK Hog1. *Nature genetics*, 40(11):1300–6.

Chambers, J. M. (1983). *Graphical Methods for Data Analysis*. Wadsworth International Group, California, USA.

Conover, W. J. (1971). Practical Nonparametric Statistics.

Cover, T. M. and Thomas, J. A. (2005). *Elements of Information Theory*. John Wiley & Sons.

Crick, F. (1970). Central dogma of molecular biology. *Nature*, 227(5258):561–563.

Davis, J. and Goadrich, M. (2006). The Relationship Between Precision-Recall and ROC Curves. *Proceedings of the 23rd International Conference on Machine learning – ICML'06*, pages 233–240.

De Smet, R. and Marchal, K. (2010). Advantages and limitations of current network inference methods. *Nat Rev Microbiol*, 8(10):717–29.

Ding, C. and Peng, H. (2005). Minimum redundancy feature selection from microarray gene expression data. *Journal of bioinformatics and computational biology*, 3(2):185–205.

Efron, B. and Tibshirani, R. J. (1994). *An introduction to the bootstrap*. CRC press.

Elith, J., Leathwick, J. R., and Hastie, T. (2008). A working guide to boosted regression trees. *Journal of Animal Ecology*.

Entringer, R., Jackson, D., and Snyder, D. (1976). Distance in graphs. *Czechoslovak Mathematical Journal*.

Erdös, P. and Rényi, A. (1960). On the evolution of random graphs. *Publ. Math. Inst. Hungar. Acad. Sci.*

Faith, J., Driscoll, M., and Fusaro, V. (2008). Many Microbe Microarrays Database: uniformly normalized Affymetrix compendia with structured experimental metadata. *Nucleic acids*.

Faith, J. J., Hayete, B., Thaden, J. T., Mogno, I., Wierzbowski, J., and Cottarel, G. (2007). Large-scale mapping and validation of escherichia coli transcriptional regulation from a compendium of expression profiles. *PLoS biol*, 5.

Feizi, S., Marbach, D., Médard, M., and Kellis, M. (2013). Network deconvolution as a general method to distinguish direct dependencies in networks. *Nature biotechnology*, 31(8):726–33.

Fong, S., Joyce, A., and Palsson, B. (2005). Parallel adaptive evolution cultures of Escherichia coli lead to convergent growth phenotypes with different gene expression states. *Genome research*.

Freund, Y. and Schapire, R. (1995). A desicion-theoretic generalization of on-line learning and an application to boosting. *European conference on computational learning*.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232.

Friedman, J. H. (2002). Stochastic gradient boosting. *Computational Statistics and Data Analysis*, 38(4):367–378.

Gama-Castro, S., Jiménez-Jacinto, V., Peralta-Gil, M., Santos-Zavaleta, A., Peñaloza-Spinola, M. I., Contreras-Moreira, B., Segura-Salazar, J., Muñiz-Rascado, L., Martínez-Flores, I., Salgado, H., Bonavides-Martínez, C., Abreu-Goodger, C., Rodríguez-Penagos, C., Miranda-Ríos, J., Morett, E., Merino, E., Huerta, A. M., Treviño-Quintanilla, L., and Collado-Vides, J. (2008). RegulonDB (version 6.0): Gene regulation model of Escherichia coli K-12 beyond transcription, active (experimental) annotated promoters and Textpresso navigation. *Nucleic Acids Research*, 36(SUPPL. 1).

Gama-Castro, S., Salgado, H., and Peralta-Gil, M. (2011). RegulonDB version 7.0: transcriptional regulation of Escherichia coli K-12 integrated within genetic sensory response units (Gensor Units). *Nucleic acids*.

Gardner, T. S., di Bernardo, D., Lorenz, D., and Collins, J. J. (2003). Inferring genetic networks and identifying compound mode of action via expression profiling. *Science (New York, N.Y.)*, 301(5629):102–5.

Geladi, P. and Kowalski, B. R. (1986). Partial least-squares regression: a tutorial. *Analytica Chimica Acta*, 185(C):1–17.

Gentleman, R. C., Carey, V. J., Bates, D. M., Bolstad, B., Dettling, M., and Dudoit, S. (2004). Bioconductor: open software development for computational biology and bioinformatics. *Genome Biol*, 5.

Goadrich, M., Oliphant, L., and Shavlik, J. (2004). Learning Ensembles of First-Order Clauses for Recall-Precision Curves: A Case Study. *Biomedical Information Extraction (Proceedings of the Fourteenth International Conference on Inductive Logic Programming*, pages 98–115.

Halfon, M., Gallo, S., and Bergman, C. (2008). REDfly 2.0: an integrated database of cis-regulatory modules and transcription factor binding sites in Drosophila. *Nucleic acids research*.

Hase, T., Ghosh, S., Yamanaka, R., and Kitano, H. (2013). Harnessing Diversity towards the Reconstructing of Large Scale Gene Regulatory Networks. *PLoS Computational Biology*, 9(11).

Hastie, T., Tibshirani, R. J., and Friedman, J. H. (2009). The Elements of Statistical Learning. *Elements*, 1:337–387.

Haury, A.-C., Gestraud, P., and Vert, J. P. (2011). The influence of feature selection methods on accuracy, stability and interpretability of molecular signatures. *PLoS ONE*, 6(12).

Haury, A.-C., Mordelet, F., Vera-Licona, P., and Vert, J.-P. (2012). TIGRESS: Trustful Inference of Gene REgulation using Stability Selection. *BMC Systems Biology*, 6(1):145.

Hayes, W., Sun, K., and Pržulj, N. (2013). Graphlet-based measures are suitable for biological network comparison. *Bioinformatics*, 29(4):483–491.

Huber, W., Heydebreck, A. V., and Sültmann, H. (2002). Variance stabilization applied to microarray data calibration and to the quantification of differential expression. *Bioinformatics*, 18:S96–S104.

Huynh-Thu, V. A., Irrthum, A., Wehenkel, L., and Geurts, P. (2010). Inferring regulatory networks from expression data using tree-based methods. *PloS ONE*, 5(9).

Hwang, C.-R. (1988). Simulated annealing: theory and applications. *Acta Applicandae Mathematicae*, 12:108–111.

Jeong, H., Tombor, B., Albert, R., Oltvai, Z. N., and Barabási, A.-L. (2000). The large-scale organization of metabolic networks. *Nature*, 407(6804):651–654.

Johnson, W. E., Li, C., and Rabinovic, A. (2007). Adjusting batch effects in microarray expression data using empirical Bayes methods. *Biostatistics (Oxford, England)*, 8(1):118–27.

Jolliffe, I. (2002). *Principal component analysis*. Wiley Online Library.

Krishnan, A., Giuliani, A., and Tomita, M. (2007). Indeterminacy of reverse engineering of gene regulatory networks: The curse of gene elasticity. *PLoS ONE*, 2(6).

Langfelder, P. and Horvath, S. (2008). WGCNA: an R package for weighted correlation network analysis. *BMC bioinformatics*, 9:559.

Louppe, G., Wehenkel, L., Sutera, A., and Geurts, P. (2013). Understanding variable importances in forests of randomized trees. *Advances in neural information processing systems*, pages 431–439.

Madhamshettiwar, P. B., Maetschke, S. R., Davis, M. J., Reverter, A., and Ragan, M. A. (2012). Gene regulatory network inference: evaluation and application to ovarian cancer allows the prioritization of drug targets. *Genome Med*, 4(5):41.

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

Marbach, D. (2009). *Evolutionary reverse engineering of gene networks*. PhD thesis, EPFL.

Marbach, D., Costello, J. C., Küffner, R., Vega, N. M., Prill, R. J., Camacho, D. M., Allison, K. R., Kellis, M., Collins, J. J., and Stolovitzky, G. (2012a). Wisdom of crowds for robust gene network inference. *Nat Methods*, 9(8):796–804.

Marbach, D., Prill, R. J., Schaffter, T., Mattiussi, C., Floreano, D., and Stolovitzky, G. (2010). Revealing strengths and weaknesses of methods for gene network inference. *Proc Natl Acad Sci*, 107(14):6286–91.

Marbach, D., Roy, S., Ay, F., Meyer, P. E., Candeias, R., and Kahveci, T. (2012b). Predictive regulatory models in drosophila melanogaster by integrative inference of transcriptional networks. *Genome Res*, 22.

Marbach, D., Schaffter, T., Mattiussi, C., and Floreano, D. (2009). Generating realistic in silico gene networks for performance assessment of reverse engineering methods. *Journal of computational biology : a journal of computational molecular cell biology*, 16(2):1–8.

Margolin, A. A., Nemenman, I., Basso, K., Wiggins, C., Stolovitzky, G., Favera, R. D., and Califano, A. (2006). ARACNE: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. *BMC Bioinformatics*, 7.

Meinshausen, N. and Bühlmann, P. (2010). Stability selection. *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, 72(4):417–473.

Meyer, P. E., Kontos, K., Lafitte, F., and Bontempi, G. (2007). Information-theoretic inference of large transcriptional regulatory networks. *EURASIP J Bioinform Syst Biol*, 2007.

Meyer, P. E., Marbach, D., Roy, S., and Kellis, M. (2010). Information-Theoretic Inference of Gene Networks Using Backward Elimination. In *BIOCOMP, International Conference on Bioinformatics and Computational Biology*, pages 700—-5.

Mitchell, M. (1998). *An introduction to genetic algorithms*. MIT press.

Obayashi, T. and Kinoshita, K. (2009). Rank of correlation coefficient as a comparable measure for biological significance of gene coexpression. *DNA Research*, 16(5):249–260.

Olsen, C., Meyer, P. E., and Bontempi, G. (2009). On the impact of entropy estimation on transcriptional regulatory network inference based on mutual information. *EURASIP J Bioinform Syst Biol*, 2009:308959.

Opgen-Rhein, R. and Strimmer, K. (2007). From correlation to causation networks: a simple approximate learning algorithm and its application to high-dimensional plant gene expression data. *BMC systems biology*, 1:37.

Pachter, L. (2014). Why I read the network nonsense papers. https://liorpachter.wordpress.com/2014/02/12/why-i-read-the-network-nonsense-papers/.

Pham, N. C., Haibe-Kains, B., Bellot, P., Bontempi, G., and Meyer, P. E. (2016). Study of Meta-Analysis Strategies for Network Inference using Information-Theoretic Approaches. *Biological Knowledge Discovery and Data Mining*.

Powers, D. M. W. (2011). Evaluation: From Precision, Recall and F-Measure To Roc, Informedness, Markedness & Correlation. *Journal of Machine Learning Technologies*, 2(1):37–63.

Prill, R. J., Marbach, D., Saez-Rodriguez, J., Sorger, P. K., Alexopoulos, L. G., Xue, X., Clarke, N. D., Altan-Bonnet, G., and Stolovitzky, G. (2010). Towards a rigorous assessment of systems biology models: The DREAM3 challenges. *PLoS ONE*, 5(2).

Pržulj, N. (2007). Biological network comparison using graphlet degree distribution. In *Bioinformatics*, volume 23.

Pržulj, N. (2010). Erratum to Biological network comparison using graphlet degree distribution. *Bioinformatics (Oxford, England)*, 26:853–854.

Pržulj, N., Corneil, D., and Jurisica, I. (2004). Modeling interactome: scale-free or geometric? *Bioinformatics*.

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

Rogers, S. and Girolami, M. (2005). A bayesian regression approach to the inference of regulatory networks from gene expression data. *Bioinformatics*, 21(14):3131–3137.

Roy, S., Ernst, J., Kharchenko, P. V., Kheradpour, P., Negre, N., Eaton, M. L., Landolin, J. M., Bristow, C. A., Ma, L., Lin, M. F., Washietl, S., Arshinoff, B. I., Ay, F., Meyer, P. E., Robine, N., Washington, N. L., Di Stefano, L., Berezikov, E., Brown, C. D., Candeias, R., Carlson, J. W., Carr, A., Jungreis, I., Marbach, D., Sealfon, R., Tolstorukov, M. Y., Will, S., Alekseyenko, A. A., Artieri, C., Booth, B. W., Brooks, A. N., Dai, Q., Davis, C. A., Duff, M. O., Feng, X., Gorchakov, A. A., Gu, T., Henikoff, J. G., Kapranov, P., Li, R., MacAlpine, H. K., Malone, J., Minoda, A., Nordman, J., Okamura, K., Perry, M., Powell, S. K., Riddle, N. C., Sakai, A., Samsonova, A., Sandler, J. E., Schwartz, Y. B., Sher, N., Spokony, R., Sturgill, D., van Baren, M., Wan, K. H., Yang, L., Yu, C., Feingold, E., Good, P., Guyer, M., Lowdon, R., Ahmad, K., Andrews, J., Berger, B., Brenner, S. E., Brent, M. R., Cherbas, L., Elgin, S. C. R., Gingeras, T. R., Grossman, R., Hoskins, R. A., Kaufman, T. C., Kent, W., Kuroda, M. I., Orr-Weaver, T., Perrimon, N., Pirrotta, V., Posakony, J. W., Ren, B., Russell, S., Cherbas, P., Graveley, B. R., Lewis, S., Micklem, G., Oliver, B., Park, P. J., Celniker, S. E., Henikoff, S., Karpen, G. H., Lai, E. C., MacAlpine, D. M., Stein, L. D., White, K. P., and Kellis, M. (2010). Identification of functional elements and regulatory circuits by Drosophila modENCODE. *Science*, 330(6012):1787–97.

Ruyssinck, J., Demeester, P., Dhaene, T., and Saeys, Y. (2016). Netter: re-ranking gene network inference predictions using structural network properties. *BMC Bioinformatics*, 17(1):76.

Salgado, H., Martínez-Flores, I., and Lopez-Fuentes, A. (2012). Extracting regulatory networks of Escherichia coli from RegulonDB. *Networks: Methods and . . . .*

Sangurdekar, D. and Srienc, F. (2006). A classification based framework for quantitative description of large-scale microarray data. *Genome*.

Schaffter, T., Marbach, D., and Floreano, D. (2011). GeneNetWeaver: In silico benchmark generation and performance profiling of network inference methods. *Bioinformatics*, 27(16):2263–2270.

Schimek, M. G., Budinská, E., Kugler, K. G., Švendová, V., Ding, J., and Lin, S. (2015). TopKLists: a comprehensive R package for statistical inference, stochastic aggregation, and visualization of multiple omics ranked lists. *in Genetics and . . . .*

Schölkopf, B., Tsuda, K., and Vert, J. P. (2004). Kernel Methods in Computational Biology. *Academic Radiology*, 11:400.

Schöning, U. (1988). Graph isomorphism is in the low hierarchy. *Journal of Computer and System Sciences*, 37(3):312–323.

Simoes, R. d. M. and Emmert-Streib, F. (2012). Bagging statistical network inference from large-scale gene expression data. *PLoS One*.

Simon, P. (2013). *Too Big to Ignore: The Business Case for Big Data*. John Wiley & Sons.

Sims, A. H., Smethurst, G. J., Hey, Y., Okoniewski, M. J., Pepper, S. D., Howell, A., Miller, C. J., and Clarke, R. B. (2008). The removal of multiplicative, systematic bias allows integration of breast cancer gene expression datasets - improving meta-analysis and prediction of prognosis. *BMC medical genomics*, 1:42.

Steuer, R., Kurths, J., Daub, C. O., Weise, J., and Selbig, J. (2002). The mutual information: detecting and evaluating dependencies between variables. *Bioinformatics*, 18(Suppl 2):S231–S240.

Stolovitzky, G., Monroe, D., and Califano, A. (2007). Dialogue on Reverse-Engineering Assessment and Methods. *Annals of the New York Academy of Sciences*, 1115(1):1–22.

Stolovitzky, G. a., Kundaje, A., Held, G. a., Duggar, K. H., Haudenschild, C. D., Zhou, D., Vasicek, T. J., Smith, K. D., Aderem, a., and Roach, J. C. (2005). Statistical analysis of mpss measurements: application to the study of lps-activated macrophage gene expression. *Proceedings of the*, 102(5):1402–7.

Tibshirani, R. J. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (*.

Van den Bulcke, T., Van Leemput, K., Naudts, B., van Remortel, P., Ma, H., and Verschoren, A. (2006). Syntren: a generator of synthetic gene expression data for design and analysis of structure learning algorithms. *BMC Bioinformatics*, 7.

Vert, J. and Kanehisa, M. (2002). Graph-driven feature extraction from microarray data using diffusion kernels and kernel CCA. *Advances in neural information*.

Vishwanathan, S. V. N., Borgwardt, K. M., Kondor, I. R., and Schraudolph, N. N. (2010). Graph Kernels. *Journal of Machine Learning Research*, 11:1201–1242.

Xiao, G., Wang, X., and Khodursky, A. (2011). Modeling three-dimensional chromosome structures using gene expression data. *Journal of the American*.

Xiao, Y., Dong, H., Wu, W., Xiong, M., Wang, W., and Shi, B. (2008). Structure-based graph distance measures of high degree of precision. *Pattern Recognition*, 41(12):3547–3561.

Zavlanos, M. M., Julius, A. A., Boyd, S. P., and Pappas, G. J. (2011). Inferring stable genetic networks from steady-state data. *Automatica*, 47(6):1113–1122.

Appendix A

Supplemental Material

A.1 Chapter 3

The results presented in Chapter 3 were obtained with the DREAM framework. The authors code is originally in MATLAB and we have ported to R. The package is available at https://github.com/paubellot/DREAM_Evaluation.

Table A.1 GRN methods and their R packages.

Method	R package
Aracne	Minet 3.30
C3net	c3net 1.1.1
CLR	Minet 3.30
Genie3	Netbenchmark 1.6
MRNET	Minet
PCIT	PCIT 1.5
Zscore	Netbenchmark 1.6

A.2 Chapter 4

How to use Netbenchmark package

In the last decade, several methods have tackled the challenge of reconstructing gene regulatory networks from gene expression data. Several papers have compared and evaluated the different network inference methods relying on simulated data.

This is a new comparison that assesses different methods in a high-heterogeneity data scenario which could reveal the specialization of methods for the different network types and data.

This package allows repeating the comparison between different network inference algorithms with only one line of code.

This package allows replication this comparison between the different networks inference algorithms with only one line of code.

Toy example for main benchmark:

```
library(netbenchmark)
## Loading required package: grndata
top5.aupr ← netbenchmark(methods="all",datasources.names = "Toy",
                                    local.noise=20,global.noise=10,
                                    no.topedges = 5,datasets.num = 2,
                                    noiseType=c("normal","lognormal"),
                                    experiments = 40,verbose=FALSE
                                    seed=1422976420)
## Warning:
## The specified number of experiments and datasets is bigger than
## the orginal number of experiments in the datasource: toy,
## sampling with replacement will be used
```

The first element of the returned list is the $AUPR_{5\%}$:

Origin	experiments	Aracne	C3Net	CLR	GeneNet	Genie3	Mrnet	mutrank	mrnetb	pcit	zscore	rand
toy	48	0.1230	0.0864	0.1269	0.0044	0.1150	0.1207	0.0897	0.1293	0.1221	0.0125	0.0035
toy	35	0.0703	0.0661	0.0676	0.0228	0.0943	0.0802	0.0439	0.0756	0.1003	0.0056	0.0094

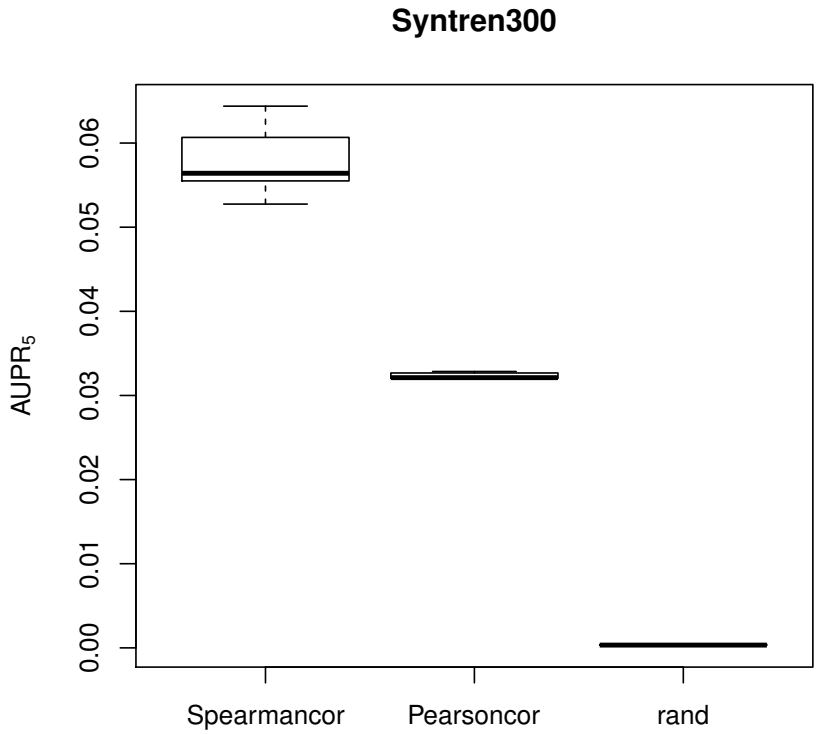
The package provides an easy way to compare new techniques with state-of-the-art ones and to make new different benchmarks in the future. First, define the wrapper functions:

```
Spearmancor ← function(data){
        cor(data,method="spearman")
}
Pearsoncor ← function(data){
        cor(data,method="pearson")
}
```

Note that the wrapper function returns a matrix which is the weighted adjacency matrix of the network inferred by the algorithm and that the columns and rows are named.

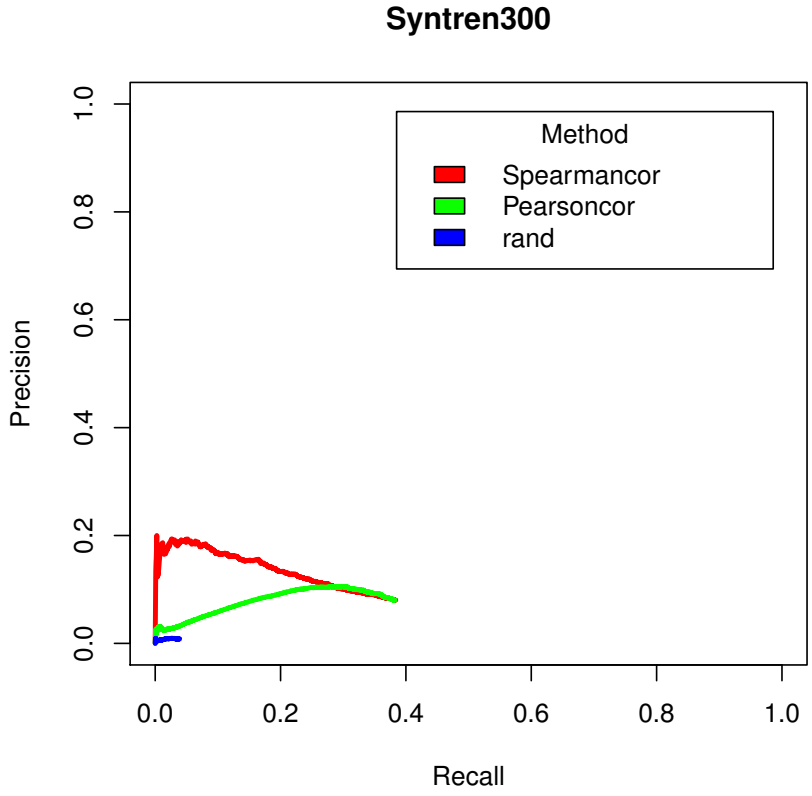
Evaluate five times these two simple inference methods with syntren300 datasource:

```
res ← netbenchmark(datasources.names="syntren300",no.topedges = 5,
        methods=c("Spearmancor","Pearsoncor"),verbose=FALSE)
aupr ← res[[1]][,-(1:2)]
# Make a boxplot of the AUPR5% results:
boxplot(aupr, main="Syntren300",ylab=expression('AUPR'[5]))
```



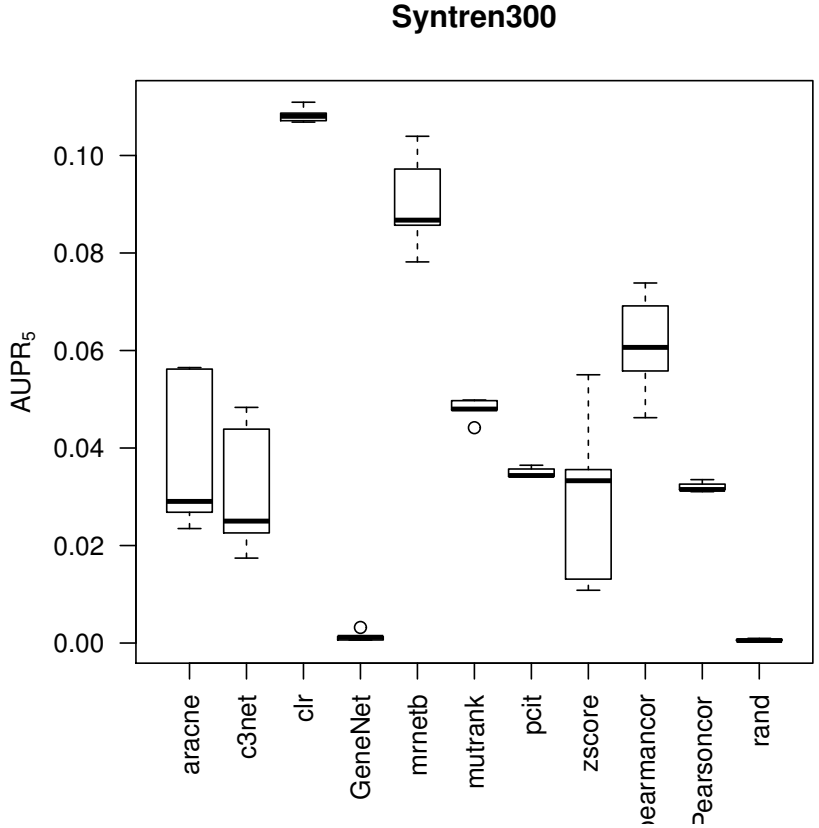
Plot the mean Precision-Recall curves:

```
PR ← res[[5]][[1]]
col ← rainbow(3)
plot(PR$rec[,1],PR$pre[,1],type="l",lwd=3,col=col[1],
  xlab="Recall",ylab="Precision",main="Syntren300",
  xlim=c(0,1),ylim=c(0,1))
lines(PR$rec[,2],PR$pre[,2],type="l",lwd=3,col=col[2])
lines(PR$rec[,3],PR$pre[,3],type="l",lwd=3,col=col[3])
legend("topright", inset=.05,title="Method",colnames(PR$rec),
  fill=col)
```



We can also compare these two simple inference methods with the fast network inference algorithms using syntren300 datasource:

```
comp ← netbenchmark(datasources.names="syntren300",verbose=FALSE
  no.topedges = 5, methods=c("all.fast","Spearmancor",
  "Pearsoncor"))
aupr ← comp[[1]][,-(1:2)]
# Make a boxplot of the AUPR5% results:
# make the name look prety
library("tools")
colnames(aupr) ← sapply(colnames(aupr),file_path_sans_ext)
boxplot(aupr, main="Syntren300", ylab=expression('AUPR'[5]))
```


Syntren300
AUPR5
0.00
0.02
0.04
0.06
0.08
0.10
aracne
c3net
clr
GeneNet
mrnetb
mutrank
pcit
zscore
earmancor
earsoncor
rand