

Guido Sanguinetti
Vân Anh Huynh-Thu *Editors*



Gene Regulatory Networks

Methods and Protocols

METHODS IN MOLECULAR BIOLOGY

Series Editor

John M. Walker

School of Life and Medical Sciences

University of Hertfordshire

Hatfield, Hertfordshire, AL10 9AB, UK

For further volumes:
<http://www.springer.com/series/7651>

Gene Regulatory Networks

Methods and Protocols

Edited by

Guido Sanguinetti

School of Informatics, University of Edinburgh, Edinburgh, UK

Vân Anh Huynh-Thu

Department of Electrical Engineering and Computer Science, University of Liège, Liège, Belgium

Editors

Guido Sanguinetti
School of Informatics
University of Edinburgh
Edinburgh, UK

Vân Anh Huynh-Thu
Department of Electrical Engineering
and Computer Science
University of Liège
Liège, Belgium

ISSN 1064-3745 ISSN 1940-6029 (electronic)
Methods in Molecular Biology ISBN 978-1-4939-8881-5 ISBN 978-1-4939-8882-2 (eBook)
<https://doi.org/10.1007/978-1-4939-8882-2>

Library of Congress Control Number: 2018962962

© Springer Science+Business Media, LLC, part of Springer Nature 2019

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Humana Press imprint is published by the registered company Springer Science+Business Media, LLC part of Springer Nature.

The registered company address is: 233 Spring Street, New York, NY 10013, U.S.A.

Preface

High-throughput technologies have brought about a revolution in molecular biology. Over the last two decades, the research paradigm has moved from a characterization of individual genes and their function increasingly toward a systems-level appreciation of how the complex interactions between multiple genes shape the dynamics and functions of biological systems. At the same time, the computational and statistical challenges posed by the interpretation of such data have motivated an exciting cross-fertilization between the disciplines of biology and the mathematical and computational sciences, leading to the birth of the interdisciplinary field of systems biology.

A crucial computational task in systems biology is the so-called *reverse engineering* task: given observations of multiple biological features (e.g., protein levels) across different time points/conditions, determine computationally the interaction structure (the *network*) that best explains the data. Within the context of modeling gene expression, this is the task of inferring *gene regulatory networks* (GRNs) from data.

GRN inference has been a major challenge in systems biology for nearly two decades, and, while challenges still abound, it is rapidly reaching maturity both in terms of the concepts involved, and in terms of the software tools available. In this book, we aim to take stock of the situation, providing an overview of methods that cover the majority of recent developments, as well as indicating the path forward for future developments.

The book opens with a tutorial overview of the main biological and mathematical concepts and a survey of the current software landscape. This is meant to be an entry level chapter, which the interested, graduate-level practitioner (either computational or biological) can consult as a rough guide to the concepts described more in detail in the more technical chapters in the book. The next two chapters then focus on Bayesian methods to infer networks from time varying data, while Chapters 4 and 5 describe how to attempt to extract causal information (as opposed to purely correlative) from biological data. Chapters 6 and 7 describe network inference techniques in the presence of multiple heterogeneous data sets, while Chapters 8 and 9 focus on nonparametric and hybrid statistical methods for network inference. The following five Chapters 10–14 focus on the idea of inference of different (but related) networks, arising either from intrinsic heterogeneity (such as in single-cell data) or due to multiple conditions being assayed, and further explore concepts of differential networks and network stability. Finally, the last two chapters focus more on a mechanistic view of the biological process, covering methods for exploring networks within large, mechanistic models of biological dynamics.

As most books, this volume presents an incomplete snapshot of an evolving field, and, given the considerable research activity in this area, it is clear that we can look forward to considerable progress within the next decades. Our hope is that this collection will be instrumental in assessing the current state of the art and in focusing research on the common challenges faced by the field.

*Edinburgh, UK
Liège, Belgium*

*Guido Sanguinetti
Vân Anh Huynh-Thú*

Contents

<i>Preface</i>	v
<i>Contributors</i>	ix
1 Gene Regulatory Network Inference: An Introductory Survey	1
<i>Vân Anh Huynh-Thu and Guido Sanguinetti</i>	
2 Statistical Network Inference for Time-Varying Molecular Data with Dynamic Bayesian Networks	25
<i>Frank Dondelinger and Sach Mukherjee</i>	
3 Overview and Evaluation of Recent Methods for Statistical Inference of Gene Regulatory Networks from Time Series Data	49
<i>Marco Grzegorczyk, Andrej Aderhold and Dirk Husmeier</i>	
4 Whole-Transcriptome Causal Network Inference with Genomic and Transcriptomic Data	95
<i>Lingfei Wang and Tom Michoel</i>	
5 Causal Queries from Observational Data in Biological Systems via Bayesian Networks: An Empirical Study in Small Networks	111
<i>Alex White and Matthieu Vignes</i>	
6 A Multiattribute Gaussian Graphical Model for Inferring Multiscale Regulatory Networks: An Application in Breast Cancer	143
<i>Julien Chiquet, Guillem Rigail, and Martina Sundqvist</i>	
7 Integrative Approaches for Inference of Genome-Scale Gene Regulatory Networks	161
<i>Alireza Fotouhi Siahpirani, Deborah Chasman, and Sushmita Roy</i>	
8 Unsupervised Gene Network Inference with Decision Trees and Random Forests	195
<i>Vân Anh Huynh-Thu and Pierre Geurts</i>	
9 Tree-Based Learning of Regulatory Network Topologies and Dynamics with Jump3	217
<i>Vân Anh Huynh-Thu and Guido Sanguinetti</i>	
10 Network Inference from Single-Cell Transcriptomic Data	235
<i>Helena Todorov, Robrecht Cannoodt, Wouter Saelens, and Yvan Saeys</i>	
11 Inferring Gene Regulatory Networks from Multiple Datasets	251
<i>Christopher A. Penfold, Iulia Gherman, Anastasiya Sybirna, and David L. Wild</i>	
12 Unsupervised GRN Ensemble	283
<i>Pau Bellot, Philippe Salembier, Ngoc C. Pham, and Patrick E. Meyer</i>	
13 Learning Differential Module Networks Across Multiple Experimental Conditions	303
<i>Pau Erola, Eric Bonnet, and Tom Michoel</i>	

14	Stability in GRN Inference	323
	<i>Giuseppe Jurman, Michele Filosi, Roberto Visintainer, Samantha Riccadonna, and Cesare Furlanello</i>	
15	Gene Regulatory Networks: A Primer in Biological Processes and Statistical Modelling	347
	<i>Olivia Angelin-Bonnet, Patrick J. Biggs and Matthieu Vignes</i>	
16	Scalable Inference of Ordinary Differential Equation Models of Biochemical Processes	385
	<i>Fabian Fröhlich, Carolin Loos, and Jan Hasenauer</i>	
	<i>Index</i>	423

Contributors

- ANDREJ ADERHOLD • *School of Mathematics and Statistics, University of Glasgow, Glasgow, UK*
- OLIVIA ANGELIN-BONNET • *Institute of Fundamental Sciences, Palmerston North, New Zealand*
- PAU BELLOT • *Centre for Research in Agricultural Genomics (CRAG), CSIC-IRTA-UAB-UB Consortium, Bellaterra, Barcelona, Spain*
- PATRICK J. BIGGS • *Institute of Fundamental Sciences, Palmerston North, New Zealand; School of Veterinary Science, Massey University, Palmerston North, New Zealand*
- ERIC BONNET • *Centre Nationale de Recherche en Génomique Humaine, Institut de Biologie François Jacob, Direction de la Recherche Fondamentale, CEA, Evry, France*
- ROBRECHT CANNOODT • *Data Mining and Modelling for Biomedicine, VIB Center for Inflammation Research, Ghent, Belgium; Center for Medical Genetics, Ghent University Hospital, Ghent, Belgium*
- DEBORAH CHASMAN • *Wisconsin Institute for Discovery, University of Wisconsin-Madison, Madison, WI, USA*
- JULIEN CHIQUET • *UMR MIA-Paris, AgroParisTech, INRA, Université Paris-Saclay, Paris, France*
- FRANK DONDELINGER • *Lancaster Medical School, Lancaster University, Lancaster, UK*
- PAU EROLA • *Division of Genetics and Genomics, The Roslin Institute, The University of Edinburgh, Midlothian, Scotland, UK*
- MICHELE FILOSI • *CIBIO, University of Trento, Trento, Italy*
- FABIAN FRÖHLICH • *Institute of Computational Biology, Helmholtz Zentrum München, Neuherberg, Germany; Center for Mathematics, Technische Universität München, Garching, Germany*
- CESARE FURLANELLO • *Fondazione Bruno Kessler, Trento, Italy*
- PIERRE GEURTS • *Department of Electrical Engineering and Computer Science, University of Liège, Liège, Belgium*
- IULIA GHerman • *Warwick Integrative Synthetic Biology Centre, School of Engineering, University of Warwick, Coventry, UK*
- MARCO GRZEGORCZYK • *Johann Bernoulli Institute, University of Groningen, Groningen, The Netherlands*
- JAN HASENAUER • *Institute of Computational Biology, Helmholtz Zentrum Muenchen, Neuherberg, Germany; Center for Mathematics, Technische Universität München, Garching, Germany*
- DIRK HUSMEIER • *School of Mathematics and Statistics, University of Glasgow, Glasgow, UK*
- VÂN ANH HUYNH-THU • *Department of Electrical Engineering and Computer Science, University of Liège, Liège, Belgium*
- GIUSEPPE JURMAN • *Fondazione Bruno Kessler, Trento, Italy*
- CAROLIN LOOS • *Institute of Computational Biology, Helmholtz Zentrum München, Neuherberg, Germany; Center for Mathematics, Technische Universität München, Garching, Germany*

- PATRICK E. MEYER • *Bioinformatics and Systems Biology (BioSys) Unit, Université de Liège, Liège, Belgium*
- TOM MICHOEL • *Division of Genetics and Genomics, The Roslin Institute, The University of Edinburgh, Midlothian, Scotland, UK; Computational Biology Unit, Department of Informatics, University of Bergen, Bergen, Norway*
- SACH MUKHERJEE • *German Center for Neurodegenerative Diseases (DZNE), Bonn, Germany*
- CHRISTOPHER A. PENFOLD • *Wellcome/CRUK Gurdon Institute, University of Cambridge, Cambridge, UK*
- NGOC C. PHAM • *Bioinformatics and Systems Biology (BioSys) Unit, Universite de Liège, Liège, Belgium*
- SAMANTHA RICCADONNA • *Fondazione Edmund Mach, San Michele all'Adige, Italy*
- GUILLEM RIGAILL • *Institute of Plant Sciences Paris-Saclay, UMR 9213/UMR1403, CNRS, INRA, Université Paris -Sud, Université d'Evry, Université Paris-Diderot, Sorbonne Paris-Cité, Paris, France; Laboratoire de Mathématiques et Modélisation d'Evry (LaMME), Université d'Evry, Val d'Essonne, UMR CNRS 8071, ENSIIE, USC INRA, Paris, France*
- SUSHMITA ROY • *Wisconsin Institute for Discovery, University of Wisconsin-Madison, Madison, WI, USA; Department of Biostatistics and Medical Informatics, University of Wisconsin-Madison, Madison, WI, USA*
- WOUTER SAELENS • *Data Mining and Modelling for Biomedicine, VIB Center for Inflammation Research, Ghent, Belgium; Department of Applied Mathematics, Computer Science and Statistics, Ghent University, Ghent, Belgium*
- YVAN SAEYS • *Data Mining and Modelling for Biomedicine, VIB Center for Inflammation Research, Ghent, Belgium; Department of Applied Mathematics, Computer Science and Statistics, Ghent University, Ghent, Belgium*
- PHILIPPE SALEMBIER • *Universitat Politècnica de Catalunya, Barcelona, Spain*
- GUIDO SANGUINETTI • *School of Informatics, University of Edinburgh, Edinburgh, UK*
- ALIREZA FOTUHI SIAHIRANI • *Wisconsin Institute for Discovery, University of Wisconsin-Madison, Madison, WI, USA; Department of Computer Sciences, University of Wisconsin-Madison, Madison, WI, USA*
- MARTINA SUNDQVIST • *UMR MIA-Paris, AgroTechParis, INRA, Université Paris-Saclay, Paris, France*
- ANASTASIYA SYBIRNA • *Wellcome/CRUK Gurdon Institute, University of Cambridge, Cambridge, UK; Wellcome/MRC Cambridge Stem Cell Institute, University of Cambridge, Cambridge, UK; Physiology, Development and Neuroscience Department, University of Cambridge, Cambridge, UK*
- HELENA TODOROV • *Data Mining and Modelling for Biomedicine, VIB Center for Inflammation Research, Ghent, Belgium; Department of Applied Mathematics, Computer Science and Statistics, Ghent University, Ghent, Belgium; Centre International de Recherche en Infectiologie, Inserm, U1111, Université Claude Bernard Lyon 1, CNRS, UMR5308, École Normale Supérieure de Lyon, Univ Lyon, Lyon, France*
- MATTHIEU VIGNES • *Institute of Fundamental Sciences, Massey University, Palmerston North, New Zealand*
- ROBERTO VISINTAINER • *CIBIO, University of Trento, Trento, Italy*
- LINGFEI WANG • *Division of Genetics and Genomics, The Roslin Institute, The University of Edinburgh, Midlothian, Scotland, UK*

ALEX WHITE • *Institute of Fundamental Sciences, Massey University, Palmerston North, New Zealand*

DAVID L. WILD • *Department of Statistics and Systems Biology Centre, University of Warwick, Coventry, UK*



Chapter 1

Gene Regulatory Network Inference: An Introductory Survey

Vân Anh Huynh-Thu and Guido Sanguinetti

Abstract

Gene regulatory networks are powerful abstractions of biological systems. Since the advent of high-throughput measurement technologies in biology in the late 1990s, reconstructing the structure of such networks has been a central computational problem in systems biology. While the problem is certainly not solved in its entirety, considerable progress has been made in the last two decades, with mature tools now available. This chapter aims to provide an introduction to the basic concepts underpinning network inference tools, attempting a categorization which highlights commonalities and relative strengths. While the chapter is meant to be self-contained, the material presented should provide a useful background to the later, more specialized chapters of this book.

Key words Gene regulatory networks, Network inference, Network reverse-engineering, Unsupervised inference, Data-driven methods, Probabilistic models, Dynamical models

1 Introduction: The Biological Problem

The discovery of the biochemical basis of life is one of the great scientific success stories of the past century. Remarkably, the amazing diversity of life can be explained from a relatively small set of biochemical actors and their interactions. Heritable information is stored in chromosomes, very long polymers of double stranded DNA, which encode information as a sequence of symbols from a four letter alphabet, A, C, G, T, the nucleotides constituting the building blocks of DNA. Just as DNA is the universal information storage medium, information flow also follows a consistent biochemical pathway across all organisms. Stored information can be dynamically read through the process of *gene expression*, a two-step process whereby DNA gets transcribed into RNA, an intermediate, single stranded polymer of nucleic acids (with the T nucleotide replaced by uracil, U), and RNA is subsequently translated into proteins, macromolecules formed of amino-acids which carry out most cellular functions. This process is of such

fundamental importance in biology to have earned the moniker of *central dogma of molecular biology* [1]; it constitutes the universal flow of information across all living creatures (the most notable exception being reverse transcription of viral RNA).

Not all DNA within a cell codes for proteins, and not all DNA is transcribed; indeed, genes, the stretches of DNA encoding some functionality (either protein or other classes of functional RNAs), constitute a small fraction of the overall genome. One of the surprising outcomes of the major genome-sequencing projects at the turn of the millennium was the realization of just how little DNA codes for proteins (approximately 3% of the human genome, with similar percentages in other higher eukaryotes). Moreover, the number of genes in different organisms is relatively constant across scales of organismal complexity: the humble baker's yeast *Saccharomyces cerevisiae* has approximately 6000 genes, more than a quarter the number of genes in the human genome. Apart from raising overdue questions on our anthropocentric worldview, the natural corollary of this observation is that complexity in life does not arise from a disparity in the number of available components (genes), but from the nature and dynamics of the interactions between such components.

Measuring interactions is difficult within live cells. On the other hand, measuring components' abundances (e.g., mRNA levels) is considerably easier, and technological advances within the last two decades have enabled increasingly large-scale measurements of gene expression at steadily decreasing costs. This trend has provided a powerful motivation to attempt to reconstruct *computationally* the interaction structures underpinning patterns of gene expression: these interactions collectively are denoted as *Gene Regulatory Networks* (GRNs). Reconstructing such networks has been a central effort of the interdisciplinary field of *Systems Biology*.

In this chapter, we provide a tutorial overview of the field, aimed at a novice computational scientist or biologist wishing to approach the subject. We first provide a brief introduction to the core biological concepts, as well as the main sources of data currently available. We then introduce the core mathematical concepts, and briefly attempt a categorization of the main methodological approaches available. This chapter is intended to be a self-contained introduction which will provide some essential background to the book; later chapters will describe more advanced concepts, and associated tools for GRN reconstruction across the breadth of their biological application.

1.1 Mechanisms of Gene Regulation

The molecular bases of the transcription process have been intensely studied over the last 60 years. Many excellent monographs are available on the subject; we refer the reader in particular to the classic books by Ptashne and collaborators [2, 3]

(see also this recent review [4] for a historical perspective). Here we give a brief intuitive description of the process, taking, as an illustrative example, the transcriptional response of the bacterium *Escherichia coli* in response to changes in oxygen availability (see ref. [5] for a modern review of this field). Transcription is carried out by the enzyme RNA polymerase (RNAP), that slides along the DNA, opening the double strand and producing a faithful RNA copy of the gene. The rate of recruitment of RNAP at a gene can be modulated by the presence or absence of specific *transcription factor* (TF) proteins, which contain a DNA-binding module that enables them to recognize specific DNA-sequence signals near the start of genes (promoter regions). The classical view of gene regulation holds that changes in cellular state are orchestrated by changes in binding by TFs.

For example, in *E. coli*, oxygen withdrawal leads to dimerization of the master regulator protein Fumarate Nitrate Reductase (FNR); FNR dimers (but not monomers) can bind specifically to DNA, and change the rate of recruitment of RNAP at the FNR target genes, thereby changing their levels of expression to enable the cell to adapt to the changed conditions. However, FNR is not the only regulator responding to changes in oxygen availability: another master regulator, the two component system ArcAB, also senses oxygen changes, albeit through a different mechanism, and changes its binding to hundreds of genes as a result. FNR and ArcAB share many targets, and through their combined action they can give rise to highly complex dynamics [6, 7].

Two important observations can be made from the previous discussion. Firstly, the regulation of gene expression levels is enacted through the action of gene products themselves: therefore, in principle, one may hope to be able to describe the dynamics of gene expression as an *autonomous* system. Secondly, even in the simple case of the bacterium *Escherichia coli*, regulation of gene expression is a complex process, likely to involve the interactions of several molecular players.

In higher organisms, the basic components of the transcriptional regulatory machinery are remarkably similar. However, many more levels of regulatory control are present: in particular, chemical modifications of the DNA itself (in particular methylation of C nucleotides) and of the structural histone proteins, around which DNA is wound, can affect the structural properties of the DNA, and hence the local accessibility to the transcriptional machinery. Such effects, collectively known as *epigenetic modifications*, have strong associations with transcription [8–11], and are generally thought to encode processes of cellular memory associated with long-term adaptation or cell-type differentiation.

Finally, while we have primarily focused on transcription, subsequent steps of gene expression are also tightly regulated: RNA processing, translation, and RNA and protein degradation all pro-

vide additional levels at which gene expression can be controlled. Mechanisms of post-transcriptional control of gene expression are less well explored, but it is widely believed that such processes, mostly effected through proteins or RNAs binding to RNA targets, may be as prevalent as transcriptional controls [12, 13] (see also Chapter 15 for perspectives on incorporating post-transcriptional regulation in GRN inference). Therefore, while a gene may have no effect on the expression of another gene at the RNA level, it may well be extremely important for the protein expression.

1.2 *High-Throughput Measurements Techniques*

As we have seen in the previous subsection, the control of gene expression is effected through the action of gene products themselves. Naturally, in order to discover and quantify such controls, one must then be able to simultaneously measure the levels of expression of multiple genes. Measurements of gene expression have progressed dramatically in the last 20 years, with technological advances driving a seemingly unstoppable expansion in the scope of such experiments.

Proteins are the final product of the process of gene expression. Methods based on quantitative mass spectrometry have been highly effective in quantifying hundreds to thousands of proteins within samples. Despite that, intrinsic limits to their sensitivity and a relatively complex analysis pipeline mean that such methods do not yet reach the comprehensiveness of transcriptomic measurements [14].

Methods for assaying RNA levels have progressed immensely in the last two decades. Microarray technology first provided enormous impetus to the field in the late 1990s [15]. Microarrays consist of thousands of short fragments of DNA (probes) arranged on a substrate chip (usually glass or some other inert material); by designing probes to complement thousands of genomic regions from target organisms, one can obtain a readout of the (steady state) concentration of thousands of transcripts within a population of cells.

Microarrays represented a turning point in our ability to comprehensively measure genetic materials; however, the design of the probes implicitly defines what can be measured, biasing the assay and limiting the scope for discovery of unexpected biological facts, e.g., previously unobserved transcripts. Next generation sequencing (NGS) technologies proved revolutionary in this context. NGS provides a massively parallel implementation of DNA sequencing protocols, which enabled it to dramatically reduce costs and expand throughput. RNA-seq is the main NGS technology used to measure transcript abundances [16]: RNA from a population of cells is reverse transcribed (usually after a selective enrichment process to filter out highly abundant ribosomal RNAs), fragmented and the resulting complementary DNA is sequenced and mapped to a reference genome. The number of

fragments mapped to a particular gene, suitably normalized [17], then gives a raw measurement of gene expression.

One of the major success stories of NGS technologies is the ability of combining them with a variety of biochemical assays, greatly expanding the scope of potential measurements. Of particular relevance for GRNs is the ability to select fragments of DNA bound to specific proteins by a process called immunoprecipitation. Genomic material is fragmented, and an antibody specific to a particular DNA-binding protein is added, allowing separation by centrifugation of the protein. The bound DNA fragments are then released, sequenced, and mapped to a reference genome to identify where the protein was bound. This technique, Chromatin Immuno-Precipitation followed by sequencing (ChIP-seq), has been instrumental in obtaining *in vivo* mappings of possible regulatory relationships [18].

2 Introduction: The Mathematical Formulation

In the previous section, we have given a condensed tour of the fundamental biological problem addressed in this book. We have seen that interactions between gene products are the fundamental processes underpinning the cell’s ability to modulate gene expression. High-throughput measuring techniques paved the way to the use of computational statistics techniques to reconstruct statistically such interactions, a process sometimes called *reverse engineering*. In this section we introduce some of the fundamental mathematical concepts common to all methods for reverse engineering GRNs, *see*, e.g., [19] for a more comprehensive review of these concepts.

Definition 1 (Network). A (directed) network or graph is a pair (V, E) where V is a finite set of *vertices* (or nodes) and E is a set of *edges* (or arcs) connecting the vertices. If \mathcal{I} is a set indexing the nodes, the set of edges is a subset of the Cartesian product $E \subset \mathcal{I} \times \mathcal{I}$, with element (ij) indicating the presence of an edge between node i and node j . An *undirected* network is a network where the edge set is symmetric under swapping the indices of the nodes, i.e., whenever edge (ij) exists also edge (ji) exists.

Within the GRN context, network nodes universally represent the expression level of genes. Edges are intuitively linked to associations between genes, but the precise meaning of an edge depends on the mathematical model of the system. Networks are abstract representations of systems, and per se do not have a semantic interpretation that could link the network to node behaviors, e.g., their collective dynamics. Nevertheless, the structure of a network (the *topology*) can provide an intuitively appealing visualization of the system, and often be informative in itself. Informally, the aim of a network abstraction is to condense in a simple representation

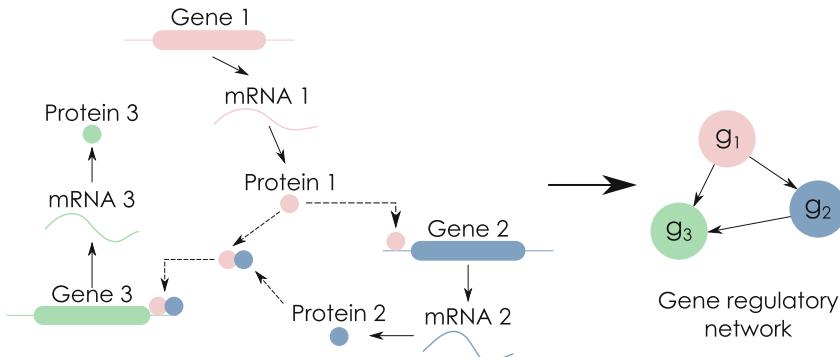


Fig. 1 A cartoon schematic of a gene regulatory network. A complex biophysical model describes the interaction between three genes, involving both direct regulation (gene 2 by gene 1) and combinatorial regulation via complex formation (gene 3 by genes 1 and 2). The abstracted structure of the system is given in the (directed) network on the right

the complexity of interactions underpinning gene expression, see Fig. 1 for a cartoon representation. One of the most important quantities in this regard is the *degree* of a node, i.e., the number of edges that are attached to the node, and the *degree distribution* of the network, i.e., the empirical distribution of degrees across all nodes in the network. Degree distributions often encode intuitively interpretable properties of networks such as the presence of hubs or the ability to reach rapidly any node from any starting node, and in many cases they can be related to distinct stochastic mechanisms by which the network may arise. In the case of directed networks, one may further distinguish between *in-degree* (also called *fan-in*), the number of edges terminating at a node, and *out-degree* (also called *fan-out*), the number of edges starting at a node.

Finally, in many cases the bare topological description is insufficient in capturing aspects of interest, such as the different importance of different edges. To obviate this problem, one can consider *weighted* networks, where each edge is associated with a real number, its weight. We will see that in most cases reconstructed networks, the topic of this book, arise naturally as weighted networks, where the weight is intuitively associated with the support that the data offers for the existence of an edge. Weighted networks are often visualized as networks with edges of different thickness, retaining the visual immediacy of the network abstraction but effectively conveying more information. A schematic example of a standard graphical representation for directed, undirected, and weighted networks is given in Fig. 2.

Network science is a rich interdisciplinary field of research, and this whistle-stop tour of the basic mathematical concepts cannot do justice to such a field. Nevertheless, we now have the essential tools to understand, at least at a high level, many of the common strategies for reconstructing GRNs.

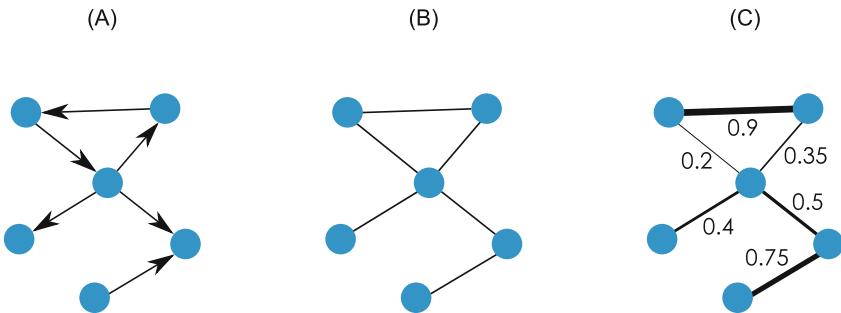


Fig. 2 Examples of network types: directed (a), undirected (b), and weighted (c), where the weights are represented by edge thickness. Note that a weighted network can be directed or undirected

3 Data-Driven Methods

The first class of GRN reconstruction methods considers a fully connected network and associates a weight to each edge by estimating gene dependencies directly from the data. The output of such methods is therefore a weighted network, which can be suitably thresholded to yield the topology of the network. Such methods are generally simple to implement, computationally efficient (they scale with the number of possible edges, which is quadratic in the number of nodes), and have proved often remarkably accurate and effective. For these reasons, some of the most popular tools for GRN inference pertain to this category.

3.1 Correlation Networks

The simplest score that one may associate to a pair of vector-valued measurements is their correlation. This is computed in the following way: given two zero-mean vectors \mathbf{v}_i and \mathbf{v}_j , the (Pearson) correlation between the vectors is given by

$$\text{corr}(\mathbf{v}_i, \mathbf{v}_j) = \rho_{ij} = \frac{\mathbf{v}_i \cdot \mathbf{v}_j}{\|\mathbf{v}_i\| \|\mathbf{v}_j\|} \quad (1)$$

where \cdot indicates the scalar product and $\|\mathbf{v}_i\|$ is the Euclidean norm of vector \mathbf{v}_i (square root of the sum of the squares of the elements). Practically, given a set of N expression measurements (e.g., different conditions) for G genes, one arranges them into a data matrix $D \in \mathbb{R}^{N \times G}$. Computing correlations between the columns of D yields a $G \times G$ matrix of pairwise gene correlations, which can be taken as the weights of an undirected network and suitably thresholded to obtain a network structure. Variations of this approach involve taking a different measure of correlation (e.g., Kendall's or Spearman's correlation), or raising each correlation to a power to effectively filter out spurious low correlations (weighted correlations).

Correlation networks are extremely simple to implement; their complexity scales linearly with the number of experiments and quadratically with the number of genes, so they can be easily deployed on genome-wide studies with very high numbers of experiments. The assumption that interacting genes should have correlated expression is biologically plausible, and methods such as WGCNA (weighted gene coexpression network analysis [20]) have proved consistently reliable and are widely adopted.

Correlation networks however also have some limitations. First, two genes might appear correlated not because they genuinely interact, but because of the effect of a third gene (or several other genes). For example, a high correlation might appear between two genes that share a common regulator. Correlation networks are also unable to distinguish between direct and indirect interactions: if gene i regulates gene j which in turn regulates gene k , it is likely that there will be a high correlation between gene i and gene k . Correlation networks are therefore vulnerable to false positives. In this respect, *partial correlation networks* (see Subheading 4.1) offer a conceptually appealing solution to the problem, at the cost of some additional assumptions. Another drawback of correlation networks is that limited sample sizes (which are common in small to medium scale studies) may produce apparent high correlations which are not statistically significant. Furthermore, Pearson correlation is a linear measure of correlation, therefore nonlinear regulatory effects might easily be missed, creating a vulnerability to false negatives as well.

Since the correlation is a symmetric metric, correlation networks are intrinsically undirected. Also, correlation is purely a measure of statistical association; therefore, these models are not predictive, in the sense that knowledge of some node values would not allow us to make a quantitative prediction about the remaining nodes.

3.2 Information Theoretic Scores

As we have seen before, the linearity of Pearson correlation may limit its suitability to capture complex regulatory relations. To obviate this problem, several groups have considered alternative scores based on information theory. The main mathematical concept is the mutual information, defined as follows. Let X and Y be two discrete random variables, and let $P(X, Y)$ be their joint probability distribution. The mutual information between the two random variables is then defined as

$$\begin{aligned} \text{MI}[X, Y] &= \sum_{x_i, y_j} P(x_i, y_j) \log \frac{P(x_i, y_j)}{P(x_i)P(y_j)} \\ &= \sum_{x_i, y_j} P(x_i, y_j) \log \frac{P(x_i|y_j)}{P(x_i)} \end{aligned} \quad (2)$$

where x_j and y_j are the values the two random variables can take, and $P(X)$ (resp. $P(Y)$) is the marginal distribution obtained by summing out the values of Y (resp. X) in the joint distribution. Intuitively, the mutual information quantifies the degree of dependence of the two random variables: it is zero when the two random variables are independent (as is clear from the second formulation in Eq. (2)), and, when the two variables are deterministically linked, it returns the entropy of the marginal distribution. The mutual information is still a symmetric score, so mutual information networks are naturally undirected. Nevertheless, it can accommodate more subtle dependencies than the linear correlation score in (1), therefore potentially catering for a broader class of regulatory interactions.

In the GRN context, the idea is to replace the probability distributions in (2) with empirical distributions (estimated from the samples) of gene expression levels for each pair of genes. This gives a weight to each possible edge within a fully connected, weighted undirected network; thresholding at a user-defined parameter then returns a network topology called *relevance network* [21]. A number of methods have been proposed to filter out indirect or spurious links in relevance networks, the most popular methods being ARACNE [22], CLR [23], and MRNET [24].

Mutual information networks are among the most widely used GRN inference methods. They scale to genome-wide networks, even if they are slightly more computationally intensive than correlation-based methods, as their computational complexity is quadratic in the number of genes and samples. However, they also stop short of providing a predictive framework. Furthermore, estimation of the joint probabilities in Eq. (2) might be highly sensitive to noise when the sample size is medium-small.

3.3 Regression-Based Methods

An alternative approach to quantify the dependence of two variables is to predict one from the other. In the simplest case, one may try a linear regression approach, where the slope of the regression line may be used to quantify the strength of the dependence. In a GRN context this would amount to regressing each gene in turn against all other genes in order to obtain network weights. Thus, for every gene \mathcal{g} , denoting by x_{gi} its expression level in sample i , we would solve the regression problem

$$x_{gi} = \sum_{j \neq g} w_j x_{ji} + \varepsilon_i, \quad (3)$$

with ε_i a noise term, and use the resulting weight w_j as the weight associated with the network edge between gene j and gene \mathcal{g} . Notice that in this case the regression formulation naturally gives a direction to the network (even though bidirectional edges are clearly possible).

This idea is at the core of several successful methods for GRN reconstruction. TIGRESS [25] adopts directly the framework of Eq. (3), introducing a L1 regularization penalty, which forces some of the weights w_j to be strictly zero, to ensure the identifiability of the system (in general, unless the number of samples is higher than the number of genes, these are overparametrized systems). An alternative idea is to replace the linear regression model of (3) with a more flexible, non-parametric regression model. GENIE3 [26], another widely used method, and subsequent developments [27, 28] also follow this strategy, replacing linear regression with an ensemble of regression trees. The score for the edge (jg) is then the importance of gene j in the tree model predicting gene g , which can be interpreted as the fraction of variance of the expression of gene g that can be explained by gene j . Finally, the regression approach is also extremely popular to handle time series data, with the simple modification that the expression of gene g at time t is regressed against the expression of the other genes *at the previous time point $t - 1$ (autoregressive model)* [29]. In this book, regression-based methods are discussed in Chapters 8 and 9.

Methods based on a regression approach are among the most popular and scalable approaches for reconstructing directed networks. Compared to other data-driven methods, they are generally computationally more intensive, but they have predictive capability, in the sense that, given the expression of a subset of genes, one may in principle predict the expression levels of the remaining genes. Moreover, regression-based methods are potentially able to capture high-order conditional dependencies between gene expression patterns, while correlation- and mutual information-based methods only focus on pairwise dependencies. Practically, the identifiability of regression models from limited data may be problematic: different genes often have strongly correlated expression patterns, and (regularized) regression with correlated covariates is notoriously prone to spurious results.

4 Probabilistic Models

The data-driven-based methods described before all start from some statistical or information theoretic measure of dependence, but do not explicitly formulate a model of the data in terms of probabilities. In this section, we briefly introduce two classes of methods that start explicitly from a probabilistic model of the data, using global measures of fit (joint likelihood) or Bayesian approaches to identify the network structure.

4.1 Gaussian Graphical Models

The simplest probabilistic model one may wish to consider is a multivariate normal distribution. The probability density for a multivariate normal vector $\mathbf{x} \in \mathbb{R}^G$ is given by

$$p(\mathbf{x}|\mathbf{m}, \Sigma) = \frac{1}{\sqrt{2\pi|\Sigma|}} \exp\left[-\frac{1}{2} (\mathbf{x} - \mathbf{m})^T \Sigma^{-1} (\mathbf{x} - \mathbf{m})\right] \quad (4)$$

where the mean vector \mathbf{m} and variance-covariance matrix Σ represent the parameters of the distribution. The off-diagonal entries of the symmetric matrix Σ give the covariance between different entries of the random vector \mathbf{x} , which is related to the correlation via multiplication by the marginal standard deviations.

An important result is that the inverse of the variance-covariance matrix, the *precision matrix* $C = \Sigma^{-1}$, contains the *partial correlations* between entries in the random vector \mathbf{x} . The partial correlation represents the residual correlation between two variables once the effect of all the other variables has been removed. As such, it provides a better measure of association than simple correlation, as it is less vulnerable to spurious associations.

This insight has been effectively used in the context of GRNs by a class of models known as *Gaussian Graphical Models* [30]. The idea is to treat gene expression measurements as a multivariate normal random vector (each entry of the vector representing the expression of one gene), and then estimate the precision matrix from multiple conditions using maximum likelihood estimation. Since this requires estimating a number of parameters which is proportional to the square of the number of genes, regularization techniques are needed; sparse regularization techniques such as L1 regularization (also known as *graphical lasso* [31]) have the added advantage of returning a more interpretable result, with the non-zero entries of the precision matrix representing the edges of the (undirected) regulatory network. Several algorithmic approaches have been proposed to carry out this estimation efficiently, and Gaussian Graphical Models represent a popular network inference approach. Within this book, Chapter 6 discusses the most recent developments in Gaussian Graphical Models usage.

While Gaussian Graphical Models are certainly a success story, as usual they come with limitations. Estimating a high-dimensional precision matrix from limited data is difficult, and, while using a consistent estimator such as penalized maximum likelihood brings guarantees in the infinite sample limit, the accuracy of the reconstruction for finite samples is more difficult to quantify a priori. More problematically, Gaussian Graphical Models assume normality of the data, which implies linearity in the relationship between the various genes. While this can be a reasonable approximation, and surprisingly effective inferentially, it certainly is a strong modelling limitation.

4.2 Bayesian Networks

All methods described so far address the problem of network reconstruction from a top-down approach: start with a fully connected network, compute pairwise scores (or estimate jointly a precision matrix in the case of Gaussian Graphical Models), and

then threshold/regularize to obtain a sparse network structure. In this subsection we will briefly introduce a very popular class of methods that takes the opposite approach, constructing a joint probabilistic model out of local conditional terms, *Bayesian networks*.

The starting point is the product rule of probability, which holds that for any two random variables X and Y , $P(X, Y) = P(X|Y)P(Y)$. Applying this rule recursively, one has that for G variables

$$P(X_1, \dots, X_G) = P(X_1) \prod_{i=2}^G P(X_i|X_1, \dots, X_{i-1}) \quad (5)$$

This factorization is general and clearly not unique, since the ordering of the random variables is arbitrary. Bayesian networks start from this general factorization, but create structure by imposing that only a subset of all possible variables are relevant in the conditioning set [32]. More formally, for each variable X_i , we define the set of *parents* of X_i , $\pi_i \subset \{X_1, \dots, X_{i-1}, X_{i+1}, \dots, X_G\}$. We then construct a directed network by connecting parents and children (the direction of the arrow goes from parents to children); the network structure corresponds to a special factorization of the joint probability as

$$P(X_1, \dots, X_G|\mathcal{G}) = \prod_{i=1}^G P(X_i|X_{\pi_i}) \quad (6)$$

where we introduce the variable \mathcal{G} to denote the graph structure of the Bayesian network. When the parent set π_i is empty, the conditional distribution $P(X_i|X_{\pi_i})$ is equal to the marginal distribution $P(X_i)$. See Fig. 3 for an example. Two remarks are important: not all parents–children assignments will lead to a valid factorization of the joint probability distribution. A fundamental result is that only networks without directed loops (*directed acyclic graphs, DAGs*) specify valid probability distributions (i.e., you cannot return to the same place walking on the network along the direction of the arrows). This global constraint poses considerable difficulties to reconstruction algorithms. Furthermore, even with the DAG constraint, the correspondence between networks and probability distributions is not one-to-one. As already highlighted in the case of the factorization (5), there can be multiple valid factorizations of a joint probability distribution, leading to different networks encoding exactly the same probability distribution. This issue is known as *Markov equivalence* in probability theory; see, e.g., [33] Ch. 3 for more details about the mathematical aspects of graphical statistics.

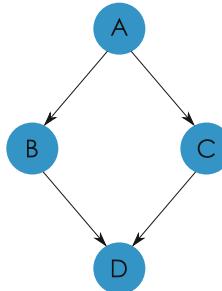


Fig. 3 Example of a valid Bayesian Network with four nodes and four edges. Given this structure \mathcal{G} , the joint distribution $P(A, B, C, D|\mathcal{G})$ factorizes as $P(A)P(B|A)P(C|A)P(D|B, C)$

Within a GRN context, Bayesian networks have been hugely popular due to the simplicity with which prior information (e.g., in the form of known interactions) can be incorporated (*see*, for example, Chapter 7 for applications of this paradigm to modern problems). As usual, gene expression levels are taken to represent the nodes of the network. For computational convenience, all conditional distributions are generally assumed to be Gaussian or discrete (multinomial), which enables the distributional parameters to be efficiently marginalized. In this way, one can easily compute the marginal likelihood function by evaluating the probability of the data under the model. The outstanding problem then remains the identification of the network structure. This is a very difficult combinatorial optimization problem. Greedily searching the space of networks structures for an optimum of the likelihood was an early solution [34]: although this can be surprisingly effective, in practice the cardinality of the space of network structures increases super-exponentially with the number of nodes, creating a formidable computational problem. This problem is compounded by the existence of multiple optima (due to Markov equivalence) and by the fact that the search must be constrained by the global DAG condition. As an alternative, Bayesian statistical methods have been extensively studied. This approach usually proceeds by constructing a biased random walk in the space of allowable network structures such that structures with a higher posterior probability are visited more often (a procedure called Markov Chain Monte Carlo) [35]. The Bayesian approach has considerable advantages in the ease with which prior information can be encoded, and in the way the intrinsic uncertainty in the system is represented: typically, such methods return an ensemble of plausible network structures, weighted by their posterior probability. Nevertheless, Bayesian methods suffer from considerable computational overheads and, despite recent advances [36], the scalability of Bayesian network methods to genome-wide data sets remains a challenge.

5 Dynamical Models

One of the central questions in biology is how organisms adapt to changing conditions. Therefore, a substantial fraction of high-throughput experiments have a time series design, e.g., they assay the same system at different time points to follow the evolution of the system in time. GRNs play a fundamental role in the mathematical modelling of such processes; unsurprisingly, several GRN reconstruction techniques are tailored towards the analysis of time series data. In this section, we introduce two broadly used classes of methods to infer network structures from dynamic data.

5.1 Dynamic Bayesian Networks

As we have seen in the previous chapter, a fundamental requirement on the structure of a Bayesian network is the absence of loops (DAG condition). Within the GRN context, this has long been seen as one of their main limitations: biological systems often exhibit feedback loops as a mechanism to engender robustness and stability. An elegant solution is provided by *Dynamic Bayesian Networks* (DBNs), a special class of Bayesian networks adapted for time series data.

DBNs work around the DAG condition by expanding the set of random variables under consideration, so that the nodes of the network now represent expression of genes *at a specific time point*. Network edges may now only connect nodes pertaining to different time points, so that a gene can only influence the expression of another gene (or, indeed, itself) at a later time point (*see* Fig. 4 for an example). In this way, the DAG condition is automatically satisfied, while at the same time biologically plausible features such as feedback mechanisms can be easily incorporated. In most cases, the dynamic structure of a DBN is chosen such that edges are only present between nodes at consecutive time points, with time-independent transition probabilities: this assumption of a homogeneous, first order Markov process is a plausible approximation in many cases, and, particularly when the conditional distributions are chosen to be Gaussian, it allows the modeller to leverage a rich literature on signal processing in autoregressive models.

DBNs are extremely popular in the GRN context, and are implemented in several software tools (*see* [37] for a recent review, and also Chapters 2 and 3 in the present volume). Structure learning within DBNs is easier than in standard Bayesian Networks, since the DAG condition is automatically satisfied, however, it still remains computationally demanding, particularly in a Bayesian setting. From the modelling point of view, most implementations assume a linear dynamic model, which is clearly a limitation. Extensions exist which include nonlinear mappings between time points [38, 39] or that relax the time-homogeneity assumption [40], however, these incur generally higher computational costs

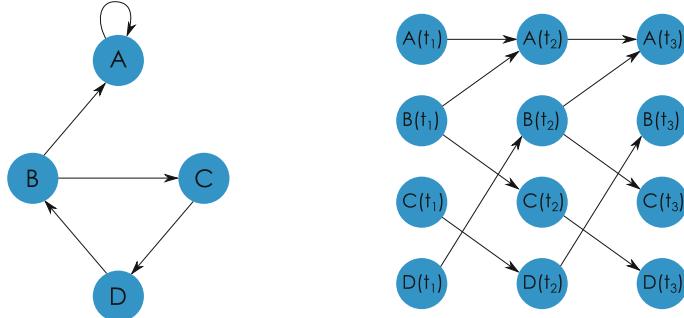


Fig. 4 Example of a Dynamic Bayesian Network with four nodes: static representation (with cycles) on the left, and unrolled dynamic representation on the right

and/or place strong restrictions on the class of nonlinear functions allowed. Most often, DBNs are implemented so that each time point in the model corresponds to an observation time. While this is somewhat natural, it constrains all biological processes to have essentially the same time-scale, which can be a serious limit; this is addressed by using a continuous-time semantic within the model, as in the case of continuous-time Bayesian Networks [41] or, more generally, of differential equation models.

5.2 Differential Equation Methods

Differential equations represent perhaps the best studied and most widely used class of dynamical models in science and engineering. They provide an infinitesimal description of the system dynamics by relating the rate of change (time derivative) of a variable to its value,

$$\frac{dx}{dt} = f(x, \Theta, u(t), t). \quad (7)$$

Here f is a general, time dependent, vector-valued function of the variable of interest x itself, taking as additional inputs a set of parameters Θ and possibly also a set of external signals $u(t)$. When the function f does not depend explicitly on time, the system is said to be *time homogeneous*, and when it does not depend on external inputs it is said to be *autonomous*.

Within a GRN context, the variables x are the expression levels of the set of genes we are interested in modelling, and the interactions between genes are encoded in the parameters Θ . By far the most widely used class of models are linear, autonomous and time homogeneous models, where Eq. (7) simplifies to

$$\frac{dx}{dt} = Ax \quad (8)$$

where the parameters Θ form the interaction matrix A . A non-zero entry A_{ij} signifies an influence of gene j on the time evolution of gene i , and hence a directed edge between j and i in the GRN.

Equation (8) or variants thereof are at the core of several methods for inferring GRNs. The Inferelator [42] is one such popular approach, where the derivative on the left-hand side of (8) is approximated with the difference of observed values at consecutive time points, and the network structure is recovered via L1 regularized regression. Other approaches solve directly the differential equation (8), positing the solution to be a linear combination of basis functions [43] or a draw from a Gaussian process [44], and then take a Bayesian approach to infer the parameters of the differential equation under a suitable, sparsity inducing prior distribution. Finally, the restriction to linear dynamics is not central to methods based on differential equations, and indeed methods using nonlinear dynamics (such as Hill kinetics [45]) have been proposed. See Chapter 16 for a comprehensive description of state-of-the-art methods for inferring GRNs using differential equations.

Differential equation models offer several potential advantages: their continuous-time semantics is closer to the class of models used in biophysical approaches to systems biology [46], so that in principle such approaches can benefit from a more mechanistic interpretation. Employing a continuous-time semantics also has the added advantage of limiting the influence of experimental design decisions (e.g., choice of time points/sampling frequencies) on the final result. In other respects, differential equation models are subject to the same computational hurdles as other methods, and they suffer from similar identifiability issues.

6 Multi-Network Models

All of our previous discussion has assumed that all the data can be explained by a single network structure. While this may be reasonable when all data comes from similar conditions, it is a very strong assumption when one is trying to jointly model data from heterogeneous scenarios, as different biological conditions may lead to different pathways being activated, so that effectively different network structures may be more appropriate.

This idea has been fruitfully exploited in two main directions. Several papers have considered the scenario where data (e.g., time series) is available from different, but related conditions. Therefore, one may reasonably assume some commonalities between the underlying network structures, so that methods that can transfer information across conditions are needed. This transfer can be achieved via introducing a shared diversity penalty within different optimization problems [47, 48]. Equivalently but more flexibly, the joint reconstruction of the different networks can be achieved by adopting a hierarchical Bayesian approach [49, 50].

Another direction that has seen considerable interest is the idea of *time-varying networks*. Here, the assumption is that the network structure itself can rewire across time, for example, to account for checkpoints during development or cancer evolution. The solution is generally composed of two steps: the identification of the change-points, and a joint learning of related networks across the homogeneous stretches of the time series. This idea has been explored both in the context of optimization approaches [51, 52] and more extensively in a Bayesian scenario [53–55].

Some of these ideas are explored in Chapters 2, 10, 11, and 13 of this volume.

7 Evaluation

During our discussion of various methods for GRN inference, we have often referred to several methods as successful or effective, without specifying how the performance of a particular method may be evaluated. This is a difficult issue: GRN inference methods are motivated precisely by the difficulty of directly measuring regulatory relationships between genes, therefore almost by definition gold standard scenarios where such interactions are known with high confidence are rare. One possibility is the recourse to simulated data. One may employ a biochemically plausible interaction model to generate some simulated gene expression measurements, and then evaluate the accuracy of the method against this gold standard. This strategy has been advocated by major international initiatives such as the Dialogue for Reverse Engineering of Models (DREAM) [56], which has organized a long-running challenge on GRN inference, providing both a stimulus and a benchmark for methodological development. Another direction has been the use of synthetic gene circuits as a benchmark for network reconstruction algorithms. The most well-known example of this is probably the IRMA network [57], a synthetic network of five genes engineered within living yeast cells. While this synthetic biology approach is appealingly close to biological reality, so far technological limitations mean that such an approach has been limited to small networks containing a handful of genes.

Having decided on a benchmark data generation procedure, the next step in evaluating a GRN inference algorithm is the choice of a suitable metric. Naively, one may consider thresholding the algorithm’s outputs and reporting an average accuracy in detecting the presence or absence of edges. This strategy is however flawed since GRNs are typically very sparse, so that an algorithm constantly predicting the absence of edges would potentially achieve high accuracy. A better strategy is to consider the fraction of true positive calls relative to all positives (*sensitivity* or *recall*) and the fraction of true positive calls out of all positive calls (*precision* or *positive predictive value*).

Naturally, precision and recall depend on the threshold chosen: with a very lax cutoff, we will likely recall many true positives (high recall), at the cost of many false positives (low precision). To elucidate the effectiveness of an algorithm in handling the precision/recall trade-off, a visually appealing strategy is the use of *precision-recall curves*. These are constructed as follows: given the output of a GRN inference algorithm as a weighted network, one starts by thresholding at a very strict (high) threshold, where precision is expected to be high and recall will be low. Decreasing the threshold, one will progressively lower precision by introducing some false positives, but also increase recall, until at zero threshold (fully connected network) recall is 1 and precision is the fraction of actual edges over possible edges (positives fraction). This procedure results in a curve in precision-recall space (see Fig. 5, right panel, for an illustration) indicative of the overall performance of the inference algorithm: a random predictor will always have an expected precision equal to the positives fraction, while an ideal algorithm will have precision 1 for any recall between 0 and 1. These observations justify the use of the *area under the curve* as a global metric of performance for an algorithm, a choice almost universally adopted in evaluating GRN inference methods.

Alternatively, a receiver operating characteristic (ROC) curve may be used to evaluate a weighted network against a gold standard. A ROC curve plots the recall versus the *false positive rate* (the fraction of false positive calls relative to all negatives) for different thresholds on the weights, again progressively lowering the threshold. Precision-recall curves are however more suited than ROC curves for problems where the number of negatives is much higher than the number of positives, which is typically the case of GRNs [58].

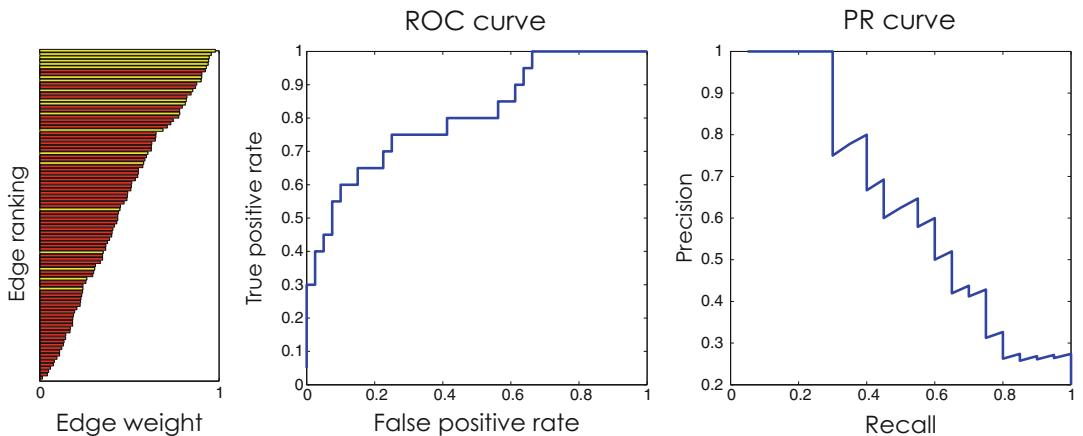


Fig. 5 Evaluation of inferred networks: an algorithm typically outputs a ranked list of edges, with the weight of each edge being given by either a score or a posterior probability (left panel, where true and false edges are colored in yellow and red, respectively). By progressively lowering the threshold for acceptance, one can construct either a ROC curve (central panel) or a precision-recall curve (right panel)

8 Software Tools

Most of the methods described above have been implemented in software tools which have been made freely available to the community. As it is perhaps to be expected of such a diverse and dynamic field, no single method has yet emerged as an industry standard, and tools differ widely in their usability and implementation. We provide here a summary list of some of the main software tools, as a reference list for the practitioner. All information is up-to-date at the time of writing (November 2017), and may clearly change. Naturally, this list is incomplete, and we would like to stress that any omissions do not reflect a judgement on the methods, but rather a restriction in space.

- WGCNA, weighted correlation network analysis, an R package available from the comprehensive R archive CRAN.

<https://labs.genetics.ucla.edu/horvath/CoexpressionNetwork/Rpackages/WGCNA/index.html>

- ARACNe, mutual information-based network inference approach. Source code in C++ available, as well as several OS-compatible versions and plugins.

<http://califano.c2b2.columbia.edu/aracne/>

- CLR, context likelihood of relatedness, mutual information-based network inference approach, originally implemented in MATLAB.

http://m3d.mssm.edu/network_inference.html

- MRNET, mutual information-based network inference approach. R implementation available in the Bioconductor package minet (also contains R implementations of ARACNe and CLR).

<https://www.bioconductor.org/packages/release/bioc/html/minet.html>

- GENIE3 and other tree-based methods, available as MATLAB, Python, and R packages.

<http://www.montefiore.ulg.ac.be/~huynh-thu/software.html>

- GeneNet, R package implementing Gaussian Graphical Models network inference, available from CRAN.

<https://cran.r-project.org/web/packages/GeneNet/index.html>

- CatNet, R package for (discrete) Bayesian Network structure learning, available from CRAN.

<https://cran.r-project.org/web/packages/catnet/index.html>

- Banjo, Java package for Bayesian Networks structure learning.
<https://users.cs.duke.edu/~amink/software/banjo/>
- G1DBN, R package for Dynamic Bayesian Network inference, available from CRAN.
<https://cran.r-project.org/web/packages/G1DBN/index.html>
- GRENTS, Bioconductor package for Dynamic Bayesian Network inference.
<https://bioconductor.org/packages/release/bioc/html/GRENITS.html>
- TSNI, differential equations-based method, available as MATLAB package.
<http://dibernardo.tigem.it/softwares/time-series-network-identification-tsni>
- Inferelator, differential equations-based method.
<http://bonneaulab.bio.nyu.edu/networks.html>
- netbenchmark, R package for benchmarking GRN inference methods (also contains R implementations of several methods such as ARACNe, C3NET, CLR, GeneNet, and GENIE3).
<https://www.bioconductor.org/packages/release/bioc/html/netbenchmark.html>

9 Discussion and Outlook

GRN inference is a mature field of methodological research, with widespread and increasing applications in biomedical research. In this chapter, we have attempted a broad brush introduction to the field, highlighting the biological motivation and the technological advances in data collection that have underpinned its recent flourishing. We then proceeded to give a bird’s eye view of the statistical principles underpinning some of the most popular methodologies for GRN inference. Our focus has been on the foundations, attempting a coarse categorization of different methods based on their assumptions and semantics. Of course, many interesting contributions fall at the intersection of different categories, and are not well accommodated by our simplifying approach.

Naturally, it is impossible to do justice to a rich and wide research area within a short introductory review. Our aim here is to prepare the reader for more advanced concepts to be described in subsequent chapters of this book; nevertheless, we hope that this chapter will also form a worthwhile introduction for the novice to the field, and have attempted to make it as self-contained as possible.

Acknowledgements

GS acknowledges support from the European Research Council under grant MLCS 306999. VAHT is a Post-doctoral Fellow of the F.R.S.-FNRS.

References

1. Crick F (1970) Central dogma of molecular biology. *Nature* 227(5258):561–563
2. Ptashne M, Gann A (2002) Genes & signals, vol 192. Cold Spring Harbor Laboratory Press, Cold Spring Harbor
3. Ptashne M (2004) A genetic switch: phage lambda revisited. Cold Spring Harbor Laboratory Press, Cold Spring Harbor
4. Ptashne M (2014) The chemistry of regulation of genes and other things. *J Biol Chem* 289(9):5417–5435
5. Bettenbrock K, Bai H, Ederer M, Green J, Hellingwerf KJ, Holcombe M, Kunz S, Rolfe MD, Sanguinetti G, Sawodny O, et al (2014) Towards a systems level understanding of the oxygen response of *Escherichia coli*. *Adv Microb Physiol* 64:65–114
6. Partridge JD, Sanguinetti G, Dibden DP, Roberts RE, Poole RK, Green J (2007) Transition of *Escherichia coli* from aerobic to micro-aerobic conditions involves fast and slow reacting regulatory components. *J Biol Chem* 282(15):11230–11237
7. Rolfe MD, Ter Beek A, Graham AI, Trotter EW, Asif HS, Sanguinetti G, de Mattos JT, Poole RK, Green J (2011) Transcript profiling and inference of *Escherichia coli* K-12 ArcA activity across the range of physiologically relevant oxygen concentrations. *J Biol Chem* 286(12):10147–10154
8. Alberts B, Bray D, Lewis J, Raff M, Roberts K, Watson JD (1994) Molecular biology of the cell, 3rd edn., vol 43(1294). Garland Pub, New York, p 67
9. Bird A (2002) DNA methylation patterns and epigenetic memory. *Genes Dev* 16(1):6–21
10. Karlić R, Chung HR, Lasserre J, Vlahoviček K, Vingron M (2010) Histone modification levels are predictive for gene expression. *Proc Natl Acad Sci* 107(7):2926–2931
11. Benveniste D, Sonntag HJ, Sanguinetti G, Sproul D (2014) Transcription factor binding predicts histone modifications in human cell lines. *Proc Natl Acad Sci* 111(37):13367–13372
12. Hogan DJ, Riordan DP, Gerber AP, Herschlag D, Brown PO (2008) Diverse RNA-binding proteins interact with functionally related sets of RNAs, suggesting an extensive regulatory system. *PLoS Biol* 6(10):e255
13. Tebaldi T, Re A, Viero G, Pegoretti I, Passerini A, Blanzieri E, Quattrone A (2012) Widespread uncoupling between transcriptome and translatome variations after a stimulus in mammalian cells. *BMC Genomics* 13(1):220
14. Bantscheff M, Lemeer S, Savitski MM, Kuster B (2012) Quantitative mass spectrometry in proteomics: critical review update from 2007 to the present. *Anal Bioanal Chem* 404(4):939–965
15. Brown PO, Botstein D (1999) Exploring the new world of the genome with DNA microarrays. *Nat Genet* 21:33–37
16. Wang Z, Gerstein M, Snyder M (2009) RNA-Seq: a revolutionary tool for transcriptomics. *Nat Rev Genet* 10(1):57–63
17. Evans C, Hardin J, Stoebel DM (2017) Selecting between-sample RNA-Seq normalization methods from the perspective of their assumptions. *Brief Bioinform*. <https://doi.org/10.1093/bib/bbx008>
18. Park PJ (2009) ChIP-seq: advantages and challenges of a maturing technology. *Nat Rev Genet* 10(10):669–680
19. West DB (2001) Introduction to graph theory, vol 2. Prentice Hall, Upper Saddle River
20. Zhang B, Horvath S (2005) A general framework for weighted gene co-expression network analysis. *Stat Appl Genet Mol Biol* 4(1)
21. Butte AJ, Kohane IS (1999) Mutual information relevance networks: functional genomic clustering using pairwise entropy measurements. In: Pacific symposium on biocomputing 2000. World Scientific, Singapore, pp 418–429
22. Margolin AA, Nemenman I, Basso K, Wiggin C, Stolovitzky G, Dalla Favera R, Califano A (2006) ARACNE: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. *BMC Bioinf* 7(1):S7

23. Faith JJ, Hayete B, Thaden JT, Mogno I, Wierzbowski J, Cottarel G, Kasif S, Collins JJ, Gardner TS (2007) Large-scale mapping and validation of *Escherichia coli* transcriptional regulation from a compendium of expression profiles. PLoS Biol 5(1):e8
24. Meyer PE, Kontos K, Lafitte F, Bontempi G (2007) Information-theoretic inference of large transcriptional regulatory networks. EURASIP J Bioinf Syst Biol 2007(1):79879
25. Haury AC, Mordelet F, Vert JP, Vera-Licona P (2012) TIGRESS: trustful inference of gene regulation using stability selection. BMC Syst Biol 6(1):145
26. Huynh-Thu VA, Irrthum A, Wehenkel L, Geurts P (2010) Inferring regulatory networks from expression data using tree-based methods. PLoS One 5(9):e12776
27. Huynh-Thu VA, Wehenkel L, Geurts P (2013) Gene regulatory network inference from systems genetics data using tree-based methods. In: Gene network inference: verification of methods for systems genetics data. Springer, Berlin, p 63
28. Huynh-Thu VA, Sanguinetti G (2015) Combining tree-based and dynamical systems for the inference of gene regulatory networks. Bioinformatics 31(10):1614–1622
29. Michailidis G, d'Alché Buc F (2013) Autoregressive models for gene regulatory network inference: sparsity, stability and causality issues. Math Biosci 246(2):326–334
30. Schäfer J, Strimmer K (2004) An empirical Bayes approach to inferring large-scale gene association networks. Bioinformatics 21(6):754–764
31. Friedman J, Hastie T, Tibshirani R (2008) Sparse inverse covariance estimation with the graphical lasso. Biostatistics 9(3):432–441
32. Pearl J (2014) Probabilistic reasoning in intelligent systems: networks of plausible inference. Morgan Kaufmann, San Francisco
33. Barber D (2012) Bayesian reasoning and machine learning. Cambridge University Press, Cambridge
34. Friedman N, Linial M, Nachman I, Pe'er D (2000) Using Bayesian networks to analyze expression data. J Comput Biol 7(3–4):601–620
35. Friedman N, Koller D (2003) Being Bayesian about network structure. A Bayesian approach to structure discovery in Bayesian networks. Mach Learn 50(1–2):95–125
36. Hill SM, Lu Y, Molina J, Heiser LM, Spellman PT, Speed TP, Gray JW, Mills GB, Mukherjee S (2012) Bayesian inference of signaling network topology in a cancer cell line. Bioinformatics 28(21):2804–2810
37. Oates CJ, Mukherjee S (2012) Network inference and biological dynamics. Ann Appl Stat 6(3):1209
38. Morrissey ER, Juárez MA, Denby KJ, Burroughs NJ (2010) On reverse engineering of gene interaction networks using time course data with repeated measurements. Bioinformatics 26(18):2305–2312
39. Äijö T, Lähdesmäki H (2009) Learning gene regulatory networks from gene expression measurements using non-parametric molecular kinetics. Bioinformatics 25(22):2937–2944
40. Grzegorczyk M, Husmeier D (2011) Non-homogeneous dynamic Bayesian networks for continuous data. Mach Learn 83(3):355–419
41. Nodelman U, Shelton CR, Koller D (2002) Continuous time Bayesian networks. In: Proceedings of the eighteenth conference on uncertainty in artificial intelligence. Morgan Kaufmann Publishers Inc., San Francisco, pp 378–387
42. Bonneau R, Reiss DJ, Shannon P, Facciotti M, Hood L, Baliga NS, Thorsson V (2006) The Inferelator: an algorithm for learning parsimonious regulatory networks from systems-biology data sets de novo. Genome Biol 7(5):R36
43. Trejo-Banos D, Millar AJ, Sanguinetti G (2015) A Bayesian approach for structure learning in oscillating regulatory networks. Bioinformatics 31(22):3617–3624
44. Dondelinger F, Husmeier D, Rogers S, Filippone M (2013) ODE parameter inference using adaptive gradient matching with Gaussian processes. In: Artificial intelligence and statistics, pp 216–228
45. McGoff KA, Guo X, Deckard A, Kelliher CM, Leman AR, Francey LJ, Hogenesch JB, Haase SB, Harer JL (2016) The local edge machine: inference of dynamic models of gene regulation. Genome Biol 17(1):214
46. Alon U (2006) An introduction to systems biology: design principles of biological circuits. CRC Press, Boca Raton
47. Niculescu-Mizil A, Caruana R (2007) Inductive transfer for Bayesian network structure learning. In: Artificial intelligence and statistics, pp 339–346

48. Chiquet J, Grandvalet Y, Ambroise C (2011) Inferring multiple graphical structures. *Stat Comput* 21(4):537–553
49. Werhli AV, Husmeier D, et al (2007) Reconstructing gene regulatory networks with Bayesian networks by combining expression data with multiple sources of prior knowledge. *Stat Appl Genet Mol Biol* 6(1):15
50. Penfold CA, Buchanan-Wollaston V, Denby KJ, Wild DL (2012) Nonparametric Bayesian inference for perturbed and orthologous gene regulatory networks. *Bioinformatics* 28(12):i233–i241
51. Ahmed A, Xing EP (2009) Recovering time-varying networks of dependencies in social and biological studies. *Proc Natl Acad Sci* 106(29):11878–11883
52. Robinson JW, Hartemink AJ (2010) Learning non-stationary dynamic Bayesian networks. *J Mach Learn Res* 11(Dec):3647–3680
53. Lebre S, Becq J, Devaux F, Stumpf MP, Lelandais G (2010) Statistical inference of the time-varying structure of gene-regulation networks. *BMC Syst Biol* 4(1):130
54. Thorne T, Stumpf MP (2012) Inference of temporally varying Bayesian networks. *Bioinformatics* 28(24):3298–3305
55. Dondelinger F, Lèbre S, Husmeier D (2013) Non-homogeneous dynamic Bayesian networks with Bayesian regularization for inferring gene regulatory networks with gradually time-varying structure. *Mach Learn* 90(2):191–230
56. Marbach D, Costello JC, Küffner R, Vega NM, Prill RJ, Camacho DM, Allison KR, Kellis M, Collins JJ, Stolovitzky G, et al (2012) Wisdom of crowds for robust gene network inference. *Nat Methods* 9(8):796–804
57. Cantone I, Marucci L, Iorio F, Ricci MA, Belcastro V, Bansal M, Santini S, Di Bernardo M, Di Bernardo D, Cosma MP (2009) A yeast synthetic network for in vivo assessment of reverse-engineering and modeling approaches. *Cell* 137(1):172–181
58. Davis J, Goadrich M (2006) The relationship between precision-recall and ROC curves. In: Proceedings of the 23rd international conference on machine learning. ACM, New York, pp 233–240



Chapter 2

Statistical Network Inference for Time-Varying Molecular Data with Dynamic Bayesian Networks

Frank Dondelinger and Sach Mukherjee

Abstract

In this chapter, we review the problem of network inference from time-course data, focusing on a class of graphical models known as dynamic Bayesian networks (DBNs). We discuss the relationship of DBNs to models based on ordinary differential equations, and consider extensions to nonlinear time dynamics. We provide an introduction to time-varying DBN models, which allow for changes to the network structure and parameters over time. We also discuss causal perspectives on network inference, including issues around model semantics that can arise due to missing variables. We present a case study of applying time-varying DBNs to gene expression measurements over the life cycle of *Drosophila melanogaster*. We finish with a discussion of future perspectives, including possible applications of time-varying network inference to single-cell gene expression data.

Key words Time-varying networks, Dynamic Bayesian networks, Changepoint models

1 Introduction

Networks are an important conceptual and practical tool in modern molecular, cell, and systems biology [1–4]. Molecules, such as genes or proteins, interact via a number of biochemical and biophysical mechanisms, thereby influencing each other's abundance. These mutual influences can be concisely depicted as a graph, with nodes (or vertices; we use both terms interchangeably) corresponding to measured or unmeasured molecular variables and edges to interplay between them. Depending on the precise formulation, the edges may indicate direct or indirect mechanistic influence of one molecule on another or simply some notion of statistical dependence.

The learning of molecular networks from high-throughput data is now a widely used approach in many areas of biomedical research [5–8]. The motivation is to better understand molecular interplay in settings of interest, for example, in a disease state or during a physiological process such as development.

Data acquired at multiple time points during a biological process may be informative and as the costs of data acquisition fall, such designs have become more common. We refer to such data generically as time-course data, regardless of whether the data are truly longitudinal or simply successive snapshots of biological samples acquired during the process of interest.

This chapter discusses statistical methods for learning molecular networks using time-course data. For additional perspectives on this problem, please also consult Chapter 3. After introducing notation and the problem set-up, we first discuss network inference based on a class of graphical models known as dynamic Bayesian networks (DBNs). We then show the relationship between DBNs and ordinary differential equation (ODE) models, and discuss how to extend DBNs to allow for nonlinear time dynamics. Next, we describe a flexible yet relatively tractable class of time-varying DBNs that allow for network structure and/or parameters to change over time. We also discuss causal perspectives on network inference, including issues around model semantics that can arise due to missing variables, and present a case study using time-varying networks to analyze gene expression data. We close with a discussion of future perspectives.

2 Notation and Problem Set-Up

We focus on learning biological networks from molecular time-course data. Let $V = \{1 \dots p\}$ index a set of variables that have been measured in molecular assays and $X_{j,t}$ be a random variable corresponding to the measurement of variable $j \in V$ at time point t . Although below we will also consider continuous-time models, the data are observed at a set of discrete time points, hence we write $t \in \{1 \dots T\}$ where T is the total number of time points. In the interests of simplicity of exposition we do not explicitly consider replicates at a given time point (i.e., multiple realizations of $X_{j,t}$) but note that dealing with replication is an important topic in its own right that can be handled via a number of approaches including simple averaging as a pre-processing step or more principled approaches such as mixed models.

We will use $X_t \in \mathbb{R}^p$ to denote the complete observed vector at time t (some of the models we discuss can also be used for observations $X_{j,t}$ that are discrete but we will mainly focus on continuous formulations) and $X = (X_1 \dots X_T)^T$ to denote the complete $T \times p$ data matrix. For a subset $A \subseteq V$ of variable indices, we use $X_{A,t}$ to denote the random vector formed by selecting the corresponding elements of X_t .

In learning networks the main estimand is a network or graph (we use both terms interchangeably) intended to encode relationships between the variables V . Let G denote a graph-valued model parameter and Θ any other parameters needed to specify the

model (e.g., regression coefficients, reaction rates, etc.). Then, we consider likelihoods of the form $p(X_1 \dots X_T | G, \Theta)$. We denote the edge set of a graph G as $E(G)$, or simply E when the graph of interest is clear from context. It is important to note that although one can specify statistical models with graph-valued parameters, this does not immediately imply that the semantics of the model align with a mechanistic or causal interpretation.

We will generically use \mathcal{G} to denote the space of permissible graphs. Additionally, we use $N(\cdot | \mu, \Sigma)$ to denote a multivariate normal density with mean μ and covariance matrix Σ , I to denote the identity matrix, and $\mathbb{I}(\cdot)$ to denote the indicator function.

3 Dynamic Bayesian Networks

3.1 Bayesian Networks

Bayesian networks (BNs) are graphical models [9] based on directed acyclic graphs (DAGs). In a BN the likelihood factors into terms in which each variable is conditioned only on its parents in the DAG G , i.e.,

$$p(X | G, \Theta) = \prod_{j=1}^p p(X_j | X_{\text{Pa}_G(j)}, \theta_j) \quad (1)$$

where $\text{Pa}_G(j) = \{i \in V : (i, j) \in E(G)\}$ is the set of parents of node j in the graph G and $\Theta = (\theta_1 \dots \theta_p)$ are parameters required to fully specify the conditional distributions.

3.2 Dynamic Bayesian Networks

In a dynamic Bayesian network (DBN) the model has an explicit discrete time index. The model can be formulated as a BN whose DAG has one vertex for each variable at each discrete time point, i.e., with $p \times T$ vertices in total [10–12]. We denote this graph by G_{full} . The simplest form of DBN assumes that the edges are only one step forward in time (i.e., the model is first-order Markov) and that neither the dependence pattern nor parameters change with time in the sense that any edge between variables $i \in V$ and $j \in V$ appears between all pairs of successive time points (or none) and the required model parameters are also unchanging. This type of DBN is often referred to as “feedforward” to reflect the graph structure and stationary to reflect the unchanging nature of the model.

The network structure of such a model does not change over time and the DAG G_{full} is a redundant representation since each edge is repeated between each pair of successive times. A more compact representation is as a bipartite directed graph $G_{\text{bipartite}}$ with only two time slices each with p nodes, with the two slices understood to represent successive times $(t-1, t)$. A still more compact representation is as a p -node directed graph G_{reduced} in which all edges are understood to go forwards in time between

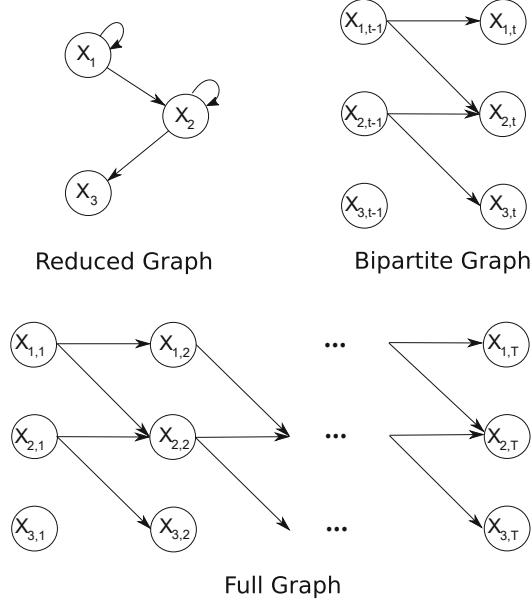


Fig. 1 Different graphical representations for a DBN. The top left shows the reduced graph G_{reduced} , where the temporal structure is implicit. The top right shows the bipartite graph $G_{\text{bipartite}}$ between any two time slices $t - 1$ and t . The bottom shows the full (or “unrolled”) graph G_{full} over all time slices

successive time points. Due to the fact that DAG G_{full} can be obtained by “unrolling” G_{reduced} through time, it is often referred to as the “unrolled” representation. These three representations are shown in Fig. 1 for an illustrative example.

Although G_{full} must be acyclic to obtain a valid overall BN model, the reduced graph G_{reduced} may have cycles. This is due to the fact that its edges are understood to go forward in time, hence the full graph G_{full} corresponding to an arbitrary directed G_{reduced} is always acyclic. For notational simplicity in the remainder of this section we use the reduced graph and denote it by G ($= G_{\text{reduced}}$).

Using the reduced graph G the likelihood for a stationary feedforward DBN can be written as

$$p(X | G, \Theta) = \prod_{j=1}^p p(X_{j,1} | \theta_j^{(0)}) \prod_{t=2}^T p(X_{j,t} | X_{\text{Pa}_G(j),t-1}, \theta_j), \quad (2)$$

where, as above, θ_j are parameters governing the conditional distributions, $\theta_j^{(0)}$ are parameters governing the (marginal) distribution at the first time point and $\Theta = (\theta_j^{(0)}, \theta_j)_{j=1 \dots p}$. Note that due to the stationary nature of the model, the parameters do not have a time index.

Thus, in a simple DBN each variable depends on a subset of variables at the previous time point, with the subset given by

the parent set of the variable in the graph G . A more general DBN formulation allows for edges within time slices, i.e., edges in the unrolled graph that go from a vertex (i, t) to (j, t) for $i, j \in V$. This type of DBN may be useful, for example, when the process dynamics and time sampling are such that influences that are contemporaneous on the timescale of the model cannot be ruled out.

3.3 Inference in DBNs

DBNs are Bayesian networks (BNs) since the unrolled graph G_{full} is always a DAG. Inference regarding the model structure G_{full} can be carried out using methods from the BN literature. Here we outline some approaches to inference regarding the model structure G_{full} .

For stationary feedforward DBNs the likelihood Eq.(2) is expressed in terms of the reduced graph G ($= G_{\text{reduced}}$; see above) and it is this graph that we treat as the estimand. A common way to proceed is via a Bayesian formulation. Since the main object of interest is G , we consider the posterior probability distribution over candidate graphs. This can be written as

$$P(G | X) = \frac{p(X | G) P(G)}{\sum_{G \in \mathcal{G}} p(X | G) P(G)} \quad (3)$$

where \mathcal{G} is the space of all directed graphs (with p vertices; not necessarily acyclic) and $P(G)$ is a suitably chosen prior on the graph space. The latter allows background information, e.g., from scientific knowledge of the system of interest, to be incorporated into inference [see 12, for a practical example]. Note that the graph space is not restricted to acyclic graphs due to the fact that it is the reduced graph that is the estimand (see above).

The posterior distribution is a rich object that can be used to provide inferences regarding any graph feature. For example, the posterior probability that a specific edge (i, j) is included in the model is

$$P((i, j) \in E(G) | X) = \sum_{G \in \mathcal{G}} \mathbb{I}[(i, j) \in E(G)] P(G | X). \quad (4)$$

The term $p(X | G)$ is known as the marginal likelihood since it is obtained by marginalizing over model parameters as

$$p(X | G) = \int p(X | G, \Theta) p(\Theta) d\Theta \quad (5)$$

$$= \prod_{t=2}^T \prod_{j=1}^p \int p(X_{j,t} | X_{\text{Pa}_G(j),t-1}, \theta_j) p(\theta_j) d\theta_j \quad (6)$$

where we have used (2) and $p(\Theta)$ is a prior on model parameters, assumed to factor as $p(\Theta) = \prod_j p(\theta_j)$.

Under conjugate formulations the quantity

$$\int p(X_{j,t} | X_{\text{Pa}_G(j),t-1}, \theta_j) p(\theta_j) d\theta_j$$

can be obtained in closed form. A popular choice is based on the normal linear model, i.e., choosing

$$X_{j,t} | X_{\text{Pa}_G(j),t-1}, \theta_j \sim N(X_{\text{Pa}_G(j),t-1}\beta_j, \sigma_j^2), \quad (7)$$

where $\theta_j = (\beta_j, \sigma_j^2)$, β_j is a p -vector of model coefficients and σ_j^2 is a noise variance.

Using a conjugate prior (here, normal-inverse-gamma) then allows the marginal likelihood to be obtained in closed form [see 12, and references therein for details]. In a similar way, other conjugate formulations (e.g., multinomial-Dirichlet for discrete data) may be used to conveniently obtain the marginal likelihood in other cases.

However, even with a closed-form marginal likelihood, characterizing the posterior over G is complicated by the size of the space \mathcal{G} , which in the absence of any further constraints has cardinality 2^{p^2} .

Markov chain Monte Carlo (MCMC) methods are widely used to sample from the posterior $P(G | X)$. These methods work by constructing a Markov chain (whose states are in this case graphs in the space \mathcal{G}), whose stationary distribution is the desired posterior. A common approach is to use a Metropolis–Hastings sampler with small changes to the graph (e.g., single edge changes) used to form the proposal distribution (this is the mechanism by which the sampler explores the graph space). Such samplers are asymptotically valid but can be very slow to converge for large graph spaces. Efficient samplers for (D)BNs remain an active area of research [13, 14].

In the special case of feedforward DBNs posterior inference is greatly simplified by the structure of the model. In particular, posterior edge probabilities can be computed via a variable selection approach in which each node is treated separately. An additional assumption of in-degree bounded by a small number then reduces the problem to polynomial in p (see [12] for full details). This means that fully Bayesian analysis of feedforward DBNs with conjugate priors is in fact feasible for very large problems, with p in the hundreds or thousands, and can be parallelized (over vertices).

3.4 Relationship to ODEs

DBNs are closely related to differential equation models. To see the connection, consider the stationary, feedforward DBN introduced above, using the normal linear variant as in Eq. (7). Suppose that intervals between all successive times are equal to Δt . Then, using a linear model for the Euler gradient approximation we get

$$\frac{X_{j,t} - X_{j,t-1}}{\Delta t} = X_{t-1}\tilde{\beta} + \varepsilon \quad (8)$$

where $\tilde{\beta}$ is a p -vector of coefficients and ε is a zero mean Gaussian noise term. Multiplying by Δt and collecting the terms corresponding to the earlier time point $t-1$ on the RHS, we see that this is simply a reparameterization of the linear DBN model in Eq. (7).

Replacing the linear model in Eq. (8) with a suitable nonlinear model gives a simple way to connect DBN type formulations to systems of coupled nonlinear ordinary differential equations (ODEs) as we discuss below. More generally, consideration of factors such as sampling intervals and proper accounting of stochasticity leads to more complicated formulations (e.g., involving stochastic differential equations). For a fuller discussion of these issues arising at the intersection of statistical network models and biological dynamics we refer the interested reader to [15].

3.5 Joint Estimation of Multiple Non-identical Networks

In some problems, the data span several related contexts, such as cell types or disease subtypes, and the task is to estimate networks that are specific to these contexts. Let $h \in \{1, \dots, H\}$ index these contexts and $X^{(1)}, \dots, X^{(H)}$ the corresponding datasets. Then the goal is to estimate graphs $G^{(1)}, \dots, G^{(H)}$ corresponding to these contexts. The simplest approach would be to apply a DBN estimator \hat{G}_{DBN} as described above to each dataset to give

$$(\hat{G}^{(1)}, \dots, \hat{G}^{(H)}) = \hat{G}_{\text{DBN}}(X^{(1)}), \dots, \hat{G}_{\text{DBN}}(X^{(H)}).$$

If there is a large amount of data per context k , this approach may work well. But in practice the amount of data per context may be limited relative to dimensionality or to the size of the model space. On the other hand, in most problems spanning multiple contexts, the contexts, while non-identical, are related. For example, if the data are from H cell types from a specific organism, although one might expect cell-type-specific differences in the graphs $G^{(h)}$, there will also be much in common due to shared biology.

This suggests the possibility of developing estimators that pool data across contexts. This is the approach taken in [16] for linear feedforward DBNs. They place a joint prior $P(G^{(1)}, \dots, G^{(H)} | G, \eta)$ on the graphs $G^{(h)}$ where G is a shared, latent graph and η additional hyperparameters. The idea is that the graphs $G^{(h)}$ are shrunk towards the latent graph G which is itself estimated from the data. This gives an overall estimator of the form

$$(\hat{G}^{(1)}, \dots, \hat{G}^{(H)}) = \hat{G}_{\text{joint}}(X^{(1)}, \dots, X^{(H)}; \eta).$$

This serves to pool information across the contexts. Such methods have been used in a multiple-context setting in cancer in [17]. Related approaches are described in [18, 19] and [20]. We note that joint priors can be viewed as penalties that encourage similarity between the context-specific graphs; in a related line of work, [21] discuss joint estimation of Gaussian graphical models (for static data) from a penalized point of view.

4 Nonlinear Models

All the models described so far are linear. The chemical processes underlying molecular time-course data are nonlinear and it is of interest to consider nonlinear extensions of the models described above. One way to do so is to consider a statistical model for the time derivative of the vector of observables, i.e.,

$$\dot{X}_t = \mathcal{g}(X_t; \Theta, G)$$

where \mathcal{g} is a suitable nonlinear model, Θ a vector of model parameters, and G a graph used to indicate some notion of sparsity or pattern of dependence (details will depend on the specific formulation).

Models of this general form can be used for inference regarding graph structure, parameters or both, using inferential approaches broadly analogous to the ideas outlined above for linear models. For example, to make inference over a set of graphs or structural models, model selection could be performed by integrating out the parameters Θ . This is the approach taken in [22], which considered a small number of candidate models of a biological process and scored these models within a Bayesian framework.

A more general approach is to consider a space of graphs \mathcal{G} , with each graph $G \in \mathcal{G}$ corresponding to a set of statements about the structural dependencies in the system. To explore this space would require some way of generating a specific instance (corresponding to G) of a class of nonlinear dynamical models. This is the approach taken in [23]. For a given candidate graph G they proceed by automatically generating a system of nonlinear ODEs (up to unknown parameters). The parameters are then integrated out to give a Bayesian marginal likelihood of the form $p(X | G)$ (where X is the data) which is then used to perform network inference over the graph space \mathcal{G} .

A general difficulty in scaling up dynamical models is that the likelihood is usually difficult to access (as it will typically involve solving a set of coupled ODEs). A commonly used simplification is to work at the level of Euler approximations as in Eq. (8). This is computationally convenient, but the approximation itself is often

poor, and this is exacerbated under the conditions of sparse time sampling typical of biological studies.

An alternative to working directly with highly parameterized ODE systems is to adopt a non-parametric approach, where $X_{j,t}$ is modeled via a nonlinear function $f(X_{\text{Pa}_G(j),t-1})$, with, for example, a Gaussian process (GP) prior on f . This is the approach taken in [24] and [20]. In the latter, the authors define a covariance matrix $K_\theta(X_{\text{Pa}_G(j),t}, X_{\text{Pa}_G(j),t'})$, which allows them to model the (mean-centered) observations as:

$$X_j \mid X'_{\text{Pa}_G(j)}, \theta \sim N(0, K_\theta + \sigma^2 I),$$

where θ represents the hyperparameters of the Gaussian process. Note that here $X_j = (X_{j,2}, \dots, X_{j,T})$ and $X'_{\text{Pa}_G(j)} = (X_{\text{Pa}_G(j),1}, \dots, X_{\text{Pa}_G(j),T-1})$. As a consequence, the model is jointly Gaussian when considering the whole of X_j , but the dynamics between time points are nonlinear. Penfold et al. [20] applied this approach as part of a hierarchical Bayesian model for reconstructing gene regulatory networks from multiple time series experiments (see also Subheading 3.5 above).

Another way to deal with deviations from strict linearity is to consider models that are linear, but not stationary, e.g., with changes to parameters and/or structure over time. We consider such extensions in the next section.

5 Time-Varying Networks

Molecular processes in living cells themselves change over time, due to changes in the environment, the development of the organism, as a consequence of the cell cycle, or due to a disease process.

To fix ideas, consider the case of development. The activity of specific genes and pathways changes during the course of development. For example, the fruit fly *Drosophila melanogaster* goes through several developmental stages, from embryo to larva to pupa to adult (we consider a detailed case study in this particular context in Subheading 7 below). Genes involved in wing muscle development would naturally fulfill different roles during the embryonal phase, when no wings are present, than they do in the adult fly, when the wings have fully developed. As a consequence of such changes, molecular networks can also vary over time, often in reaction to an environmental trigger, such as the type of growth substrate.

Thus, we are confronted with the problem of inferring molecular networks from a series of discrete measurements or observations in time, where the structure and/or parameters of the network might themselves be changing with time. Moreover, we may not

always know at which stage structural changes are likely to occur, as the underlying processes may be time-delayed, or dependent on unobserved external factors.

5.1 DBNs with Changepoint Processes

The standard assumption underlying DBNs as described above is that time series have been generated from a homogeneous Markov process. This assumption is restrictive, and can potentially lead to erroneous conclusions when the underlying cell dynamics are changing. While there have been various efforts to relax the homogeneity assumption for undirected graphical models [25, 26], time-varying DBNs are a more recent research topic with the literature including work from penalized likelihood and Bayesian perspectives [27–34].

Working in a Bayesian framework, the method proposed in [29, 30] assumes a fixed network structure and only allows the interaction parameters to vary with time. This assumption is arguably too rigid when looking at processes where changes in the overall regulatory network structure are expected, e.g., in morphogenesis or embryogenesis. The method proposed in [27, 28] requires a discretization of the data, which incurs an inevitable information loss. Lèbre [32] and Lèbre et al. [33] allow for continuous data and for network structures associated with different nodes to change with time in different ways. However, this flexibility comes at a statistical price that is relevant in practical settings with a non-large number of measurements. To gain statistical efficiency, Dondelinger et al. [35] used a sequential prior to regularize across time epochs (i.e., allowing for information sharing across time). Below we will describe the changepoint model from [33] and the extension in [35] in more detail, as exemplars of the class of time-varying DBNs.

5.1.1 Introducing a Changepoint Process

The changepoint model from [33] allows for temporal changes to the network structure by introducing a multiple changepoint

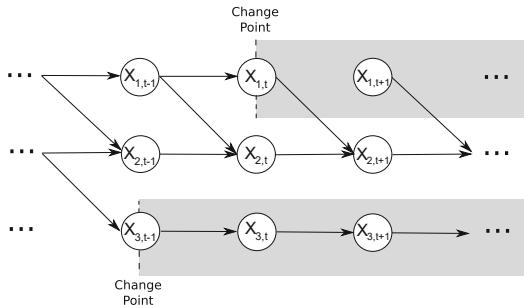


Fig. 2 Illustration of a multiple changepoint process in a DBN model. The model contains two changepoints, the first one affecting the parents of node 3 from time t , and the second one affecting the parents of node 1 from time $t + 1$, while the parents of node 2 remain unchanged over the three displayed time points

process, and performing Bayesian inference with reversible jump Markov chain Monte Carlo (RJMCMC) [36]. The model is illustrated in Fig. 2 for a network with three nodes, and two node-specific changepoints. We will refer to the model as TVDBN (Time-Varying Dynamic Bayesian Network), as in [35] ([33] refers to the same model as ARTIVA, which stands for Auto Regressive TIme-VArying model).

For each individual node in the network, we refer to the time points between two adjacent changepoints as an epoch. Within an epoch, the network structure (i.e., the parent set for that node) remains static, and hence under a first-order Markov assumption, the value that the node takes on at time t is determined by the values that the node's parents (i.e., potential regulators) take on at the previous time point, $t - 1$. More specifically, the conditional probability of the observation associated with a node at a given time point is a conditional Gaussian distribution, where the conditional mean is a linear weighted sum of the parent values at the previous time point, and the interaction parameters and parent sets depend on the time series epoch. The latter dependence adds extra flexibility to the model and thereby relaxes the homogeneity assumption.

Let p be the number of observed variables, and let $\mathbf{x} = (x_{i,t})_{1 \leq i \leq p, 1 \leq t \leq T}$ be the values measured at T time points. Following the notation in Subheading 3.2, G^b represents a directed acyclic graph of the p nodes. We define G_i^b as the subnetwork associated with target node i in epoch b , determined by the set of its parents, i.e., the nodes with a directed edge feeding into node i . Conditional on G_i^b , the random variable $X_{i,t}$ associated with node i depends on the p variables $\{X_{j,t-1}\}_{1 \leq j \leq p}$ via the linear regression:

$$X_{i,t} = \beta_{i0}^b + \sum_{j \in \text{Pa}_{G_i^b}(i)} \beta_{ij}^b X_{j,t-1} + \varepsilon_i^b(t) \quad (9)$$

with $\boldsymbol{\beta}_i^b = (\beta_{ij}^b)_{0 \leq j \leq p}$ a vector of regression coefficients, $\beta_{ij}^b \in \mathbb{R}$, and $\varepsilon_i^b(t) \sim N(0, (\sigma_i^b)^2)$; note that this is equivalent to the model in Eq. (7).

For each target node i , an unknown number k_i of changepoints define $k_i + 1$ non-overlapping epochs. Epoch $b = 1, \dots, k_i + 1$ starts at changepoint ξ_i^{b-1} and stops before ξ_i^b , where $\boldsymbol{\xi}_i = (\xi_i^0, \dots, \xi_i^{b-1}, \xi_i^b, \dots, \xi_i^{k_i+1})$ with $\xi_i^{b-1} < \xi_i^b$. To delimit the bounds, two pseudo-changepoints are introduced: $\xi_i^0 = 2$ and $\xi_i^{k_i+1} = T + 1$. Thus vector $\boldsymbol{\xi}_i$ has length $|\boldsymbol{\xi}_i| = k_i + 2$. The set of changepoints is denoted by $\boldsymbol{\xi} = (\boldsymbol{\xi}_i)_{1 \leq i \leq p}$. Each epoch b is associated with a network structure G_i^b . Let $G_i = \{G_i^b\}_{1 \leq b \leq k_i+1}$ and $G = \{G_i\}_{1 \leq i \leq p}$.

The interaction parameters, the variance parameters, the number of potential parents, the location of changepoints demarcating the time series segments, and the number of changepoints are given

(conjugate) prior distributions in a hierarchical Bayesian model. In particular, the number of changepoints k_i and the number of parents per node s_i^b are given truncated Poisson priors with mean λ and Λ , respectively. Conditional on s_i^b , the prior for the parent set G_i^b is a uniform distribution over all parent sets with cardinality s_i^b :

$$P(G_i^b | s_i^b) = 1/\binom{p}{s_i^b} = \frac{s_i^b!(p - s_i^b)!}{p!} \quad (10)$$

The overall prior on the network structures is given by marginalization:

$$P(G_i^b | \Lambda) = \sum_{s_i^b=0}^{\bar{s}} P(G_i^b | s_i^b) P(s_i^b | \Lambda) \quad (11)$$

The g-prior [37] is used for the regression coefficients β_i^b in the regression of node i in epoch b , such that:

$$P(\beta_i^b | G_i^b, \sigma_i^b) = 2\pi(\sigma_i^b)^2 |\Sigma_{G_i^b}|^{-\frac{1}{2}} \exp\left(-\frac{\{\beta_i^b\}^T \Sigma_{G_i^b}^{-1} \beta_i^b}{2(\sigma_i^b)^2}\right) \quad (12)$$

where $|.|$ denotes the determinant of a matrix, $\Sigma_{G_i^b}^{-1} = \delta^{-2} \mathbf{D}_{G_i^b}^T \mathbf{D}_{G_i^b}$ and $\mathbf{D}_{G_i^b}$ is the $(\xi_i^b - \xi_i^{b-1}) \times (s_i^b + 1)$ matrix whose first column is a vector of 1's (for the constant in model (9)) and each $(j+1)$ th column contains the observed values $(x_{j,t})_{\xi_i^{b-1}-1 \leq t < \xi_i^b}$ for each node j in G_i^b . The conjugate prior for the variance $(\sigma_i^b)^2$ is the inverse gamma distribution, $P((\sigma_i^b)^2) = IG(\nu_0, \gamma_0)$. For more details on the model priors, see [33] and [35].

From Bayes' theorem, the posterior is given by the following equation:

$$\begin{aligned} P(k, \xi, G, \beta, \sigma^2, \lambda, \Lambda, \delta^2 | \mathbf{x}) &\propto P(\delta^2) P(\lambda) P(\Lambda) \prod_{i=1}^p P(k_i | \lambda) P(\xi_i | k_i) \\ &\quad \prod_{b=1}^{k_i} P(G_i^b | \Lambda) P([\sigma_i^b]^2) P(\beta_i^b | G_i^b, [\sigma_i^b]^2, \delta^2) P(\mathbf{x}_i^b | G_i^b, \beta_i^b, [\sigma_i^b]^2) \end{aligned} \quad (13)$$

An attractive feature of the model is that the integration over the parameters β and σ^2 in the posterior distribution of Eq. (13) is analytically tractable. For details, see [33]. Consequently, the number of changepoints k and their location, ξ , the network structure G , and the hyperparameters λ , Λ , and δ^2 can be sampled from the posterior $P(k, \xi, G, \lambda, \Lambda, \delta^2 | \mathbf{x})$ with a reversible jump MCMC algorithm [36].

The changepoint model described above does not share any information across epochs; in particular, Eq. (11) only depends on Λ . This allows for drastic changes of the network structure at a changepoint. However, most biological processes show a smooth evolution, and we may want to include this prior knowledge in our model to allow our networks to change gradually over time. This is the purpose of sequential information sharing, which is discussed in the next subsection.

5.1.2 Sequential Information Sharing

Sequential information sharing over network structures makes sense when changes to the network take place gradually over the time period of the measurements. For instance, for the evolution of a gene regulatory network during embryogenesis, we would assume that the network evolves gradually and that networks associated with adjacent time intervals are *a priori* similar. We can model this assumption by introducing a prior for the network structure in each epoch, which depends on the previous epoch as illustrated in Fig. 3. Sequential information sharing provides a regularization that reduces the potential over-flexibility of the model, and thus reduces inference uncertainty.

Here we present two examples of information sharing priors from [35]. The first prior is based on using the exponential distribution proposed in [18]. The second prior uses a binomial distribution with a conjugate beta prior on the parameters. In [35], the authors additionally consider different schemes for sharing parameters of these priors across nodes in the network; for simplicity we will assume here that the parameters are the same for all nodes.

Exponential Prior

Denote by $K_i := k_i + 1$ the total number of partitions in the time series associated with node i , and recall that each time series segment \mathbf{D}_i^h is associated with a separate subnetwork G_i^h ,

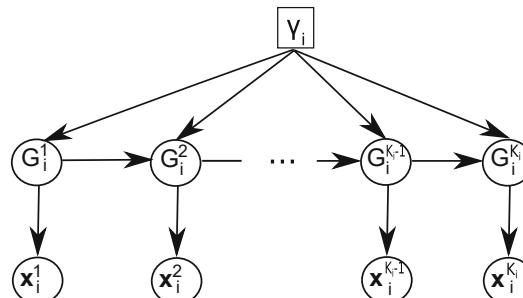


Fig. 3 Illustration of sequential information sharing. The network structure G_i^h for the subnetwork consisting of the parents of node i in epoch h is informed by the network structure in the previous epoch G_i^{h-1} , with the strength of the information coupling governed by a common hyperparameter γ_i

$1 \leq b \leq K_i$. We modify the prior from Eq.(11) by imposing a prior distribution $P(G_i^b | G_i^{b-1}, \gamma)$ on the structures, and the joint probability distribution factorizes according to a Markovian dependence:

$$\begin{aligned} P(\mathbf{D}_i^1, \dots, \mathbf{D}_i^{K_i}, G_i^1, \dots, G_i^{K_i}, \gamma) &= P(\mathbf{D}_i^1 | G_i^1) P(G_i^1) P(\gamma) \\ &\quad \prod_{b=2}^{K_i} P(\mathbf{D}_i^b | G_i^b) P(G_i^b | G_i^{b-1}, \gamma) \end{aligned} \quad (14)$$

Similar to [18] we define

$$P(G_i^b | G_i^{b-1}, \gamma) = \frac{\exp(-\gamma|G_i^b - G_i^{b-1}|)}{Z(\gamma, G_i^{b-1})} \quad (15)$$

for $b \geq 2$, where γ is a hyperparameter that defines the strength of the coupling between G_i^b and G_i^{b-1} , and $|.|$ denotes the Hamming distance. For $b = 1$, $P(G_i^b)$ is given by (11). The denominator $Z(\gamma, G_i^{b-1})$ in (15) is a normalizing constant, also known as the partition function: $Z(\gamma, G_i^{b-1}) = \sum_{G_i^b \in \mathbb{M}} e^{-\gamma|G_i^b - G_i^{b-1}|}$ where \mathbb{M} is the set of all valid subnetwork structures. If we ignore any fan-in restriction that might have been imposed a priori, then the expression for the partition function can be simplified: $Z(\gamma, G_i^{b-1}) \approx \prod_{j=1}^p Z_j(\gamma, e_{ij}^{b-1})$, where e_{ij}^b is a binary variable indicating the presence or absence of a directed edge from node j to node i in time series segment b , and $Z_j(\gamma, e_{ij}^{b-1}) = \sum_{e_{ij}^b=0}^1 e^{-\gamma|e_{ij}^b - e_{ij}^{b-1}|} = 1 + e^{-\gamma}$. Note that this expression no longer depends on G_i^{b-1} , and hence

$$Z(\gamma, G_i^{b-1}) = Z(\gamma) = (1 + e^{-\gamma})^p \quad (16)$$

Inserting this expression into (15) gives:

$$P(G_i^b | G_i^{b-1}, \gamma) = \frac{\exp(-\gamma|G_i^b - G_i^{b-1}|)}{(1 + e^{-\gamma})^p} \quad (17)$$

It is straightforward to integrate the proposed model into the RJMCMC scheme of [32] and [33], although we need to include an additional MCMC move for sampling the hyperparameter γ from the posterior distribution.

Binomial Prior

An alternative way of information sharing among segments and nodes is by using a binomial prior:

$$P(G_i^b | G_i^{b-1}, a, b) = a^{N_1^1[b,i]} (1-a)^{N_1^0[b,i]} b^{N_0^0[b,i]} (1-b)^{N_0^1[b,i]} \quad (18)$$

where we have defined the following sufficient statistics: $N_1^1[h, i]$ is the number of edges in G_i^{b-1} that are matched by an edge in G_i^b , $N_1^0[h, i]$ is the number of edges in G_i^{b-1} for which there is no edge in G_i^b , $N_0^1[h, i]$ is the number of edges in G_i^b for which there is no edge in G_i^{b-1} , and $N_0^0[h, i]$ is the number of coinciding non-edges in G_i^{b-1} and G_i^b . Since the hyperparameters are shared, the joint distribution can be expressed as:

$$\begin{aligned} P(\{G_i^b\}|\alpha, b) &= \prod_{i=1}^p P(G_i^1) \prod_{b=2}^{K_i} P(G_i^b|G_i^{b-1}, \alpha, b) \\ &= \alpha^{N_1^1} (1-\alpha)^{N_1^0} b^{N_0^0} (1-b)^{N_0^1} \prod_{i=1}^p P(G_i^1) \end{aligned} \quad (19)$$

where we have defined $N_k^l = \sum_{i=1}^p \sum_{b=2}^{K_i} N_k^l[h, i]$, and the right-hand side follows from Eq. (18). The conjugate prior for the hyperparameters α, b is a beta distribution,

$$P(\alpha, b|\alpha, \bar{\alpha}, \gamma, \bar{\gamma}) \propto \alpha^{(\alpha-1)} (1-\alpha)^{(\bar{\alpha}-1)} b^{(\gamma-1)} (1-b)^{(\bar{\gamma}-1)} \quad (20)$$

which using Bayes' rule leads to the (beta) posterior distribution:

$$\begin{aligned} P(\alpha, b|\alpha, \bar{\alpha}, \gamma, \bar{\gamma}, \{G_i^b\}) \\ \propto \alpha^{(\alpha+N_1^1-1)} (1-\alpha)^{(\bar{\alpha}+N_1^0-1)} b^{(\gamma+N_0^0-1)} (1-b)^{(\bar{\gamma}+N_0^1-1)} \end{aligned} \quad (21)$$

This allows the hyperparameters to be integrated out in closed form:

$$\begin{aligned} P(\{G_i^b\}|\alpha, \bar{\alpha}, \gamma, \bar{\gamma}) &= \int \int P(\{G_i^b\}|\alpha, b) P(\alpha, b|\alpha, \bar{\alpha}, \gamma, \bar{\gamma}) d\alpha db \\ &\propto \frac{\Gamma(\alpha + \bar{\alpha})}{\Gamma(\alpha)\Gamma(\bar{\alpha})} \frac{\Gamma(N_1^1 + \alpha)\Gamma(N_1^0 + \bar{\alpha})}{\Gamma(N_1^1 + \alpha + N_1^0 + \bar{\alpha})} \frac{\Gamma(\gamma + \bar{\gamma})}{\Gamma(\gamma)\Gamma(\bar{\gamma})} \frac{\Gamma(N_0^0 + \gamma)\Gamma(N_0^1 + \bar{\gamma})}{\Gamma(N_0^0 + \gamma + N_0^1 + \bar{\gamma})} \end{aligned} \quad (22)$$

The level-2 hyperparameters $\alpha, \bar{\alpha}, \gamma, \bar{\gamma}$ can be interpreted as fictitious prior observations due to the conjugacy of the prior. Again, the RJMCMC scheme is easily adapted, and we introduce a new move for sampling the level-2 hyperparameters.

6 Causal Extensions

Causal models allow conclusions to be drawn about changes under intervention that is about what would happen if an interventional experiment were carried out. Causal graphs are graphs that indicate

causal relationships. The semantics of graphs used in molecular biology are typically causal, in the sense that the edges are intended to describe well-defined biochemical processes whose interruption by experimental means would have outcomes in line with the edges indicated in the graph. In contrast, statistical graphical models are, in themselves, models of multivariate probability distributions that may or may not permit causal interpretation.

The distinction between the joint probability distribution over variables and a causal model can already be clearly seen in a two variable example. Consider measurements of two variables A and B . Suppose one has asymptotically many samples and that the joint distribution $p_{AB}(A, B)$ is essentially perfectly characterized. Even at this point, without further assumptions, and despite complete knowledge of the joint p_{AB} , it is impossible to know whether A causes B , or B causes A , or neither. To see why, suppose a third, unobserved variable C has a causal, mechanistic effect on both A and B . This induces a correlation or stochastic dependence between A and B (that would appear in the joint p_{AB}) despite the absence of any sequence of mechanistic events linking A and B themselves. Here, the issue is the confounding due to the missing variable C . These kinds of issues pose serious problems for causal modeling of systems such as biochemical networks and it is important to note that these issues are not in general automatically solved by the use of graph structured statistical models such as those described in this chapter, nor by large sample sizes. These issues have been extensively discussed in the causal modeling literature and we refer the interested reader to [38] for in-depth discussion.

The toy example in the preceding paragraph does not include time. It is tempting to assume that the inclusion of time, as in the DBNs and related models described in this chapter, addresses the causal issue due to the temporal ordering of causal influences (i.e., the fact that causes precede effects). However, issues such as in the toy three variable example can also arise in temporal data. Consider again the causal structure above, namely that variable C has a causal, mechanistic effect on two variables A and B which are otherwise not causally related. The effect of C on A might be more rapid than on B , which would appear as a stochastic dependence between A and B with a time lag. Given enough data, this might appear as an edge in a DBN with high confidence. However, as above, in this particular case it would be erroneous to conclude that A has a causal influence on B .

DBNs can be extended in a causal direction using notions from causal BNs as described in [38, 39]. An example of this line of work appears in [40] who model interventional data using a causal interpretation of DBNs. This approach is employed in a biological use case in [17]. Nevertheless, it remains unclear whether general and effective causal inference is possible in the context of time-varying biomolecular data and a recent consortium study focused on this

empirical question [41]. Further work—in theory, methodology, applications, and empirical validation—is needed to understand the scope and limitations of causal approaches for time-varying molecular data and this remains an active area of investigation.

7 A Case Study: Application to Gene Expression Time Series

To illustrate some of the ideas described in this chapter, we present a case study in the context of the life cycle of the fruit fly *Drosophila melanogaster*, originally presented in [35]. Here, the model is a time-varying DBN (TVDBN; see above) with sequential information sharing, and the data are gene expression time series data covering the life cycle of the fruit fly. During its life cycle, *Drosophila melanogaster* undergoes four major stages of morphogenesis: embryo, larva, pupa, and adult. Arbeitman et al. [42] obtained a gene expression time series covering all four stages. We applied TVDBNs to a subset of this gene expression time series consisting of 11 genes involved in wing muscle development. First, we investigated whether the changepoints inferred correspond to the known transitions between stages. Figure 4a shows the posterior probabilities of inferred changepoints for any gene using TVDBN-0 (unregularized by information sharing), while Fig. 4c, d show the posterior probabilities when including sequential information sharing priors. We additionally applied the ℓ_1 -regularized method proposed in [31], using the authors' software package TESLA (Fig. 4b). Another application using the same dataset can be found in [27], who applied a discrete non-homogeneous DBN, and a plot corresponding to Fig. 4b is included in their paper.

The non-homogeneous DBN methods recover changepoints for all three transitions (embryo \rightarrow larva, larva \rightarrow pupa, and pupa \rightarrow adult). As shown in Fig. 4b, the last transition, pupa \rightarrow adult, is less clearly detected with TESLA, and it is completely absent in [27]. TESLA and TVDBN both detect additional changepoints during the embryo stage, though these are not detected in [27]. It is not implausible that additional transitions at the gene regulatory network level should occur within one morphogenic phase. One would expect that a complex gene regulatory network is unlikely to transition into a new phase all at once, and some pathways might have to undergo activational changes earlier in preparation for the morphogenic transition.

In addition to the changepoints, we can infer network structures for the morphogenic stages of embryo, larva, pupa, and adult (see Fig. 5). Here we use the binomial information sharing prior in Eq. (18); see [35] for a simulation study motivating this choice. An objective assessment of the reconstruction accuracy is not feasible due to the limited existing biological knowledge and the absence of a gold standard. However, the reconstructed networks show

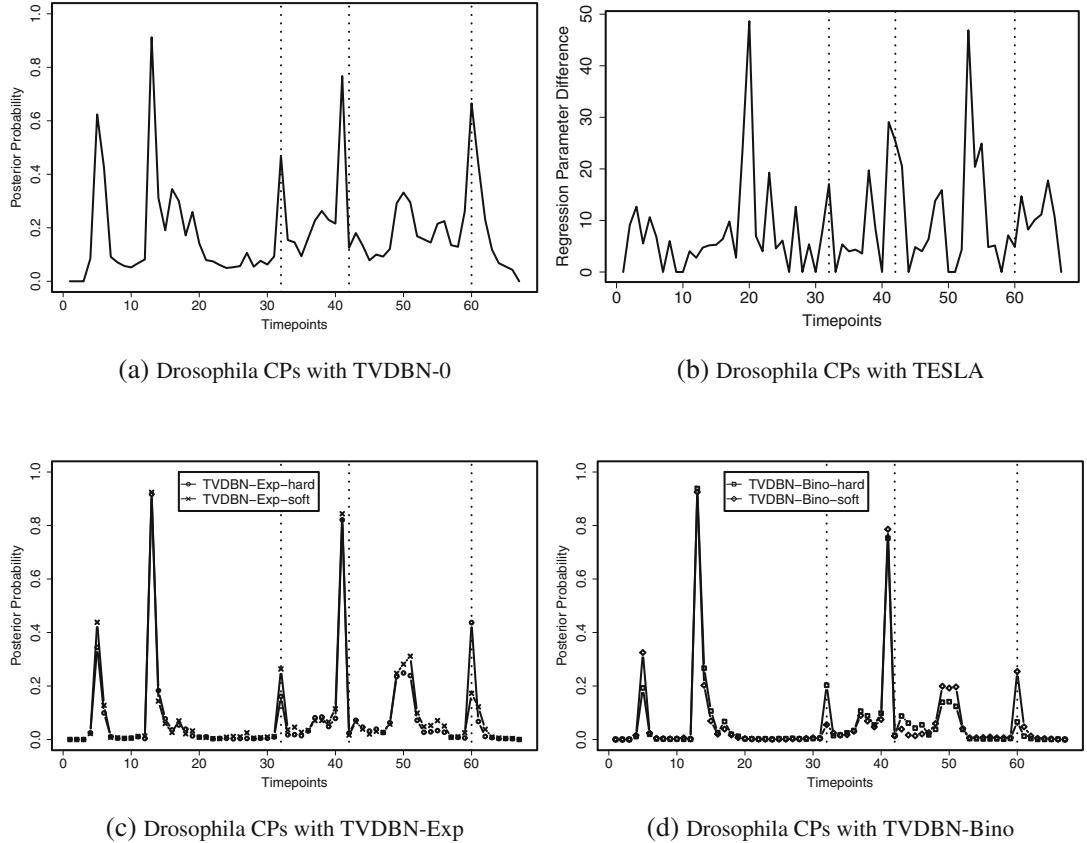


Fig. 4 Changepoints inferred from gene expression time series related to morphogenesis in *Drosophila melanogaster*. **(a)** TVDBN-0 changepoints (no information sharing). **(b)** TESLA, ℓ_1 -norm of the difference of the regression parameter vectors associated with two adjacent time points plotted against time. **(c)** TVDBN-Exp changepoints (exponential prior). **(d)** TVDBN-Bino changepoints (binomial prior). Hard and soft refer to whether the hyperparameters are shared across nodes (hard) or depend on a common level-2 hyperprior (soft)

many similarities with the networks discovered by Robinson and Hartemink [27], Guo et al. [43] and Zhao et al. [44]. For instance, we recover the interplay between two genes, *eve* and *twi*. This interplay is also reported in [43] and [44], while [27] seem to have missed it. We also recover a cluster consisting of the genes *myo6lf*, *msp300*, *mhc*, *prm*, *mlc1*, and *up* during all morphogenic phases. This result is not implausible, as all genes (except *up*) belong to the myosin family. However, unlike [27], we find that *actn* also participates as a potential regulator in this cluster. There is some indication of this in [44], where *actn* is found to regulate *prm*.

By validating the reconstructed networks using genetic and protein associations recorded in the FLIGHT database [45], we found that a number of the inferred links over all segments correspond to associations that have been reported in the literature. Some of these result from indirect paths, where the intermediate

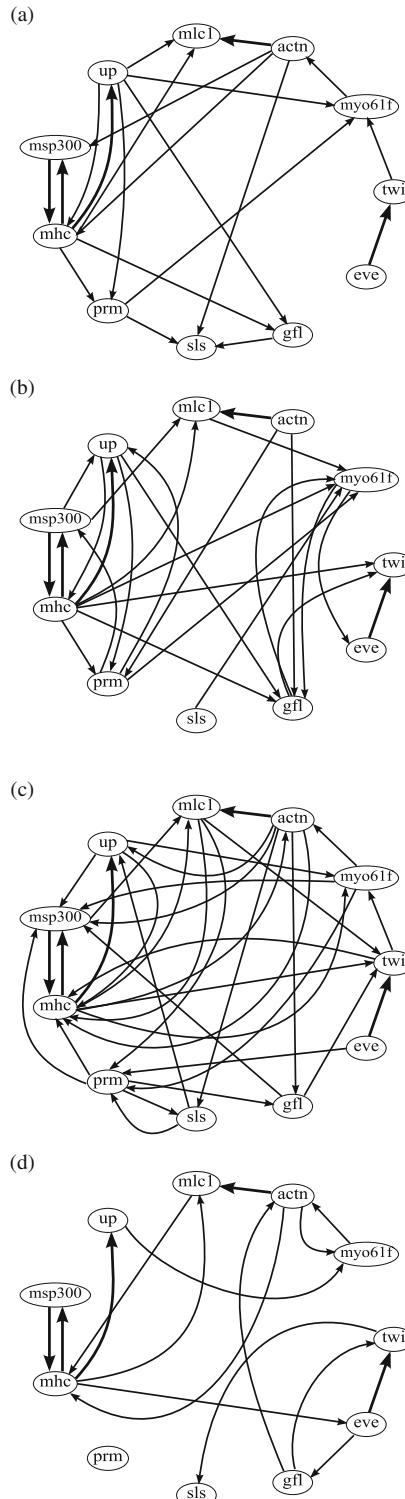


Fig. 5 Gene regulatory networks inferred from gene expression time series related to morphogenesis in *Drosophila melanogaster*, using TVDBN with

Table 1
Reconstructed gene interplay in the *Drosophila melanogaster* wing muscle development network, validated using the FLIGHT database [45]

Interaction	References	Interaction	Notes
<i>actn</i> ↔ <i>mhc</i>	[46–48]	Protein	Via missing gene <i>wupA</i>
<i>actn</i> → <i>up</i>	[46, 47]	Protein	Via missing gene <i>wupA</i>
<i>eve</i> → <i>twi</i>	[49]	Protein	Via missing gene <i>RpIII40</i>
<i>up</i> ↔ <i>mhc</i>	[46–48]	Protein	Direct interaction
<i>actn</i> → <i>msp300</i>	[50]	Gene	Via missing gene <i>TSG101</i> or missing gene <i>Hrs</i>
<i>actn</i> → <i>sls</i>	[51]	Gene	Direct interaction
<i>actn</i> → <i>prm</i>	[50]	Gene	Via missing gene <i>exo70</i>
<i>prm</i> ↔ <i>sls</i>	[50, 51]	Gene	Via missing gene <i>exo70</i> and present gene <i>actn</i>
<i>sls</i> → <i>up</i>	[51]; [50]	Protein and Gene	Via missing gene <i>Act88F</i>

gene is missing in the data. Table 1 gives an overview of the identified links with references to the biological literature.

The software used in this case study has been made available as the R package EDISON [52] on the Comprehensive R Archive Network (CRAN). Dynamic network inference in medium-sized datasets such as the *Drosophila* dataset used here takes under an hour on a modern desktop computer using the implementation of reversible-jump MCMC in EDISON. For larger networks, we recommend implementing more efficient approximate inference methods where possible, and initializing the changepoints and initial networks with a greedy optimization.

8 Conclusions and Future Outlook

Research in the last decade or so has provided a range of methodologies that can be used to efficiently infer biological networks from time-course data. Linear feedforward DBNs are the simplest

◀
Fig. 5 (continued) binomial information sharing prior. The networks were obtained by applying a threshold of 0.25 to the marginal posterior probabilities of the gene interactions. We have reconstructed a network for each morphological phase; interactions that were consistent across all four phases are marked in bold. **(a)** Embryo. **(b)** Larva. **(c)** Pupa. **(d)** Adult

and most scalable of these methods. The restrictive assumptions of these DBNs have been relaxed in various ways, including time-varying and nonlinear variants.

Scalable nonlinear modeling remains challenging. Although network inference has already been carried out using nonlinear formulations [as in 16, 22], these and related methods do not currently scale to large problems. We expect that scalable nonlinear modeling will become increasingly feasible via a combination of high-performance computing, suitable structural priors/constraints, and approximate inference techniques. This will soften the distinction between network inference on the one hand and dynamical systems biology modeling on the other, by allowing questions regarding structure and dynamics to be addressed in a unified way.

A new technology that has progressed tremendously in recent years is single-cell transcriptomics. Single-cell assays are already having a major effect in biological investigation, and it is reasonable to ask if there are opportunities for time-varying network inference. Measuring the amount of RNA expressed in individual cells allows for a new level of resolution relative to previous bulk assays that allowed only aggregate measurements over cell populations. Various methods have been proposed to leverage the extra information about inter-cell variability for the purpose of static network inference, for example, using mutual information [53], Bayesian non-parametric dependence testing [54], or Boolean networks [55]. While truly longitudinal single-cell measurements remain challenging, any single biological sample will contain cells at different stages in the cell cycle, thus introducing a temporal aspect. The so-called pseudo-time methods have been developed to disentangle this information and order the cells by their stage in the cell cycle [56–58]. It is then possible to apply dynamic network inference methods that make use of the pseudo-time ordering. A few recent examples include [59], which uses static network inference augmented by a model selection procedure that is informed by the ordering; [60] which uses a system of linear ODEs; and [61] which uses dynamic Bayesian networks. Judging by these developments, single-cell transcriptomics will likely prove to be a fruitful area for the application and development of time-varying network inference methods in the near future.

References

1. Kitano H (2002) Systems biology: a brief overview. *Science* 295(5560):1662–1664
2. Hopkins AL (2008) Network pharmacology: the next paradigm in drug discovery. *Nat Chem Biol* 4(11):682–690
3. Califano A, Butte AJ, Friend S, Ideker T, Schadt E (2012) Leveraging models of cell regulation and GWAS data in integrative network-based association studies. *Nat Genet* 44(8):841–847
4. Ideker T, Krogan NJ (2012) Differential network biology. *Mol Syst Biol* 8:565

5. Bansal M, Belcastro V, Ambesi-Impiombato A, di Bernardo D (2007) How to infer gene networks from expression profiles. *Mol Syst Biol* 3:78
6. Akbani R, Ng PKS, Werner HMJ, Shahmoradgoli M, Zhang F, Ju Z, Liu W, Yang JY, Yoshihara K, Li J, Ling S, Seviour EG, Ram PT, Minna JD, Diao L, Tong P, Heymach JV, Hill SM, Dondelinger F, Städler N, Byers LA, Meric-Bernstam F, Weinstein JN, Broom BM, Verhaak RGW, Liang H, Mukherjee S, Lu Y, Mills GB (2014) A pan-cancer proteomic perspective on the cancer genome atlas. *Nat Commun* 5:3887
7. Le Novère N (2015) Quantitative and logic modelling of molecular and gene networks. *Nat Rev Genet* 16(3):146–158
8. Yugi K, Kubota H, Hatano A, Kuroda S (2016) Trans-Omics: how to reconstruct biochemical networks across multiple ‘omic’ layers. *Trends Biotechnol* 34(4):276–290
9. Koller D, Friedman N (2009) Probabilistic graphical models: principles and techniques. MIT Press, Cambridge
10. Murphy KP (2002) Dynamic Bayesian networks: representation, inference and learning. Dissertation, University of California, Berkeley
11. Husmeier D (2003) Sensitivity and specificity of inferring genetic regulatory interactions from microarray experiments with dynamic Bayesian networks. *Bioinformatics* 19(17):2271–2282
12. Hill SM, Lu Y, Molina J, Heiser LM, Spellman PT, Speed TP, Gray JW, Mills GB, Mukherjee S (2012) Bayesian inference of signaling network topology in a cancer cell line. *Bioinformatics* 28(21):2804–2810
13. Grzegorczyk M, Husmeier D (2008) Improving the structure MCMC sampler for Bayesian networks by introducing a new edge reversal move. *Mach Learn* 71(2):265–305
14. Goudie RJ, Mukherjee S (2016) A Gibbs sampler for learning DAGs. *J Mach Learn Res* 17(1):1032–1070
15. Oates CJ, Mukherjee S (2012) Network inference and biological dynamics. *Ann Appl Stat* 6(3):1209
16. Oates CJ, Korkola J, Gray JW, Mukherjee S, et al (2014) Joint estimation of multiple related biological networks. *Ann Appl Stat* 8(3):1892–1919
17. Hill SM, Nesser NK, Johnson-Camacho K, Jeffress M, Johnson A, Boniface C, Spencer SE, Lu Y, Heiser LM, Lawrence Y, et al (2017) Context specificity in causal signaling networks revealed by phosphoprotein profiling. *Cell Syst* 4(1):73–83
18. Werhli AV, Husmeier D (2008) Gene regulatory network reconstruction by Bayesian integration of prior knowledge and/or different experimental conditions. *J Bioinformatics Comput Biol* 6(3):543–572
19. Dondelinger F, Husmeier D, Lèbre S (2012) Dynamic Bayesian networks in molecular plant science: inferring gene regulatory networks from multiple gene expression time series. *Euphytica* 183(3):361–377
20. Penfold C, Buchanan-Wollaston V, Denby K, Wild D (2012) Nonparametric Bayesian inference for perturbed and orthologous gene regulatory networks. *Bioinformatics* 28(12):i233–i241
21. Danaher P, Wang P, Witten DM (2014) The joint graphical lasso for inverse covariance estimation across multiple classes. *J R Stat Soc Ser B (Stat Methodol)* 76(2):373–397
22. Xu TR, Vyshemirsky V, Gormand A, von Kriegsheim A, Girolami M, Baillie GS, Ketley D, Dunlop AJ, Milligan G, Houslay MD, et al (2010) Inferring signaling pathway topologies from multiple perturbation measurements of specific biochemical species. *Sci Signal* 3(113):ra20
23. Oates CJ, Dondelinger F, Bayani N, Korkola J, Gray JW, Mukherjee S (2014) Causal network inference using biochemical kinetics. *Bioinformatics* 30(17):i468–i474
24. Äijö T, Lähdesmäki H (2009) Learning gene regulatory networks from gene expression measurements using non-parametric molecular kinetics. *Bioinformatics* 25(22):2937–2944
25. Talih M, Hengartner N (2005) Structural learning with time-varying components: tracking the cross-section of financial time series. *J R Stat Soc B* 67(3):321–341
26. Xuan X, Murphy K (2007) Modeling changing dependency structure in multivariate time series. In: Ghahramani Z (ed) Proceedings of the 24th annual international conference on machine learning (ICML 2007). Omnipress, Madison, pp 1055–1062
27. Robinson JW, Hartemink AJ (2009) Non-stationary dynamic Bayesian networks. In: Koller D, Schuurmans D, Bengio Y, Bottou L (eds) Advances in neural information processing systems (NIPS), vol 21, pp 1369–1376
28. Robinson J, Hartemink A (2010) Learning non-stationary dynamic Bayesian networks. *J Mach Learn Res* 11:3647–3680

29. Grzegorczyk M, Husmeier D (2009) Non-stationary continuous dynamic Bayesian networks. In: Bengio Y, Schuurmans D, Lafferty J, Williams CKI, Culotta A (eds) Advances in neural information processing systems (NIPS), vol 22, pp 682–690
30. Grzegorczyk M, Husmeier D (2011) Non-homogeneous dynamic Bayesian networks for continuous data. *Mach Learn* 83:355–419
31. Ahmed A, Xing EP (2009) Recovering time-varying networks of dependencies in social and biological studies. *Proc Natl Acad Sci* 106:11878–11883
32. Lèbre S (2007) Stochastic process analysis for genomics and dynamic Bayesian networks inference. PhD thesis, Université d'Evry-Val-d'Essonne, Évry
33. Lèbre S, Becq J, Devaux F, Lelandais G, Stumpf M (2010) Statistical inference of the time-varying structure of gene-regulation networks. *BMC Syst Biol* 4:130
34. Kolar M, Song L, Xing E (2009) Sparsistent learning of varying-coefficient models with structural changes. In: Bengio Y, Schuurmans D, Lafferty J, Williams CKI, Culotta A (eds) Advances in neural information processing systems (NIPS), vol 22, pp 1006–1014
35. Dondelinger F, Lèbre S, Husmeier D (2013) Non-homogeneous dynamic Bayesian networks with Bayesian regularization for inferring gene regulatory networks with gradually time-varying structure. *Mach Learn* 90:191–230
36. Green P (1995) Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika* 82:711–732
37. Zellner A (1986) On assessing prior distributions and Bayesian regression analysis with g-prior distributions. In: Goel P, Zellner A (eds) Bayesian inference and decision techniques. Elsevier, New York, pp 233–243
38. Pearl J (2009) Causality. Cambridge University Press, Cambridge
39. Pearl J (1995) Causal diagrams for empirical research. *Biometrika* 82(4):669–688
40. Spencer SE, Hill SM, Mukherjee S (2015) Inferring network structure from interventional time-course experiments. *Ann Appl Stat* 9(1):507–524
41. Hill SM, Heiser LM, Cokelaer T, Unger M, Nesser NK, Carlin DE, Zhang Y, Sokolov A, Paull EO, Wong CK, et al (2016) Inferring causal molecular networks: empirical assessment through a community-based effort. *Nat Methods* 13(4):310–318
42. Arbeitman M, Furlong E, Imam F, Johnson E, Null B, Baker B, Krasnow M, Scott M, Davis R, White K (2002) Gene expression during the life cycle of *Drosophila melanogaster*. *Science* 297(5590):2270–2275
43. Guo F, Hanneke S, Fu W, Xing E (2007) Recovering temporally rewiring networks: a model-based approach. In: Proceedings of the 24th international conference on machine learning. ACM, New York, p 328
44. Zhao W, Serpedin E, Dougherty E (2006) Inferring gene regulatory networks from time series data using the minimum description length principle. *Bioinformatics* 22(17):2129
45. Sims D, Bursteinas B, Gao Q, Zvelebil M, Baum B (2006) FLIGHT: database and tools for the integration and cross-correlation of large-scale RNAi phenotypic datasets. *Nucleic Acids Res* 34:D479–D483
46. Homyk T Jr, Emerson C Jr (1988) Functional interactions between unlinked muscle genes within haploinsufficient regions of the *Drosophila* genome. *Genetics* 119(1):105
47. Nongthomba U, Cummins M, Clark S, Vigoreaux J, Sparrow J (2003) Suppression of muscle hypercontraction by mutations in the myosin heavy chain gene of *Drosophila melanogaster*. *Genetics* 164(1):209
48. Montana E, Littleton J (2004) Characterization of a hypercontraction-induced myopathy in *Drosophila* caused by mutations in mhc. *J Cell Biol* 164(7):1045
49. Parkhurst S, Ish-Horowicz D (1991) wimp, a dominant maternal-effect mutation, reduces transcription of a specific subset of segmentation genes in *Drosophila*. *Genes Dev* 5(3):341
50. Formstecher E, Aresta S, Collura V, Hamburger A, Meil A, Trehan A, Reverdy C, Betin V, Maire S, Brun C, et al (2005) Protein interaction mapping: a *Drosophila* case study. *Genome Res* 15(3):376
51. Sanchez C, Lachaize C, Janody F, Bellon B, Roeder L, Euzenat J, Rechenmann F, Jacq B (1999) Grasping at molecular interactions and genetic networks in *Drosophila melanogaster* using FlyNets, an internet database. *Nucleic Acids Res* 27(1):89
52. Dondelinger F, Lèbre S (2016) EDISON: Network reconstruction and changepoint detection, Version 1.1.1
53. Chan TE, Stumpf MPH, Babtie AC (2017) Gene regulatory network inference from single-cell data using multivariate information measures. *Cell Syst* 5(3):251–267.e3

54. Filippi S, Holmes CC (2017) A Bayesian nonparametric approach to testing for dependence between random variables. *Bayesian Anal* 12(4):919–938
55. Moignard V, Woodhouse S, Haghverdi L, Lilly AJ, Tanaka Y, Wilkinson AC, Buettner F, Macaulay IC, Jawaid W, Diamanti E, Nishikawa SI, Piterman N, Kouskoff V, Theis FJ, Fisher J, Göttgens B (2015) Decoding the regulatory network of early blood development from single-cell gene expression measurements. *Nat Biotechnol* 33(3):269–276
56. Reid JE, Wernisch L (2016) Pseudotime estimation: deconfounding single cell time series. *Bioinformatics* 32(19):2973–2980
57. Campbell KR, Yau C (2016) Order under uncertainty: robust differential expression analysis using probabilistic models for pseudotime inference. *PLoS computational biology* 12.11: e1005212
58. Trapnell C, Cacchiarelli D, Grimsby J, Pokharel P, Li S, Morse M, Lennon NJ, Livak KJ, Mikkelsen TS, Rinn JL (2014) The dynamics and regulators of cell fate decisions are revealed by pseudotemporal ordering of single cells. *Nat Biotechnol* 32(4):381–386
59. Ocone A, Haghverdi L, Mueller NS, Theis FJ (2015) Reconstructing gene regulatory dynamics from high-dimensional single-cell snapshot data. *Bioinformatics* 31(12):i89–i96
60. Matsumoto H, Kiryu H, Furusawa C, Ko MSH, Ko SBH, Gouda N, Hayashi T, Nikaido I (2017) SCODE: an efficient regulatory network inference algorithm from single-cell RNA-Seq during differentiation. *Bioinformatics* 33(15):2314–2321
61. Sanchez-Castillo M, Blanco D, Tienda-Luna IM, Carrion MC, Huang Y (2018) A Bayesian framework for the inference of gene regulatory networks from time and pseudo-time series data. *Bioinformatics* 34:964–970



Chapter 3

Overview and Evaluation of Recent Methods for Statistical Inference of Gene Regulatory Networks from Time Series Data

Marco Grzegorczyk, Andrej Aderhold, and Dirk Husmeier

Abstract

A challenging problem in systems biology is the reconstruction of gene regulatory networks from postgenomic data. A variety of reverse engineering methods from machine learning and computational statistics have been proposed in the literature. However, deciding on the best method to adopt for a particular application or data set might be a confusing task. The present chapter provides a broad overview of state-of-the-art methods with an emphasis on conceptual understanding rather than a deluge of mathematical details, and the pros and cons of the various approaches are discussed. Guidance on practical applications with pointers to publicly available software implementations are included. The chapter concludes with a comprehensive comparative benchmark study on simulated data and a real-work application taken from the current plant systems biology.

Key words Gene regulatory networks, Gaussian graphical models, Sparse regression, Hierarchical Bayesian models, Gaussian processes, Bayesian networks, Chemical model averaging, Bio-PEPA, Network inference scoring scheme, Circadian regulation, *Arabidopsis thaliana*

1 Introduction

An important and challenging problem in systems biology is the inference of biochemical pathways and regulatory networks from postgenomic data. Various reverse engineering methods have been proposed in the machine learning, computational statistics, and bioinformatics literature, and it is important to understand their relative merits and shortcomings. We start this chapter, in Subheading 2, with an overview and a brief mathematical description of 11 representative state-of-the-art methods that have been developed over the last decade. This is followed, in Subheading 3, with a practical guidance to help the reader to use these methods in their own work. Given that several alternative methods and network

reconstruction paradigms have been proposed, it is important to objectively assess the network reconstruction accuracy in a comparative evaluation context. Subheading 4 describes a workflow for a performance assessment that is based on an analysis of variance (ANOVA) scheme to systematically distinguish performance differences that are intrinsic to the method itself from other confounding factors. The section also explains how to generate realistic gene expression time series that mimic real data, but come with the advantage of a known ground truth, i.e., the underlying regulatory network is known and can thus be used for objective network reconstruction assessment. Subheading 5 presents the results of a comprehensive benchmark study, which sheds light on the network reconstruction accuracy that can be achieved with the state-of-the-art methods, both in terms of a relative performance comparison and via quantitative scores that allow the reader to gauge how well the various network reconstruction methods perform in practice. The section allows quantifying the influence of other design decisions, e.g., related to the transcriptional rate estimation, on the network reconstruction accuracy. Finally, Subheading 6 describes a real-world application: the reconstruction of the central circadian regulatory network in the model plant *Arabidopsis thaliana* from mRNA concentration time series. The section compares the results from a data-driven approach, as described in this chapter, with networks developed from first principles and specific domain knowledge, as widely available in the systems biology literature. While this chapter offers a broad overview of existing methods, it necessarily cannot be exhaustive; for additional perspectives on this problem, please also consult Chapter 2.

2 Methodology: Network Inference

The starting point of our study is the mathematical formulation of transcriptional regulation [1, 2]:

$$\dot{x}_{g,t} = \frac{dx_{g,t}}{dt} = \alpha_g + f_g(\mathbf{x}_{\pi_g,t}) - \lambda_g x_{g,t} \quad (1)$$

where $x_{g,t}$ is the mRNA concentration of gene g at time t , α_g is the basal transcription rate for gene g , λ_g is the mRNA degradation rate for gene g , $f_g(\cdot)$ is an unknown regulation function, and $\mathbf{x}_{\pi_g,t}$ is the set of gene expression values of the putative regulators π_g of gene g at time t . This fundamental equation provides the basis for learning and inference in systems biology, as described by, e.g., Lawrence et al. [3]. A common approach is to approximate the time derivative on the left-hand side by a finite difference quotient:

$$\frac{dx_{g,t}}{dt} \approx \frac{x_{g,t+\Delta t} - x_{g,t}}{\Delta t} \quad (2)$$

which for a unit time delay $\Delta t = 1$ leads to

$$x_{g,t+1} = x_{g,t} + \alpha_g + f_g(\mathbf{x}_{\pi_g}, t) - \lambda_g x_{g,t} = h(x_{g,t}, \mathbf{x}_{\pi_g,t}) \quad (3)$$

for some function $h(x_{g,t}, \mathbf{x}_{\pi_g,t})$. This equation provides the basis for a variety of “dynamic” algorithms, including dynamic Bayesian networks [4], time-delay mutual information methods [5] and time-shifted regression methods [6]. However, as we demonstrate in more detail in Subheading 5.1, the finite difference approximation of Eq. (2) is not particularly good, and we therefore work with the explicit representation of Eq. (1). This might look like a “static” method, as no time-shift operation is needed, but the dynamics are explicitly represented by the time derivative $y_{g,t} = \frac{dx_{g,t}}{dt}$. For each individual gene $g = 1, \dots, N$, we use its observed mRNA concentrations $x_{g,1}, \dots, x_{g,T}$ to compute the gradients $y_{g,t}$ ($t = 1, \dots, T$), and we then consider the gradients $y_{g,1}, \dots, y_{g,T}$ as realizations of a target variable y_g , which monitors the transcription rates of gene g over time. Henceforth, we refer to the variable y_g and its realizations $y_{g,t}$ ($t = 1, \dots, T$) as the mRNA concentration time derivatives, gradients, or transcription rates synonymously. Mathematically, our goal is to find the regulators of each target variable y_g ($g = 1, \dots, N$), i.e., to identify variables with an effect on the transcription rates y_g of gene g .

2.1 Notation

For the models reviewed in this chapter, we have target variables y_g ($g = 1, \dots, N$), each representing the mRNA time derivative or gradient of a particular gene g . The realizations of each target variable y_g can then be written as a vector $\mathbf{y}_g = (y_{g,1}, \dots, y_{g,T})^\top$, where $y_{g,t}$ is the realization (or observation) of y_g at data point t . As we consider sets of time series, we refer to the index t as time point and data point synonymously, in particular we also say that $y_{g,t}$ is the observation of y_g at time t . For each gene g , there are G_g *potential* regulators, $x_1^g, \dots, x_{G_g}^g$, which are either gene or protein concentrations (see Note 1). The task is to infer a set of regulators π_g with $\pi_g \subset \{x_1^g, \dots, x_{G_g}^g\}$ for each target variable y_g (see Note 2). The collection of regulators $\{\pi_1, \dots, \pi_N\}$ can then be thought of as a regulatory interaction graph, \mathcal{M} . In \mathcal{M} , the regulators and the target variables represent the nodes and from each regulator in π_g a directed edge is pointing to the target node y_g . Hence, in terms of graphical models the graph \mathcal{M} possesses a bipartite structure, where the *potential* regulators $x_1^g, \dots, x_{G_g}^g$ are the potential parent nodes of the target variable y_g ($g = 1, \dots, N$), and there is a directed edge from x_i^g to y_g in \mathcal{M} , symbolically $x_i^g \rightarrow y_g$, if $x_i^g \in \pi_g$. In regression models, the regulators are usually referred to as covariates, and throughout this chapter we therefore use the terms regulator(s), parent node(s), and covariate(s) interchangeably.

In regression models, the observations of all the *potential* covariates of the target y_g can be collected in a design matrix \mathbf{X}_g such that each row of \mathbf{X}_g corresponds to a covariate and contains all T observations of that particular covariate. An additional row with constant elements equal to 1 is added to \mathbf{X}_g to take the intercept into account. In addition, for a fixed subset of covariates, π_g , we define \mathbf{X}_{π_g} to be the sub-matrix of the full design matrix, \mathbf{X}_g , where all rows that belong to covariates which are not in π_g have been deleted. To paraphrase that, in the restricted design matrix \mathbf{X}_{π_g} we keep only those rows of \mathbf{X}_g that correspond either to the intercept or to the covariates in the set π_g .

For non-regression models, we additionally define two vectors. For $t = 1, \dots, T$, let $\mathbf{x}_{g,t} := (x_{1,t}^g, \dots, x_{G_g,t}^g)^\top$ denote the vector of the concentrations of all G_g *potential* regulators for gene g at time t . Let $\mathbf{z}_{g,t} := (y_{g,t}, \mathbf{x}_{g,t}^\top)^\top$ extend the vector $\mathbf{x}_{g,t}$ by including the value of the response $y_{g,t}$, i.e., the derivative of the concentration of the target gene g at time t ($t = 1, \dots, T$). In addition, for a fixed subset of regulators, π_g , we define $\mathbf{x}_{\pi_g,t}$ and $\mathbf{z}_{\pi_g,t}$ to be the corresponding sub-vectors of $\mathbf{x}_{g,t}$ and $\mathbf{z}_{g,t}$, respectively, where all elements that do not correspond to regulators in π_g have been deleted.

We note that motivated by Eq. (1), all methods described in this chapter aim to predict the time derivative of a target gene's mRNA concentrations from the concentrations of the putative regulators. Where a method has not been originally designed for this purpose, a few trivial modifications have to be implemented; for example, for a dynamic method that aims to predict time-shifted target mRNA concentrations at time point $t + 1$ from mRNA concentrations at time point t , the time shift has to be undone, and the target mRNA concentration has to be replaced by its time derivative. Motivated by Eq. (1), a forced link from a target gene's mRNA concentration to its time derivative is built into all regression methods to allow for mRNA degradation (represented by the linear decay term in Eq. (1)); this is a natural implementation of biological prior knowledge about the nature of transcriptional regulation.

Finally, denote by \mathbf{X}_g^* and $\mathbf{X}_{\pi_g}^*$ the sub-matrices of the design matrices \mathbf{X}_g and \mathbf{X}_{π_g} in which the constant row for the intercept has been removed. For the state-space models (SSMs), described in Subheading 2.8, we define $\mathbf{x}_{.,t}$ as the vector of the observations of *all* potential regulators at time t (see Note 3). A complete overview of the notation is given in Table 1.

2.2 Gaussian Graphical Models (GGM)

The method of Gaussian graphical models (GGMs) is based on the insight that for random vectors \mathbf{z} from a multivariate Gaussian distribution, $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{C})$, the components z_g and $z_{g'}$, corresponding, e.g., to two genes g and g' , are stochastically

Table 1
Overview of all symbols, introduced in Subheading 2

Symbol	Short verbal description
g	Target gene g ($g = 1, \dots, N$)
x_g	Variable measuring the mRNA concentration of target gene g
$x_{g,t}$	Variable x_g at time t ($t = 1, \dots, T$)
y_g	Target (response) variable, gradient corresponding to target gene g
$y_{g,t}$	Target (response) variable y_g at time t , derivative of x_g at time t ($t = 1, \dots, T$)
$\mathbf{y}_{\cdot,t}$	Vector of all target variables (gradients) at time t , $\mathbf{y}_{\cdot,t} := (y_{1,t}, \dots, y_{N,t})^T$
\mathbf{y}_g	Vector of all T observations for the target gene y_g , $\mathbf{y}_g := (y_{g,1}, \dots, y_{g,T})^T$
G_g	The number of potential regulators for target gene g
x_i^g	The i -th regulator for target gene g ($i = 1, \dots, G_g$)
$x_{i,t}^g$	The observation for the i -th regulator for target gene g at time t
π_g	Concrete set of regulators (covariates, parent nodes) for target gene g , $\pi_g \subset \{x_1^g, \dots, x_{G_g}^g\}$
\mathcal{M}	The bipartite graph structure $\mathcal{M} = \{\pi_1, \dots, \pi_N\}$
\mathbf{X}_g	Full design matrix for gene g including all G_g potential regulators for g
\mathbf{X}_{π_g}	Restricted design matrix for gene g , \mathbf{X}_g restricted to regulators in the set π_g
$\mathbf{x}_{g,t}$	Vector of observations for all G_g regulators of gene g at time t , $\mathbf{x}_{g,t} := (x_{1,t}^g, \dots, x_{G_g,t}^g)^T$
$\mathbf{z}_{g,t}$	Response variable for gene g and concentrations of all its G_g potential regulators at time t , $\mathbf{z}_{g,t} := (y_{g,t}, \mathbf{x}_{g,t}^T)^T$
$\mathbf{x}_{\pi_g,t}$	Vector of observations for the $ \pi_g $ regulators in π_g at time t
$\mathbf{z}_{\pi_g,t}$	Response variable for gene g and concentrations of its regulators in the set π_g at time t , $\mathbf{z}_{\pi_g,t} := (y_{g,t}, \mathbf{x}_{\pi_g,t}^T)^T$
\mathbf{X}_g^*	The matrix (or set) of all T observations for the G_g potential regulators of g similar to the full design matrix \mathbf{X}_g , but without the row for the intercept
$\mathbf{X}_{\pi_g}^*$	The matrix (or set) of all T observations for the regulators in π_g similar to the restricted design matrix \mathbf{X}_{π_g} , but without the intercept row
$\mathbf{x}_{\cdot,t}$	Vector of observations of <i>all</i> potential regulators at time t , i.e., this vector includes every available regulator, and it is not target specific

These notations are used throughout this chapter. For more detailed descriptions, see main text in Subheading 2

independent conditional on the remaining system

$$p(z_g, z_{g'} | \{z_i\}_{i \neq g, g'}) = p(z_g | \{z_i\}_{i \neq g, g'}) p(z_{g'} | \{z_i\}_{i \neq g, g'}) \quad (4)$$

if and only if the corresponding element (g, g') in the inverse covariance matrix \mathbf{C}^{-1} is zero. Hence, if \mathbf{C} is known, an undirected graph of gene dependence structures can be obtained by connecting all genes (g, g') with $[\mathbf{C}^{-1}]_{g,g'} \neq 0$ by an (undirected)

edge. In practice, \mathbf{C} is unknown and has to be approximated by the empirical covariance matrix:

$$\mathbf{S} = \frac{1}{(T-1)} \sum_{t=1}^T (\mathbf{z}_t - \bar{\mathbf{z}})(\mathbf{z}_t - \bar{\mathbf{z}})^T \quad (5)$$

where $\mathbf{z}_1, \dots, \mathbf{z}_T$ is an i.i.d. sample. If T is less than the dimension of \mathbf{z}_t , then the estimated covariance matrix \mathbf{S} is rank deficient. To deal with this problem, two main approaches have been proposed. The first approach, proposed in [7], is to use shrinkage and replace the empirical covariance matrix \mathbf{S} by the following regularized matrix:

$$\mathbf{S}^* = (1 - \lambda)\mathbf{S} + \lambda\mathbf{I} \quad (6)$$

where \mathbf{I} is the identity matrix [various alternatives are discussed in 7] and $\lambda > 0$ is a regularization parameter, which can be optimized with empirical risk minimization; see Eqns. (8) and (10) in [7] for explicit expressions. The second approach, proposed by Friedman et al. [8] and termed “Glasso” (for “Graphical Lasso”) is to maximize the penalized likelihood subject to an L1-regularization term applied to the matrix elements:

$$\log \det(\boldsymbol{\Theta}) - \text{trace}(\mathbf{S}\boldsymbol{\Theta}) - \lambda \|\boldsymbol{\Theta}\|_1 \quad (7)$$

where $\boldsymbol{\Theta}$ is the estimated inverse covariance matrix. To apply GGMs to the reconstruction of gene regulatory networks, we consider the random vectors $\mathbf{z}_{g,t} := (y_{g,t}, \mathbf{x}_{g,t}^T)^T$, where $y_{g,t}$ is the time derivative of the mRNA concentration of target gene g at time t , see Eq. (1), and $\mathbf{x}_{g,t}$ is the vector of the concentrations of the G_g potential regulators at time t ($t = 1, \dots, T$). For each potential target variable y_g ($g = 1, \dots, N$), we extract a GGM from the sample $\{\mathbf{z}_{g,t}\}_{t=1,\dots,T}$. We then consider the first row (or column) of the resulting precision matrix. By standardization, we obtain the partial correlations $\rho(y_g, x_j^g | \{x_i^g\}_{i \neq j})$ between the target variable y_g and its potential regulators x_j^g ($j = 1, \dots, G_g$). Note that since the direction of causality is always directed towards the target variable y_g , the edges in the reconstructed graphs are directed, symbolically: $\{x_1^g, \dots, x_{G_g}^g\} \rightarrow y_g$. Hence, the application of an algorithm to convert undirected into directed edges, as proposed by Opgen-Rhein and Strimmer [9], becomes obsolete (see **Note 4**). The absolute values of the partial correlations $|\rho(y_g, x_j^g | \{x_i^g\}_{i \neq j})|$ can be used to score the regulatory interactions $x_j^g \rightarrow y_g$ ($g = 1, \dots, N$ and $j = 1, \dots, G_g$) with respect to their strengths.

2.3 Lasso and Elastic Net

An efficient and widely applied linear regression method that provides network sparsity is the Least Absolute Shrinkage and Selection Operator (Lasso) introduced by Tibshirani [10]. The

Lasso optimizes the regression parameters \mathbf{w}_g of a linear model based on the residual sum of squares subject to an $L1$ -norm penalty term, $\lambda_1 \|\mathbf{w}_g\|_1$, where λ_1 is a regularization parameter, and $\|\mathbf{w}_g\|_1$ is the sum of the absolute values of the components of \mathbf{w}_g (see Note 5):

$$\hat{\mathbf{w}}_g = \operatorname{argmin} \left\{ \|\mathbf{y}_g - \mathbf{X}_g^\top \mathbf{w}_g\|_2^2 + \lambda_1 \|\mathbf{w}_g\|_1 \right\} \quad (8)$$

For definitions of the full design matrix \mathbf{X}_g and the target gradient vector \mathbf{y}_g , see Table 1. Equation (8) is a convex optimization problem, for which a variety of fast and effective algorithms exist [e.g., 11]. The effect of Eq. (8) is to simultaneously shrink and prune the parameters in \mathbf{w}_g , thereby promoting a sparse network. The degree of sparsity depends on the regularization parameter λ_1 , which can be optimized with cross-validation or information criteria, like BIC.

The shortcomings are that the Lasso will only select one predictor from a set of highly correlated variables, and that it can maximally select T variables, thereby potentially suffering from saturation effects. These difficulties are addressed with the Elastic Net method, proposed by Zou and Hastie [12], which combines the Lasso penalty with a ridge regression penalty that constitutes a squared $L2$ -norm $\|\mathbf{w}_g\|_2^2$:

$$\hat{\mathbf{w}}_g = \operatorname{argmin} \left\{ \|\mathbf{y}_g - \mathbf{X}_g^\top \mathbf{w}_g\|_2^2 + \lambda_1 \|\mathbf{w}_g\|_1 + \lambda_2 \|\mathbf{w}_g\|_2^2 \right\} \quad (9)$$

Again, this is a convex optimization problem for which effective algorithms exist, and the regularization parameters λ_1 and λ_2 can be optimized with cross-validation or BIC. For these two approaches (Lasso and Elastic Net), one can use the absolute values of the elements of the estimated regression parameter vectors $\hat{\mathbf{w}}_g$ to rank the regulatory influences on y_g ($g = 1, \dots, G$) with respect to their effect.

2.4 Time-Varying Sparse Regression (Tesla)

Ahmed and Xing [13] proposed a time-varying generalization of sparse regression, which they called Tesla. The idea is to divide a time series into segments and perform sparse regression for each time series segment separately, subject to an additional sparsity constraint that penalizes differences between regression parameters associated with adjacent time series segments. Consider a time series of expression values for gene g , which is divided into \mathcal{K}_g disjunct segments, marked by \mathcal{K}_g+1 demarcation points $1 = \tau_{g,1} \leq \dots \leq \tau_{g,b} \leq \dots \leq \tau_{(g+1)} = T$. Each segment is associated with a different set of regression parameters, $\mathbf{w}_{g,b}$, where $b \in \{1, \dots, \mathcal{K}_g\}$ is a label that identifies the segment. To prevent overcomplexity and avoid overfitting, an additional $L1$ -norm penalty is imposed

on the parameter differences for adjacent time series segments, i.e., $\mathbf{w}_{g,h} - \mathbf{w}_{g,h-1}$ for $h > 1$:

$$\hat{\mathbf{w}}_{g,1}, \dots, \hat{\mathbf{w}}_{g,\mathcal{K}_g} = \operatorname{argmin} \left\{ \sum_{h=1}^{\mathcal{K}_g} \|\mathbf{y}_{g,h} - \mathbf{X}_{g,h}^\top \mathbf{w}_{g,h}\|_2^2 + \lambda_1 \sum_{h=1}^{\mathcal{K}_g} \|\mathbf{w}_{g,h}\|_1 \right. \\ \left. + \lambda_2 \sum_{h=2}^{\mathcal{K}_g} \|\mathbf{w}_{g,h} - \mathbf{w}_{g,h-1}\|_1 \right\} \quad (10)$$

where $\mathbf{y}_{g,h} = (y_{g,(\tau_{g,h}+1)}, \dots, y_{g,\tau_{g,h+1}})^\top$ is the subvector of observations in the temporal segment h , and $\mathbf{X}_{g,h}$ is the corresponding segment-specific design matrix. Given the regularization parameters λ_1 and λ_2 , the optimal regression parameters $\{\hat{\mathbf{w}}_{g,h}\}$ can be found with convex programming [13]. The regularization parameters themselves can be optimized with cross-validation or information criteria, like BIC. Note that different genes g can have different time series segmentations, with different values of \mathcal{K}_g , and that the segmentations have to be defined in advance. General guidelines for the choice of coarseness of segmentation can be found in the publication of Ahmed and Xing [13]. For applications in plant systems biology, as discussed later in this chapter, the segmentation is naturally suggested by the light phase (day, twilight, and night). Also, note that the original formulation of Tesla, proposed by Ahmed and Xing [13], is for logistic regression and binary data. The modification to linear regression, as in Eq.(10), is straightforward and more appropriate for applications to nonbinary data.

2.5 Hierarchical Bayesian Regression Models (HBR)

In the hierarchical Bayesian regression (HBR) approach, we assume a linear regression model for the target vectors \mathbf{y}_g :

$$\mathbf{y}_g | (\mathbf{w}_g, \sigma_g, \mathbf{X}_{\pi_g}) \sim \mathcal{N}(\mathbf{X}_{\pi_g}^\top \mathbf{w}_g, \sigma_g^2 \mathbf{I}) \quad (11)$$

where \mathbf{w}_g is the vector of regression parameters, \mathbf{X}_{π_g} is the restricted design matrix whose rows correspond to the variables in the covariate set π_g with an additional constant row for the intercept, and σ_g^2 is the noise variance. We impose a Gaussian prior on the regression parameter vector:

$$\mathbf{w}_g | (\sigma_g, \delta_g, \mathbf{X}_{\pi_g}) \sim \mathcal{N}(\mathbf{0}, \delta_g \sigma_g^2 \mathbf{I}) \quad (12)$$

The hyperparameter δ_g can be interpreted as the “signal-to-noise” (SNR) ratio [14]. For the posterior distribution, we get [e.g., 15, Section 3.3]:

$$\mathbf{w}_g | (\sigma_g, \delta_g, \mathbf{X}_{\pi_g}, \mathbf{y}_g) \sim \mathcal{N}(\boldsymbol{\Sigma}_g \mathbf{X}_{\pi_g} \mathbf{y}_g, \sigma_g^2 \boldsymbol{\Sigma}_g) \quad (13)$$

where $\boldsymbol{\Sigma}_g^{-1} = \delta_g^{-1}\mathbf{I} + \mathbf{X}_{\pi_g}\mathbf{X}_{\pi_g}^\top$. The marginal likelihood, $p(\mathbf{y}_g|\mathbf{X}_{\pi_g}, \sigma_g^2, \delta_g)$, can be obtained by application of standard results for Gaussian integrals [e.g., 15, Appendix B]:

$$\begin{aligned} p(\mathbf{y}_g|\mathbf{X}_{\pi_g}, \sigma_g^2, \delta_g) &= \int p(\mathbf{y}_g|\mathbf{X}_{\pi_g}, \sigma_g^2, \mathbf{w}_g)p(\mathbf{w}_g|\sigma_g^2, \delta_g, \mathbf{X}_{\pi_g})d\mathbf{w}_g \\ &= \mathcal{N}(\mathbf{y}_g|\mathbf{0}, \sigma_g^2(\mathbf{I} + \delta_g\mathbf{X}_{\pi_g}^\top\mathbf{X}_{\pi_g})) \end{aligned} \quad (14)$$

For σ_g^{-2} and δ_g^{-2} , we choose conjugate gamma priors, $\sigma_g^{-2} \sim \text{Gam}(v, v)$, and $\delta_g^{-1} \sim \text{Gam}(A_\delta, B_\delta)$ (see Note 6). The integral resulting from the marginalization over σ_g^{-2} ,

$$p(\mathbf{y}_g|\mathbf{X}_{\pi_g}, \delta_g) = \int_0^\infty p(\mathbf{y}_g|\mathbf{X}_{\pi_g}, \sigma_g^2, \delta_g)p(\sigma_g^{-2}|v)d\sigma_g^{-2} \quad (15)$$

is a multivariate Student t -distribution with a closed-form solution [e.g., 14, 15].

Given the data for all the potential regulators of y_g , i.e., given the full design matrix \mathbf{X}_g , the objective is to infer the set of covariates π_g from the marginal posterior distribution:

$$P(\pi_g|\mathbf{X}_g, \mathbf{y}_g, \delta_g) = \frac{P(\pi_g)p(\mathbf{y}_g|\mathbf{X}_{\pi_g}, \delta_g)}{\sum_{\pi_g^*} P(\pi_g^*)p(\mathbf{y}_g|\mathbf{X}_{\pi_g^*}, \delta_g)} \propto P(\pi_g)p(\mathbf{y}_g|\mathbf{X}_{\pi_g}, \delta_g) \quad (16)$$

where the sum in the denominator is over all valid covariate sets, π_g^* , and $P(\pi_g)$ is a uniform distribution over all covariate sets subject to a maximal cardinality, typically $|\pi_g| \leq 3$. We sample sets of covariates (or regulators) π_g , signal-to-noise hyper-parameters δ_g , and noise variances σ_g^2 from the joint posterior distribution with Markov chain Monte Carlo (MCMC), following the Metropolis–Hastings within partially collapsed Gibbs scheme from Grzegorczyk and Husmeier [14]. Within that scheme, we sample covariate sets π_g from Eq. (16) with Metropolis–Hastings, using the proposal mechanism from Grzegorczyk and Husmeier [14]: given the current covariate set π_g , randomly propose a new covariate set from the system of all covariate sets such that it can be reached: (1) either by removing a single covariate from π_g , (2) or by adding a single covariate to π_g , (3) or by a covariate flip move. The (hyper-)parameters δ_g^{-1} , \mathbf{w}_g , and σ_g^{-2} can be sampled with Gibbs sampling steps. As shown in Grzegorczyk and Husmeier [14], the full conditional distributions of δ_g^{-1} and \mathbf{w}_g are given by:

$$\delta_g^{-1} | (\mathbf{w}_g, \sigma_g^2) \sim \text{Gam}\left(A_\delta + \frac{|\pi_g| + 1}{2}, B_\delta + \frac{1}{2\sigma_g^2} \mathbf{w}_g^\top \mathbf{w}_g\right) \quad (17)$$

$$\mathbf{w}_g | (\mathbf{y}_g, \mathbf{X}_{\pi_g}, \sigma_g^2, \delta_g) \sim \mathcal{N}(\boldsymbol{\Sigma}_g^\star \mathbf{X}_{\pi_g} \mathbf{y}_g, \sigma_g^2 \boldsymbol{\Sigma}_g^\star) \quad (18)$$

where $|\pi_g|$ is the cardinality of the parent set, π_g , and $\Sigma_g^* = \left(\delta_g^{-1}\mathbf{I} + \mathbf{X}_{\pi_g}\mathbf{X}_{\pi_g}^\top\right)^{-1}$. The inverse variance hyperparameters, σ_g^{-2} , can be sampled with a collapsed Gibbs sampling step, in which the regression parameter vectors, \mathbf{w}_g , have been integrated out. This marginalization yields [e.g., 14]:

$$\sigma_g^{-2} | (\mathbf{y}_g, \mathbf{X}_{\pi_g}, \delta_g) \sim \text{Gam}\left(v + \frac{T}{2}, v + \frac{\mathbf{y}_g^\top (\mathbf{I} + \delta_g \mathbf{X}_{\pi_g}^\top \mathbf{X}_{\pi_g})^{-1} \mathbf{y}_g}{2}\right) \quad (19)$$

where T is the number of observations.

2.6 Sparse Bayesian Regression (SBR)

The method of automatic relevance determination (ARD) in the context of sparse Bayesian regression (SBR) was proposed by Tipping [16] and was first applied to learning gene regulation networks by [17]. It is related to the Bayesian regression method discussed in Subheading 2.5, with the following modification of the prior on the regression parameters \mathbf{w}_g : Eq. (12) is replaced by:

$$p(\mathbf{w}_g | \alpha_g) = \mathcal{N}(\mathbf{0}, \text{diag}[\alpha_g]^{-1}) \quad (20)$$

where α_g is a vector of interaction hyperparameters of the same dimension as \mathbf{w}_g , and $\text{diag}[\alpha_g]$ is a diagonal matrix with α_g in the diagonal. The marginal likelihood, Eq. (14), now becomes

$$\begin{aligned} p(\mathbf{y}_g | \mathbf{X}_g, \sigma_g^2, \alpha_g) &= \int p(\mathbf{y}_g | \mathbf{X}_g, \sigma_g^2, \mathbf{w}_g) p(\mathbf{w}_g | \alpha_g) d\mathbf{w}_g \\ &= \mathcal{N}(\mathbf{y}_g | \mathbf{0}, \sigma_g^{-2}\mathbf{I} + \mathbf{X}_g^\top \text{diag}[\alpha_g]^{-1} \mathbf{X}_g) \end{aligned} \quad (21)$$

and is optimized with respect to the hyperparameters α_g in a maximum likelihood type-II manner. Note that as opposed to Eq. (14), Eq. (21) depends on the full design matrix \mathbf{X}_g , not the design matrix restricted to a subset of regulators π_g , \mathbf{X}_{π_g} , and the discrete search in structure space, π_g , is replaced by a continuous search in hyperparameter space, α_g , which is much faster. Hyperparameters $\alpha_{g,i}$ associated with irrelevant regulators x_i^g will be driven to $\alpha_{g,i} \rightarrow \infty$, as explained in Section 13.7 of [18]. The consequence is that the associated regression parameters will be driven to zero, $w_{g,i} \rightarrow 0$, and irrelevant regulators x_i^g will effectively be pruned; hence the name “automatic relevance determination” (ARD). For fixed values of the hyperparameters, the posterior of the regression parameters \mathbf{w}_g can be obtained, and the method was therefore called “sparse Bayesian regression” (SBR). However, as opposed to the fully Bayesian method discussed in Subheading 2.5, SBR is only “Bayesian” about the values of the regression

parameters \mathbf{w}_g and does not reflect any uncertainty about $\boldsymbol{\alpha}_g$, which is typically of more interest. Hence, in comparison with Subheading 2.5, SBR gains computational speed at the expense of less thorough, approximate inference. How does SBR compare with the sparse regression methods of Subheading 2.3? As shown in Section 5 of [16], the interaction parameters $\boldsymbol{\alpha}_g$ can in principle be integrated out analytically (although this is not advisable for computational reasons). The resulting prior distribution of the regression parameters is $p(w_{g,i}) \propto \frac{1}{|w_{g,i}|}$, where $w_{g,i}$ is the i -th element of the regression parameter vector \mathbf{w}_g . The latter prior has more probability mass for $w_{g,i} \rightarrow 0$ than the Lasso prior, $p(w_{g,i}) \propto \exp(-|w_{g,i}|)$. Hence, SBR will lead to sparser network structures than Lasso. In the same way as for the Lasso, we can use the absolute values of the elements of the estimated regression parameter vectors, $\hat{\mathbf{w}}_g$, to rank the regulatory effects on the target variable y_g ($g = 1, \dots, G$) with respect to their strengths.

2.7 Bayesian Spline Autoregression (BSA)

The Bayesian spline autoregression method (BSA) proposed by Morrissey et al. [6] is related to the hierarchical Bayesian regression method of Subheading 2.5 with the essential difference that in the restricted design matrix \mathbf{X}_{π_g} the original covariates are augmented with m B-spline basis functions of degree l defined over a set of k evenly spaced knots, where (m, l, k) are user-defined parameters. Consequently, the strength of the interaction between a regulator x_i^g and the target variable y_g , which was modeled with a scalar in the method of Subheading 2.5, now becomes a vector. That is, each individual element $w_{g,i}$ of the regression parameter vector $\mathbf{w}_g := (w_{g,0}, w_{g,1}, \dots, w_{g,G_g})^\top$, where $i = 0$ corresponds to the intercept, is substituted for a vector $\mathbf{w}_{g,i}$, spanning the entire range of B-spline basis functions. To deal with the increased dimension of the resulting total parameter vector $\mathbf{w}_g := (w_{g,0}^\top, w_{g,1}^\top, \dots, w_{g,G_g}^\top)^\top$ and encourage network sparsity, a spike-and-slab-like Bayesian variable selection scheme, first proposed by Smith and Kohn [19], is used. Define $\mathbf{w}_{g,i} = \gamma_{g,i} \mathbf{u}_{g,i}$, where $\gamma_{g,i} \in \{0, 1\}$ is a binary variable to indicate whether the interaction $x_i^g \rightarrow y_g$ is on ($\gamma_{g,i} = 1$) or off ($\gamma_{g,i} = 0$). The indicator variables $\gamma_{g,i}$ are given a Beta-Bernoulli prior, meaning a Bernoulli prior on $\gamma_{g,i}$ with hyperparameters from a Beta distribution. The higher-level hyperparameters of the Beta distribution have a Jeffreys prior. The parameter vectors $\mathbf{u}_{g,i}$ are given a Gaussian prior to shrink them towards the origin:

$$p(\mathbf{u}_{g,i} | \tau_{g,i}) = \mathcal{N}(\mathbf{u}_{g,i} | \mathbf{0}, \tau_{g,i} \mathbf{K})$$

where the structure of the covariance matrix \mathbf{K} is constructed from the second-order differences between adjacent coefficients, and $\tau_{g,i}$ is a smoothness hyperparameter that defines the trade-off between fitting an interpolating spline ($\tau_{g,i} \rightarrow 0$) and a straight line ($\tau_{g,i} \rightarrow \infty$). Several priors for $\tau_{g,i}$ were tested by Morrissey

et al. [6], with the best performance achieved with an inverted Pareto distribution. Like for the hierarchical Bayesian regression method of Subheading 2.5, there is no closed-form expression for the posterior distribution, and MCMC sampling based on a Metropolis-within-Gibbs scheme is used: the technical details can be found in [6]. The resulting MCMC samples $\gamma_{g,i}^{(1)}, \dots, \gamma_{g,i}^{(H)}$ ($g = 1, \dots, G$ and $i = 1, \dots, G_g$) are used to estimate the marginal posterior probability of the regulatory interactions $x_i^g \rightarrow y_g$:

$$P(x_i^g \rightarrow y_g) = \frac{1}{H} \sum_{h=1}^H \gamma_{g,i}^{(h)} \quad (22)$$

For the Bayesian spline autoregression method, we use these marginal interaction posterior probabilities to score the regulatory interactions with respect to their strengths. The method was originally designed for time series data of the form of Eq. (3). However, as already discussed at the beginning of Subheading 2 and confirmed in Subheading 5, the underlying approximation Eq. (2) is sub-optimal. In the benchmark study of Subheading 5, the method was therefore also applied to target variables $y_{g,t}$ of the form of Eq. (1).

2.8 State-Space Models (SSM)

The state-space model (SSM) proposed by Beal et al. [20] is a Kalman filter with additional Markovian dependence among the observation vectors, and additional dependence of the latent vectors on the observation vectors from the previous time point; see Eqns. (6–7) in [20]. The parameters are estimated with variational Bayesian inference; since all distributions are multivariate Gaussian, this gives closed-form update equations that are carried out iteratively with a modified version of the expectation maximization algorithm. From these parameters, interaction strengths among the genes can be derived; see Eq. (8) in [20] for an explicit expression. The interactions contain two separate contributions: direct interactions, describing how gene expression values at the previous time point influence the current expression values, and indirect interactions, modeling gene interactions mediated via the unobserved latent factors. The dimension of the latent vector is unknown and needs to be set using cross-validation or an estimate of the lower bound on the marginal likelihood. The intrinsic Markovian nature of the SSM from [20] is consistent with Eq. (3), but not with Eq. (1). However, a modification to our data format is straightforward by reverting to an alternative form of the SSM, proposed in [21], Chapter 5, and shown in Fig. 1. In fact, the model in [20] is equivalent to the one in [21], with the external inputs replaced by the previous observations. The mathematical form of the model is as follows:

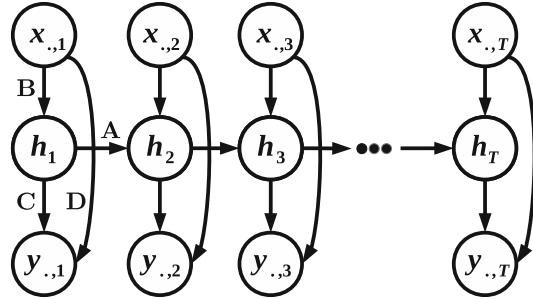


Fig. 1 Graphical model representation of the state-space model (SSM). The figure is adapted from Figure 5.2 of [21]. $\mathbf{y}_{\cdot,t}$ represents the vector of all response variables (i.e., mRNA concentration derivatives) at time t . $\mathbf{x}_{\cdot,t}$ represents the vector of all potential regulators at time t ; depending on the problem, these are either mRNA or protein concentrations. \mathbf{h}_t denotes the vector of unknown latent factors at time t . The arrows indicate probabilistic dependence relations. The parameters of the model are the four transition matrices shown in capital letters $\mathbf{A}, \mathbf{B}, \mathbf{C}$, and \mathbf{D} . These parameters are given prior distributions, which depend on further hyperparameters. For the full hierarchical Bayesian model representation, see [21]

$$\mathbf{h}_{t+1} = \mathbf{Ah}_t + \mathbf{Bx}_{\cdot,t} + \boldsymbol{\epsilon}_t$$

$$\mathbf{y}_{\cdot,t} = \mathbf{Ch}_t + \mathbf{Dx}_{\cdot,t} + \boldsymbol{\xi}_t$$

The symbols have the following meaning: $\mathbf{y}_{\cdot,t}$ is the vector of all response variables (i.e., mRNA concentration derivatives) at time t . $\mathbf{x}_{\cdot,t}$ is the vector of all potential regulators at time t ; these are either mRNA or protein concentrations. \mathbf{h}_t denotes the vector of unknown latent factors at time t . $\boldsymbol{\epsilon}_t$ and $\boldsymbol{\xi}_t$ are vectors of iid white Gaussian noise. The parameters of the model are the transition matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$, and \mathbf{D} . These parameters are given prior distributions, which depend on further hyperparameters. For the full hierarchical Bayesian model representation, see [21]. As described in [21], the posterior expectation of the interaction matrix $\mathbf{CB} + \mathbf{D}$ can be employed to assess the strengths of the individual network interactions; see Subheading 3.7 for details.

2.9 Gaussian Processes (GP)

Gaussian processes provide a popular method in nonparametric Bayesian statistics for defining a prior distribution directly in the function space rather than the parameter space. By definition, a Gaussian process is a collection of random variables, of which any finite subset has a joint Gaussian distribution. For a gene g , the process can be fully represented by a mean function $m_g(\cdot)$ and a covariance function $k_g(\cdot, \cdot)$:

$$f_g(\mathbf{x}_{\pi_g, t}) \sim \mathcal{GP}(m_g(\mathbf{x}_{\pi_g, t}), k_g(\mathbf{x}_{\pi_g, t}, \mathbf{x}_{\pi_g, t'})) \quad (23)$$

where $\mathbf{x}_{\pi_g, t}$ and $\mathbf{x}_{\pi_g, t'}$ are vectors of explanatory variables for target gene g ; these are the gene expression values of the set of regulators π_g , and $\mathbf{x}_{\pi_g, t}$, $\mathbf{x}_{\pi_g, t'}$ are the corresponding subsets of \mathbf{X}_{π_g} ; see Table 1 for an overview of the notation. The mean function $m_g(\cdot)$ is usually set to zero, which presents prior ignorance about the trend (i.e., we are equally unsure whether a trend is up or down). An important feature of Gaussian processes is that, due to the Gaussianity assumption, marginalization integrals have closed-form solutions. In particular, we get for the marginal likelihood, under the assumption of independent and identically distributed additive Gaussian noise with variance σ_g^2 [22]:

$$p(\mathbf{y}_g | \mathbf{X}_{\pi_g}, \boldsymbol{\theta}_g) = \frac{1}{\sqrt{(2\pi)^T |\mathbf{K}_g + \sigma_g^2 \mathbf{I}|}} \exp\left(-\frac{1}{2} \mathbf{y}_g^\top (\mathbf{K}_g + \sigma_g^2 \mathbf{I})^{-1} \mathbf{y}_g\right) \quad (24)$$

where $\mathbf{y}_g = (y_{g,1}, \dots, y_{g,T})^\top$ is a vector of target values for gene g , and \mathbf{K}_g is a T -by- T covariance matrix, with elements $K_{g,t,t'} = k_g(\mathbf{x}_{\pi_g, t}, \mathbf{x}_{\pi_g, t'})$. The arguments of the kernel function $k_g(\cdot, \cdot)$ are the vectors of gene expression values associated with the putative regulators of gene g , π_g , taken at the time points t and t' ; these vectors are extracted from the (restricted) design matrix \mathbf{X}_{π_g} . The kernel function depends on certain hyperparameters $\boldsymbol{\theta}_g$. For the widely applied squared exponential kernel:

$$k_g(\mathbf{x}_{\pi_g, t}, \mathbf{x}_{\pi_g, t'}) = a_g \exp\left(-\frac{(\mathbf{x}_{\pi_g, t} - \mathbf{x}_{\pi_g, t'})^2}{2l_g^2}\right) \quad (25)$$

these are the length scale l_g and amplitude a_g : $\boldsymbol{\theta}_g = (l_g, a_g)$. Äijö and Lähdesmäki [23] chose a Matérn class kernel:

$$\begin{aligned} k_g(\mathbf{x}_{\pi_g, t}, \mathbf{x}_{\pi_g, t'}) &= a_g \left(1 + \sqrt{\frac{3}{l_g^2}} (\mathbf{x}_{\pi_g, t} - \mathbf{x}_{\pi_g, t'})^\top (\mathbf{x}_{\pi_g, t} - \mathbf{x}_{\pi_g, t'}) \right) \\ &\quad \exp\left(-\sqrt{\frac{3}{l_g^2}} (\mathbf{x}_{\pi_g, t} - \mathbf{x}_{\pi_g, t'})^\top (\mathbf{x}_{\pi_g, t} - \mathbf{x}_{\pi_g, t'}) \right) \end{aligned} \quad (26)$$

which provides a better compromise between smoothness and roughness. Like for the squared exponential kernel, the hyperparameters $\boldsymbol{\theta}_g$ consist of a length scale and an amplitude parameter: $\boldsymbol{\theta}_g = (l_g, a_g)$. In order to apply Gaussian processes to the inference of gene regulatory networks, Äijö and Lähdesmäki [23] have proposed the following procedure. The starting point is the mathematical formulation of transcriptional regulation of Eq. (1), whose right-hand side can be reformulated as follows:

$$\tilde{f}_g(\mathbf{x}_{\pi_g, t}) = f_g(\mathbf{x}_{\pi_g, t}) + \mathbf{h}_g^\top \boldsymbol{\beta}_g \quad (27)$$

where $\beta_g = (\alpha_g, \lambda_g)$ and $\mathbf{h}_g = (1, -x_{g,t})$. Äijö and Lähdesmäki [23] then impose a normal distribution with mean vector \mathbf{b} and covariance matrix $\mathbf{B} = \sigma_b^2 \mathbf{I}$ on β_g :

$$\beta_g \sim N(\mathbf{b}, \mathbf{B}) = N(\mathbf{b}, \sigma_b^2 \mathbf{I}) \quad (28)$$

It can then be shown [22] that a Gaussian process assumption for f_g

$$f_g(\mathbf{x}_{\pi_g, t}) \sim \mathcal{GP}(0, k_g(\mathbf{x}_{\pi_g, t}, \mathbf{x}_{\pi_g, t'})) \quad (29)$$

implies a Gaussian process for \tilde{f}_g of the following form:

$$\tilde{f}_g(\mathbf{x}_{\pi_g, t}) \sim \mathcal{GP}(\mathbf{h}_g^\top \mathbf{b}, k_g(\mathbf{x}_{\pi_g, t}, \mathbf{x}_{\pi_g, t'}) + \mathbf{h}_g^\top \mathbf{B} \mathbf{h}_g) \quad (30)$$

This gives, in modification of Eq. (24), a closed-form expression for the marginal likelihood:

$$p(\mathbf{y}_g | \mathbf{X}_{\pi_g}, \theta_g, \sigma_g^2, \mathbf{b}, \sigma_b^2) \quad (31)$$

for which the explicit expression can be obtained from Äijö and Lähdesmäki [23]. Note that the target values \mathbf{y}_g are time derivatives, which Äijö and Lähdesmäki [23] approximate by difference quotients. The hyperparameters $\theta_g = (\alpha_g, l_g)$ and the noise variance σ_g^2 are optimized so as to maximize the marginal likelihood in Eq. (31). This can be achieved with the Polack–Ribiere conjugate gradient method, as described by Rasmussen and Williams [22]. To avoid negative values of β_g , which are biologically implausible, Äijö and Lähdesmäki [23] suggested setting the hyperparameters \mathbf{b} and σ_b^2 to fixed values such that plausible values of β_g have high probability. To accomplish structure learning for a target variable y_g , the posterior probability for a selected set of regulators, π_g , can be obtained from Bayes' theorem:

$$P(\pi_g | \mathbf{y}_g, \mathbf{X}_g, \theta_g, \sigma_g^2, \mathbf{b}, \sigma_b^2) = \frac{p(\mathbf{y}_g | \mathbf{X}_{\pi_g}, \theta_g, \sigma_g^2, \mathbf{b}, \sigma_b^2) P(\pi_g)}{\sum_{g'} p(\mathbf{y}_{g'} | \mathbf{X}_{\pi_{g'}}, \theta_g, \sigma_g^2, \mathbf{b}, \sigma_b^2) P(\pi_{g'})} \quad (32)$$

where $P(\pi_g)$ is the prior probability distribution on the set of potential regulators, for which Äijö and Lähdesmäki [23] chose a uniform distribution. The posterior probability of a particular gene interaction between the i -th regulator x_i^g and the target y_g is then given by marginalization:

$$\begin{aligned} & P(x_i^g \rightarrow y_g | \mathbf{y}_g, \mathbf{X}_g, \theta_g, \sigma_g^2, \mathbf{b}, \sigma_b^2) \\ &= \sum_{\pi_g} I(x_i^g \in \pi_g) P(\pi_g | \mathbf{y}_g, \mathbf{X}_{\pi_g}, \theta_g, \sigma_g^2, \mathbf{b}, \sigma_b^2) \end{aligned} \quad (33)$$

where $I(x_i^g \in \pi_g)$ is the indicator function, which is 1 if x_i^g is in the set of regulators π_g , and zero otherwise. For larger networks, where a complete enumeration of all potential sets of regulators is computationally prohibitive, the common approach is to impose a fan-in restriction, e.g., of 3, i.e., $P(\pi_g) = 0$ if $|\pi_g| > 3$. The posterior distribution of Eq.(33) can be used to score the regulatory interactions with respect to their strengths.

2.10 Mixture Bayesian Network Models (MBN)

A flexible Gaussian mixture model approach for inferring nonlinear network interactions has been proposed by Ko et al. [24, 25], which they call the “Mixture Bayesian network model” (see Note 7). The key idea is to model each target gene g conditional on its regulators in π_g with a conditional Gaussian mixture model. Given the vector of the variables in a regulator set π_g at time t , symbolically $\mathbf{x}_{\pi_g, t}$, we consider a Gaussian mixture model with \mathcal{K}_g mixture components and the mixture weights $\alpha_{g,1}, \dots, \alpha_{g,\mathcal{K}_g}$ for the joint distribution of the target gene $y_{g,t}$ and its regulators $\mathbf{x}_{\pi_g, t}$. Recalling the definition $\mathbf{z}_{\pi_g, t} := (y_{g,t}, \mathbf{x}_{\pi_g, t}^\top)^\top$ from Table 1, we obtain:

$$p(\mathbf{z}_{\pi_g, t}) = \sum_{h=1}^{\mathcal{K}_g} \alpha_{g,h} f_{g,h}(\mathbf{z}_{\pi_g, t}) \quad (34)$$

where each component-specific function $f_{g,h}(\cdot)$ is the density function of a $(|\pi_g| + 1)$ -dimensional Gaussian distribution with mean vector $\boldsymbol{\mu}_{g,h}$ and covariance matrix $\boldsymbol{\Sigma}_{g,h}$, and $\sum_{h=1}^{\mathcal{K}_g} \alpha_{g,h} = 1$. The marginal distribution of the vector $\mathbf{x}_{\pi_g, t}$ is then also a Gaussian mixture:

$$p(\mathbf{x}_{\pi_g, t}) = \sum_{h=1}^{\mathcal{K}_g} \alpha_{g,h} f_{g,h}^*(\mathbf{x}_{\pi_g, t}) \quad (35)$$

where the $|\pi_g|$ -dimensional Gaussian density functions $f_{g,h}^*(\cdot)$ have mean vectors $\boldsymbol{\mu}_{g,h}^*$ and covariance matrices $\boldsymbol{\Sigma}_{g,h}^*$ which are sub-vectors of $\boldsymbol{\mu}_{g,h}$ and sub-matrices of $\boldsymbol{\Sigma}_{g,h}$, respectively (see Note 8). Considering $\mathbf{z}_{\pi_g, t}$ ($t = 1, \dots, T$) as an i.i.d. sample and taking into account that the conditional distribution $p(y_{g,t} | \mathbf{x}_{\pi_g, t})$ is the ratio of the joint distribution in Eq.(34) and the marginal distribution in Eq.(35), the likelihood of the conditional Gaussian mixture model is given by:

$$\text{LL}(\mathbf{y}_g | \mathbf{X}_{\pi_g}^*, \boldsymbol{\theta}(\pi_g, \mathcal{K}_g)) = \frac{\prod_{t=1}^T \sum_{h=1}^{\mathcal{K}_g} \alpha_{g,h} f_{g,h}(\mathbf{z}_{\pi_g, t})}{\prod_{t=1}^T \sum_{h=1}^{\mathcal{K}_g} \alpha_{g,h} f_{g,h}^*(\mathbf{x}_{\pi_g, t})} \quad (36)$$

where $\theta(\pi_g, \mathcal{K}_g)$ denotes the set of mixture parameters, namely the mixture weights as well as the mean vectors and covariance matrices of the component-specific Gaussian distributions, $\mathbf{X}_{\pi_g}^*$ is the matrix of the observations of the regulators in π_g , and $\mathbf{y}_g = (y_{g,1}, \dots, y_{g,T})^\top$ is the vector of the target variable observations.

Given a fixed set of regulators, π_g , and a fixed number of mixture components, \mathcal{K}_g , the maximum likelihood (ML) estimates for the mixture parameters $\theta(\pi_g, \mathcal{K}_g)$ can be obtained with the Expectation-Maximization (EM) algorithm, as described in detail by Ko et al. [25]. Keeping π_g fixed, ML estimates, $\hat{\theta}(\pi_g, \mathcal{K}_g)$, can be computed for different numbers of mixture components \mathcal{K}_g . Having estimates $\hat{\theta}(\pi_g, \mathcal{K}_g)$ for $\mathcal{K}_g = 1, \dots, \mathcal{K}_{MAX}$, where $\mathcal{K}_{MAX} = 10$ is an imposed upper bound on the number of mixture components, the Bayesian Information Criterion (BIC) is employed to determine the *best* number of mixture components given π_g :

$$\begin{aligned} \mathcal{K}_{g|\pi_g}^{BIC} = \operatorname{argmin}_{\mathcal{K}_g} & \{-2 \log(\text{LL}(\mathbf{y}_g | \mathbf{X}_{\pi_g}^*, \hat{\theta}(\pi_g, \mathcal{K}_g))) \\ & + \log(T) |\hat{\theta}(\pi_g, \mathcal{K}_g)|\} \end{aligned} \quad (37)$$

where T is the number of observations, $|\hat{\theta}(\pi_g, \mathcal{K}_g)|$ is the number of the ML-estimated mixture parameters, and the likelihood $\text{LL}(.)$ has been defined in Eq. (36). With Eq. (37), the best number of mixture components $\mathcal{K}_{g|\pi_g}^{BIC}$ can be determined for each potential regulator set π_g . One can then systematically compute $\mathcal{K}_{g|\pi_g}^{BIC}$ for each set π_g up to a given cardinality, typically $|\pi_g| \leq 3$. Finally, the best set of regulators π_g^{BIC} for target variable y_g minimizes the BIC criterion, and is thus given by:

$$\begin{aligned} \pi_g^{BIC} = \operatorname{argmin}_{\pi_g} & \{-2 \log(\text{LL}(\mathbf{y}_g | \mathbf{x}_{\pi_g}, \hat{\theta}(\pi_g, \mathcal{K}_{g|\pi_g}^{BIC}))) \\ & + \log(T) |\hat{\theta}(\pi_g, \mathcal{K}_{g|\pi_g}^{BIC})|\} \end{aligned} \quad (38)$$

The optimization procedure outlined above is repeated several times, and we average over the obtained results, as described in Subheading 3.9, to score the individual interactions, $x_i^g \rightarrow y_g$. Each individual repetition of the optimization procedure requires the EM algorithm to be run \mathcal{K}_{MAX} times for each possible parent set π_g . For our applications, we set $\mathcal{K}_{MAX} = 10$, and we restricted the maximum cardinality of the regulator sets to $\mathcal{F} = 3$. For this setting, the EM algorithm has to be run

$$\mathcal{K}_{MAX} \cdot \sum_{i=0}^{\mathcal{F}} \binom{n}{i}$$

times, where n is the number of potential regulators for gene g (see Note 9).

2.11 Gaussian Bayesian Networks (BGe)

The BGe scoring metric was introduced by Geiger and Heckerman [26] and has become a standard modeling framework for static and dynamic Gaussian Bayesian networks (see Note 10). For $t = 1, \dots, T$, the common distribution of the target variable $y_{g,t}$ and its potential regulators $\mathbf{x}_{g,t}$ is assumed to be an i.i.d. sample from a $(G_g + 1)$ -dimensional multivariate Gaussian distribution with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$:

$$\begin{aligned} p(\mathbf{z}_{g,t} | \boldsymbol{\mu}, \boldsymbol{\Sigma}) &= (2\pi)^{-\frac{G_g+1}{2}} \det(\boldsymbol{\Sigma})^{-\frac{1}{2}} \\ &\times \exp \left\{ -\frac{1}{2} (\mathbf{z}_{g,t} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\mathbf{z}_{g,t} - \boldsymbol{\mu}) \right\} \end{aligned} \quad (39)$$

where G_g is the number of potential regulators for the target variable y_g , i.e., the length of the vector $\mathbf{x}_{\pi_g, t}$, and $\mathbf{z}_{g,t} := (y_{g,t}, \mathbf{x}_{\pi_g, t}^\top)^\top$, as defined in Table 1. Onto the unknown parameters, namely the mean vector $\boldsymbol{\mu}$ and the precision matrix $\mathbf{W} := \boldsymbol{\Sigma}^{-1}$, a normal-Wishart prior is imposed, symbolically:

$$p(\mathbf{W} | \alpha, \mathbf{T}_0) = c(G_g, \alpha) \det(\mathbf{T}_0)^{\frac{\alpha}{2}} \det(\mathbf{W})^{\frac{\alpha-G_g-1}{2}} \exp \left\{ -\frac{1}{2} \text{trace}(\mathbf{T}_0 \mathbf{W}) \right\} \quad (40)$$

$$\begin{aligned} p(\boldsymbol{\mu} | \boldsymbol{\mu}_0, (\nu \mathbf{W})^{-1}) &= (2\pi)^{-\frac{G_g+1}{2}} \det(\nu \mathbf{W})^{\frac{1}{2}} \\ &\exp \left\{ -\frac{1}{2} (\boldsymbol{\mu} - \boldsymbol{\mu}_0)^\top \nu \mathbf{W} (\boldsymbol{\mu} - \boldsymbol{\mu}_0) \right\} \end{aligned} \quad (41)$$

where

$$c(G_g, \alpha) = \left(2^{\frac{\alpha(G_g+1)}{2}} \pi^{\frac{(G_g+1)G_g}{4}} \sum_{i=1}^{G_g+1} \Gamma \left(\frac{\alpha+1-i}{2} \right) \right)^{-1}, \quad (42)$$

$\Gamma(\cdot)$ is the gamma function, and the hyperparameters α , \mathbf{T}_0 , ν , and $\boldsymbol{\mu}_0$ of the normal-Wishart distribution are chosen fixed. Geiger and Heckerman [26] show that the marginal likelihood:

$$\begin{aligned} p(\mathbf{z}_{g,1}, \dots, \mathbf{z}_{g,T}) &= \int \int \left\{ \prod_{t=1}^T p(\mathbf{z}_{g,t} | \boldsymbol{\mu}, \boldsymbol{\Sigma}) \right\} \\ &p(\boldsymbol{\mu} | \boldsymbol{\mu}_0, (\nu \mathbf{W})^{-1}) p(\mathbf{W} | \alpha, \mathbf{T}_0) d\boldsymbol{\mu} d\mathbf{W} \end{aligned} \quad (43)$$

can then be computed in closed-form. If it is further assumed that the target variable y_g , conditional on the set of regulators π_g , becomes statistically independent of all the other potential regulators, symbolically $p(y_g | \mathbf{X}_g^*, \pi_g) = p(y_g | \mathbf{X}_{\pi_g}^*)$, then the conditional

distributions:

$$p(\mathbf{y}_g | \mathbf{X}_g^*, \boldsymbol{\pi}_g) = p(\mathbf{y}_g | \mathbf{X}_{\boldsymbol{\pi}_g}^*) = \frac{p(\mathbf{y}_g, \mathbf{X}_{\boldsymbol{\pi}_g}^*)}{p(\mathbf{X}_{\boldsymbol{\pi}_g}^*)} \quad (44)$$

can also be computed in closed-form for each regulator set $\boldsymbol{\pi}_g$, see [26] for details. Imposing uniform priors on the regulator sets, $\boldsymbol{\pi}_g$, subject to a maximal cardinality restriction \mathcal{F} , the posterior distribution of the regulator set $\boldsymbol{\pi}_g$ with $|\boldsymbol{\pi}_g| \leq \mathcal{F}$ is given by:

$$P(\boldsymbol{\pi}_g | \mathbf{y}_g, \mathbf{X}_g^*) = \frac{p(\mathbf{y}_g | \mathbf{X}_{\boldsymbol{\pi}_g}^*)}{\sum_{\tilde{\boldsymbol{\pi}}_g: |\tilde{\boldsymbol{\pi}}_g| \leq \mathcal{F}} p(\mathbf{y}_g | \mathbf{X}_{\tilde{\boldsymbol{\pi}}_g}^*)} \quad (45)$$

where the sum in the denominator is over all valid regulator sets $\tilde{\boldsymbol{\pi}}_g$ whose cardinality is lower than or equal to the fan-in \mathcal{F} . The posterior probability of an interaction between x_i^g and y_g can then be computed by marginalization:

$$P(x_i^g \rightarrow y_g | \mathbf{y}_g, \mathbf{X}_g^*) = \sum_{\boldsymbol{\pi}_g} I(x_i^g \in \boldsymbol{\pi}_g) P(\boldsymbol{\pi}_g | \mathbf{y}_g, \mathbf{X}_g^*) \quad (46)$$

where $I(x_i^g \in \boldsymbol{\pi}_g)$ is the indicator function, which is 1 if x_i^g is in the set of regulators $\boldsymbol{\pi}_g$, and zero otherwise. The posterior probabilities in Eq. (46) are used to score the regulatory interactions with respect to their strengths.

2.12 Improved Chemical Model Averaging (iCheMA)

In this section, we summarize the Improved Chemical Model Averaging (iCheMA) method proposed by Aderhold et al. [27], which is an improvement on the Chemical Model Averaging (CheMA) method of Oates et al. [28]. Starting from the fundamental equation of transcriptional regulation, Eq. (1), setting the constant offset term to zero and employing Michaelis–Menten kinetics yields (see Note 11):

$$y_{i,t^*} = \frac{dx_{i,t}}{dt} |_{t=t^*} = -v_{0,i}x_{i,t^*} + \sum_{u \in \pi_i} v_{u,i} \frac{I_{u,i}x_{u,t^*} + (1 - I_{u,i})k_{u,i}}{x_{u,t^*} + k_{u,i}} \quad (47)$$

where the sum is over all genes u that are in the set of regulators π_i of gene i , and the indicator function $I_{u,i}$ indicates whether gene u is an activator ($I_{u,i} = 1$) or inhibitor ($I_{u,i} = 0$). The first term, $-v_{0,i}x_{i,t^*}$, takes the degradation of x_i into account, while $v_{u,i}$ and $k_{u,i}$ are the *maximum reaction rate* and *Michaelis–Menten* parameters for the regulatory effect of gene $u \in \pi_i$ on gene i , respectively. Equation (47) represents the typical form of transcriptional regulation; see, e.g., the supplementary material in Pokhilko et al. [29, 30]. Without loss of generality, we now assume that π_i is given by $\pi_i = \{x_1, \dots, x_s\}$. Equation (47) can then be

written in vector notation:

$$y_{i,t^*} = \frac{dx_{i,t}}{dt}|_{t=t^*} = \mathbf{D}_{i,t^*}^\top \mathbf{V}_i \quad (48)$$

where $\mathbf{V}_i = (v_{0,i}, v_{1,i}, \dots, v_{s,i})^\top$ is the vector of the maximum reaction rate parameters, and the vector \mathbf{D}_{i,t^*} depends on the measured concentrations $x_u(t^*)$ and the Michaelis–Menten parameters $k_{u,i}$ ($u \in \pi_i$) via Eq. (47):

$$\mathbf{D}_{i,t^*}^\top = \left(-x_{i,t^*}, \frac{I_{1,i}x_{1,t^*} + (1 - I_{1,i})k_{1,i}}{x_{1,t^*} + k_{1,i}}, \dots, \frac{I_{s,i}x_{s,t^*} + (1 - I_{s,i})k_{s,i}}{x_{s,t^*} + k_{s,i}} \right) \quad (49)$$

We combine the Michaelis–Menten parameters $k_{u,i}$ ($u \in \pi_i$) in a vector \mathbf{K}_i , and we arrange the T row vectors \mathbf{D}_{i,t^*}^\top ($t^* \in \{t_1, \dots, t_T\}$) in a T -by- $(|\pi_i| + 1)$ design matrix $\mathbf{D}_i = \mathbf{D}_i(\mathbf{K}_i)$. The likelihood is then:

$$p(\mathbf{y}_i | \mathbf{K}_i, \mathbf{V}_i, \sigma_i^2) = (2\pi\sigma_i^2)^{-\frac{T}{2}} e^{-\frac{1}{2\sigma_i^2} (\mathbf{y}_i - \mathbf{D}_i \mathbf{V}_i)^\top (\mathbf{y}_i - \mathbf{D}_i \mathbf{V}_i)} \quad (50)$$

where $\mathbf{y}_i := (y_{i,t_1}, \dots, y_{i,t_T})^\top$ is the vector of the rates or gradients for gene i . To ensure nonnegative Michaelis–Menten parameters, truncated Normal prior distributions are used:

$$\mathbf{K}_i \sim \mathcal{N}_{\{\mathbf{K}_i \geq 0\}}(\mathbf{1}, v\mathbf{I}) \quad (51)$$

where $\mathbf{1}$ is the one vector, \mathbf{I} is the identity matrix, $v > 0$ is a hyperparameter, and the subscript, $\{\mathbf{K}_i \geq 0\}$, indicates the truncation condition, i.e., that each element of \mathbf{K}_i has to be nonnegative. A truncated ridge regression prior is imposed on the maximum reaction rate vectors \mathbf{V}_i :

$$\mathbf{V}_i | \sigma_i^2, \delta_i^2 \sim \mathcal{N}_{\{\mathbf{V}_i \geq 0\}}(\mathbf{1}, \delta_i^2 \sigma_i^2 \mathbf{I}) \quad (52)$$

where δ_i^2 is a hyperparameter which regulates the prior strength. For σ_i^2 and δ_i^2 , we use inverse Gamma priors, $\sigma_i^2 \sim IG(a_\sigma, b_\sigma)$ and $\delta_i^2 \sim IG(a_\delta, b_\delta)$. A graphical model representation of the iCheMA model is provided in Fig. 2.

2.12.1 Posterior Inference

For computing the posterior distribution of the noise variance σ_i^2 ,

$$p(\sigma_i^2 | \mathbf{K}_i, \mathbf{y}_i) \propto p(\mathbf{y}_i | \sigma_i^2, \mathbf{K}_i) p(\sigma_i^2) \quad (53)$$

[28] approximated the marginalization integral:

$$p(\mathbf{y}_i | \sigma_i^2, \mathbf{K}_i) = \int p(\mathbf{y}_i | \mathbf{V}_i, \sigma_i^2, \mathbf{K}_i) p(\mathbf{V}_i | \sigma_i^2, \mathbf{K}_i) d\mathbf{V}_i \quad (54)$$

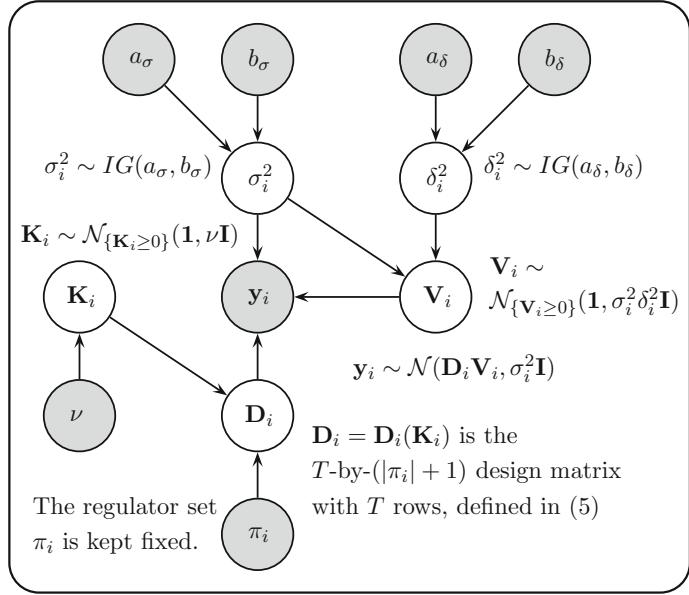


Fig. 2 Probabilistic graphical model representation of iCheMA. The figure shows a probabilistic graphical model representation of the semi-mechanistic iCheMA model described in Subheading 2.12. The symbols are defined in the main text. White nodes symbolize flexible variables that are inferred, and gray nodes represent variables that are fixed (data and higher-level hyperparameters). Figure adapted from Aderhold et al. [27]

with the closed-form solution from Marin and Robert [31], Chapter 3. This is exact if there are no restrictions on the integration bounds. However, given the underlying positivity constraint for \mathbf{V}_i , symbolically $\{\mathbf{V}_i \geq 0\}$, the integral is no longer analytically tractable and the expressions for Eqs. (53)–(54) used in Oates et al. [28] become an approximation. Aderhold et al. [27] therefore switched to an uncollapsed Gibbs sampling step, where σ_i^2 is sampled from the full conditional distribution $p(\sigma_i^2 | \mathbf{K}_i, \mathbf{V}_i, \mathbf{y}_i)$ and the marginalization integral from Eq. (54) becomes obsolete.

For gene i and a given regulator set π_i , we have to sample the maximum reaction rate vector \mathbf{V}_i , the Michaelis–Menten parameter vector \mathbf{K}_i , the noise variance σ_i^2 , and the new hyperparameter δ_i^2 from the posterior distribution:

$$p(\mathbf{V}_i, \mathbf{K}_i, \sigma_i^2, \delta_i^2 | \mathbf{y}_i) \propto p(\mathbf{y}_i | \mathbf{K}_i, \mathbf{V}_i, \sigma_i^2) p(\mathbf{V}_i | \sigma_i^2, \delta_i^2) p(\delta_i^2) p(\mathbf{K}_i) p(\sigma_i^2) \quad (55)$$

where $\mathbf{y}_i := (y_i(t_1), \dots, y_i(t_T))^\top$ is the vector of rates or gradients. For the full conditional distribution of \mathbf{V}_i , we get:

$$p(\mathbf{V}_i | \mathbf{K}_i, \sigma_i^2, \delta_i^2, \mathbf{y}_i) \propto p(\mathbf{y}_i | \mathbf{K}_i, \mathbf{V}_i, \sigma_i^2) p(\mathbf{V}_i | \delta_i^2, \sigma_i^2) \quad (56)$$

Since \mathbf{K}_i , σ_i^2 , and δ_i^2 are fixed in Eq.(56) and the (truncated) Gaussian prior on \mathbf{V}_i from Eq.(52) is conjugate to the likelihood in Eq.(50), we obtain:

$$\mathbf{V}_i | \mathbf{K}_i, \sigma_i^2, \delta_i^2, \mathbf{y}_i \sim N_{\{\mathbf{V}_i \geq 0\}}(\tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Sigma}}) \quad (57)$$

where $\tilde{\boldsymbol{\Sigma}} = \delta_i^2(\mathbf{I} + \delta_i^2 \mathbf{D}_i^\top \mathbf{D}_i)^{-1}$, $\tilde{\boldsymbol{\mu}} = \tilde{\boldsymbol{\Sigma}}(\delta_i^2 \mathbf{1} + \mathbf{D}_i^\top \mathbf{y}_i)$, and $\mathbf{D}_i = \mathbf{D}_i(\mathbf{K}_i)$ is the design matrix, built from the rows given in Eq.(49). For the full conditional distribution of δ_i^2 , we have:

$$p(\delta_i^2 | \mathbf{V}_i, \mathbf{K}_i, \sigma_i^2, \mathbf{y}_i) \propto p(\mathbf{V}_i | \sigma_i^2, \delta_i^2) p(\delta_i^2) \quad (58)$$

As \mathbf{V}_i and σ_i^2 are fixed in Eq.(58) and the inverse Gamma prior on δ_i^2 is conjugate for $p(\mathbf{V}_i | \sigma_i^2, \delta_i^2)$, defined in Eq.(52), we obtain:

$$\delta_i^2 | \mathbf{V}_i, \mathbf{K}_i, \sigma_i^2, \mathbf{y}_i \sim IG(\tilde{a}_\delta, \tilde{b}_\delta) \quad (59)$$

with $\tilde{b}_\delta = b_\delta + \frac{1}{2}\sigma_i^2(\mathbf{V}_i - \mathbf{1})^\top(\mathbf{V}_i - \mathbf{1})$, and $\tilde{a}_\delta = a_\delta + \frac{1}{2}(|\pi_i| + 1)$. For the full conditional distribution of σ_i^2 , we have:

$$p(\sigma_i^2 | \mathbf{K}_i \mathbf{V}_i, \delta_i^2, \mathbf{y}_i) \propto p(\mathbf{y}_i | \mathbf{K}_i, \mathbf{V}_i, \sigma_i^2) p(\mathbf{V}_i | \sigma_i^2, \delta_i^2) p(\sigma_i^2) \quad (60)$$

As \mathbf{K}_i , \mathbf{V}_i , and δ_i^2 are fixed in Eq.(60) and the Gaussian-Inverse-Gamma prior on $(\mathbf{V}_i, \sigma_i^2)$ is conjugate for the likelihood in Eq.(50), we get:

$$\sigma_i^2 \sim IG(\tilde{a}_\sigma, \tilde{b}_\sigma) \quad (61)$$

where $\tilde{b}_\sigma = b_\sigma + \frac{1}{2}[(\mathbf{y}_i - \mathbf{D}_i \mathbf{V}_i)^\top(\mathbf{y}_i - \mathbf{D}_i \mathbf{V}_i) + \delta_i^2(\mathbf{V}_i - \mathbf{1})^\top(\mathbf{V}_i - \mathbf{1})]$, and $\tilde{a}_\sigma = a_\sigma + \frac{1}{2}(T + |\pi_i| + 1)$. For the mathematical details, see, e.g., Chapter 3 of Marin and Robert [31].

The full conditional distribution of \mathbf{K}_i cannot be computed in closed-form so that the Michaelis–Menten parameters have to be sampled by Metropolis–Hastings MCMC steps. Given the current vector \mathbf{K}_i , a realization \mathbf{u} from a multivariate Gaussian distribution with expectation vector $\mathbf{0}$ and covariance matrix $\boldsymbol{\Sigma} = 0.1\mathbf{I}$ is sampled, and we propose the new parameter vector $\mathbf{K}_i^* = \mathbf{K}_i + \mathbf{u}$ subject to a reflection of negative values into the positive domain. The Metropolis–Hastings acceptance probability for the new vector \mathbf{K}_i^* is then $A(\mathbf{K}_i, \mathbf{K}_i^*) = \min\{1, R(\mathbf{K}_i, \mathbf{K}_i^*)\}$, with

$$R(\mathbf{K}_i, \mathbf{K}_i^*) = \frac{\exp\left\{\frac{-1}{2\sigma_i^2}(\mathbf{y}_i - \mathbf{D}_i(\mathbf{K}_i^*) \mathbf{V}_i)^\top(\mathbf{y}_i - \mathbf{D}_i(\mathbf{K}_i^*) \mathbf{V}_i)\right\}}{\exp\left\{\frac{-1}{2\sigma_i^2}(\mathbf{y}_i - \mathbf{D}_i(\mathbf{K}_i) \mathbf{V}_i)^\top(\mathbf{y}_i - \mathbf{D}_i(\mathbf{K}_i) \mathbf{V}_i)\right\}} \times PR \times HR \quad (62)$$

where the Hastings Ratio (HR) is equal to one, and the prior probability ratio (PR) is obtained from Eq. (52):

$$\text{PR}_{\text{iCheMA}} = \frac{P_{\{\mathbf{K}_i^* \geq 0\}}(\mathbf{K}_i^*)}{P_{\{\mathbf{K}_i \geq 0\}}(\mathbf{K}_i)} \quad (63)$$

If the move is accepted, we replace \mathbf{K}_i by \mathbf{K}_i^* , or otherwise we leave \mathbf{K}_i unchanged. Pseudo code of the MCMC sampling scheme for the new model variant (iCheMA) is provided in Table 2.

2.12.2 Posterior Probabilities of Interactions

The essence of the iCheMA method is the principle of “model averaging” to compute the marginal posterior probabilities of all regulator-regulatee interactions (i.e., the “edges” in the interaction graph). The marginal posterior probability for gene u being a regulator of i is given by:

$$p(u \rightarrow i | \mathbf{y}_i) = \frac{\sum_{\pi_i^\diamond \in \Pi^{(u \rightarrow i)}} p(\mathbf{y}_i | \pi_i^\diamond) p(\pi_i^\diamond)}{\sum_{\pi_i^\diamond \in \Pi} p(\mathbf{y}_i | \pi_i^\diamond) p(\pi_i^\diamond)} \quad (64)$$

where Π is the set of all possible regulator sets π_i of gene i , and $\Pi^{(u \rightarrow i)}$ is the set of all regulator sets π_i of i that contain the regulator u . For simplicity, Aderhold et al. [27] chose a uniform prior for π_i subject to a maximum cardinality of 3 for the set of regulators (“parents”) of a node. It is straightforward to replace this by a more informative prior if genuine biological prior knowledge is available. For the marginal likelihoods, one can use Chib’s method:

$$p(\mathbf{y}_i | \pi_i) = \frac{p(\mathbf{y}_i | \tilde{\theta}_i, \pi_i) p(\tilde{\theta}_i | \pi_i)}{p(\tilde{\theta}_i | \pi_i, \mathbf{y}_i)} \quad (65)$$

where the posterior near some selected “pivot” parameters $\tilde{\theta}_i$, $p(\tilde{\theta}_i | \pi_i, \mathbf{y}_i)$, is approximated with MCMC; see Chib and Jeliazkov [32] for details. There are various alternative methods that can be used, like thermodynamic integration or widely applicable information criteria. A detailed comparison is available from Aderhold et al. [27].

2.13 Rate (or Gradient) Estimation Revisited

As discussed at the beginning of this section, the fundamental concept of the interaction model is the matching of gradients between the regulator variables on the right-hand side and the rate of mRNA concentration change $\frac{dx_{g,t}}{dt}$ on the left-hand side of Eq. (1). Since direct measurements of these rates are typically unavailable, the rates need to be estimated from the mRNA concentration time series. As already discussed at the beginning of this section, and clearly demonstrated later in Subheading 5.1, the approximation of the derivative by the finite difference quotient of Eq. (2) usually leads to poor results due to noise amplification.

Table 2
Pseudo Code: MCMC sampling scheme for the iCheMA model

<p>Initialization: For gene $i \in \{1, \dots, n\}$ and a given regulator set π_i, initialize the MCMC algorithm in iteration $r = 0$ with the maximum reaction rate vector $\mathbf{V}_i^{(0)} = \mathbf{1}$, the Michaelis–Menten parameters $\mathbf{K}_i^{(0)} = \mathbf{1}$, the noise variance $\sigma_{i,(0)}^2 = 1$, and the parameter $\delta_{i,(0)}^2 = 1$.</p> <p>Let $\mathbf{y}_i = (y_i(t_1), \dots, y_i(t_T))^\top = \left(\frac{dx_i(t)}{dt} _{t=t_1}, \dots, \frac{dx_i(t)}{dt} _{t=t_T}\right)^\top$ denote the vector of the gradients, observed or computed, respectively, for gene i.</p>
<p>MCMC iterations: For $r = 1, 2, 3, \dots$</p>
<p>Given the current state $\mathbf{V}_i^{(r-1)}, \mathbf{K}_i^{(r-1)}, \sigma_{i,(r-1)}^2$, and $\delta_{i,(r-1)}^2$, successively:</p> <ul style="list-style-type: none"> Resample the maximal reaction rate parameter vector \mathbf{V}_i from its full conditional distribution:
$\mathbf{V}_i^{(r)} \sim N_{\{\mathbf{V}_i^{(r)} \geq 0\}}(\tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Sigma}}), \text{ where}$ $\tilde{\boldsymbol{\Sigma}} = \left(\delta_{i,(r-1)}^{-2} \mathbf{I} + \mathbf{D}_i^\top \mathbf{D}_i\right)^{-1}, \quad \tilde{\boldsymbol{\mu}} = \tilde{\boldsymbol{\Sigma}} \left(\delta_{i,(r-1)}^{-2} \mathbf{1} + \mathbf{D}_i^\top \mathbf{y}_i\right), \text{ and } \mathbf{D}_i = \mathbf{D}_i(\mathbf{K}_i^{(r-1)}).$
<ul style="list-style-type: none"> Resample the noise variance parameter from its full conditional distribution $\sigma_{i,(r)}^2 \sim IG(\tilde{a}_\sigma, \tilde{b}_\sigma)$, where
$\tilde{a}_\sigma = a_\sigma + \frac{1}{2}(n + \pi_i + 1), \text{ and}$ $\tilde{b}_\sigma = b_\sigma + \frac{1}{2} \left((\mathbf{y}_i - \mathbf{D}_i \mathbf{V}_i^{(r)})^\top (\mathbf{y}_i - \mathbf{D}_i \mathbf{V}_i^{(r)}) + \delta_{i,(r-1)}^2 (\mathbf{V}_i^{(r)} - \mathbf{1})^\top (\mathbf{V}_i^{(r)} - \mathbf{1}) \right)$ <p>with $\mathbf{D}_i = \mathbf{D}_i(\mathbf{K}_i^{(r-1)})$.</p>
<ul style="list-style-type: none"> Resample the δ_i-hyperparameter from its full conditional distribution $\delta_{i,(r)}^2 \sim IG(\tilde{a}_\delta, \tilde{b}_\delta)$, where
$\tilde{a}_\delta = a_\delta + \frac{1}{2}(\pi_i + 1), \text{ and}$ $\tilde{b}_\delta = b_\delta + \frac{1}{2} \sigma_{i,(r)}^2 (\mathbf{V}_i^{(r)} - \mathbf{1})^\top (\mathbf{V}_i^{(r)} - \mathbf{1})$
<ul style="list-style-type: none"> Perform a Metropolis–Hastings MCMC move that proposes to change the Michaelis–Menten parameter vector $\mathbf{K}_i^{(r-1)}$ by sampling a realization \mathbf{u} from a multivariate Gaussian distribution with expectation vector $\mathbf{0}$ and covariance $\boldsymbol{\Sigma} = 0.1 \cdot \mathbf{I}$. The newly proposed parameter vector $\mathbf{K}_i^* := \mathbf{K}_i^{(r-1)} + \mathbf{u}$ is accepted with probability $A(\mathbf{K}_i^{(r-1)}, \mathbf{K}_i^*) = \min \left\{ 1, R(\mathbf{K}_i^{(r-1)}, \mathbf{K}_i^*) \right\}$, where
$R(\mathbf{K}_i^{(r-1)}, \mathbf{K}_i^*) = \frac{\exp \left\{ \frac{-1}{2\sigma_{i,(r)}^2} (\mathbf{y}_i - \mathbf{D}_i(\mathbf{K}_i^*) \mathbf{V}_i^{(r)})^\top (\mathbf{y}_i - \mathbf{D}_i(\mathbf{K}_i^*) \mathbf{V}_i^{(r)}) \right\}}{\exp \left\{ \frac{-1}{2\sigma_{i,(r)}^2} (\mathbf{y}_i - \mathbf{D}_i(\mathbf{K}_i^{(r-1)}) \mathbf{V}_i^{(r)})^\top (\mathbf{y}_i - \mathbf{D}_i(\mathbf{K}_i^{(r-1)}) \mathbf{V}_i^{(r)}) \right\}} \cdot \frac{P_{\{\mathbf{K}_i^* \geq 0\}}(\mathbf{K}_i^*)}{P_{\{\mathbf{K}_i \geq 0\}}(\mathbf{K}_i)}$
<p>If the move is accepted, set $\mathbf{K}_i^{(r)} = \mathbf{K}_i^*$; otherwise, leave the vector unchanged, $\mathbf{K}_i^{(r)} = \mathbf{K}_i^{(r-1)}$.</p>
<p>Output: An MCMC sample from the joint posterior distribution: $(\mathbf{V}_i^{(r)}, \mathbf{K}_i^{(r)}, \sigma_{i,(r)}^2, \delta_{i,(r)}^2)_{r=1,2,3,\dots}$</p>

A better alternative is to combine differentiation with smooth interpolation to counteract this noise amplification, and Gaussian processes, reviewed in Subheading 2.9, provide a particularly powerful technique. By taking the first derivative of the kernel function, e.g., Eq. (25) or (26), we obtain a prior distribution over functions that define the first temporal derivative, i.e., the concentration gradients, for each of the time points in \mathbf{t} . Provided that the kernel function is differentiable, this is again a valid Gaussian process.

The simplest approach is to compute the expectation over these functions and thus obtain a mean estimate of the analytical solution for the gradients at each time point. For the explicit mathematical expression, see, e.g., eq. (1) in [33]. This acts as a proxy for the missing rates $y_{g,t}$ on the left-hand side of Eq.(1) (*see Note 12*). There are various alternatives to the kernels shown in Eqs.(25) and (26). For instance, if it is known that the signal is periodic, a periodic kernel can be used. We refer the reader to Chapter 4 in [34] for more details and explicit mathematical expressions.

3 Practical Guidance

In this section, we provide a brief overview of how the methods described in the previous section can be applied in practice.

3.1 Gaussian Graphical Models (GGM)

A software implementation of the GGM method developed by Schäfer and Strimmer [7] is implemented in the R package `GeneNet` and available from the CRAN R archive. One can obtain the partial correlation matrices using function `ggm.estimate.pcor` with the `static` method and default parameter settings. From these matrices, only those partial correlations need to be extracted that involve the target gradient response y_g . To obtain the partial correlations for the complete system, including partial correlations for all gradient responses $y_g, \forall g$, the GGM learning algorithm has to be applied repeatedly for each individual gradient response variable y_g . One can then treat the absolute partial correlation values as indicator for the interaction ranks.

3.2 Lasso and Elastic Net

Software implementations of Lasso and the Elastic Net are available from the R package `glmnet`, described by Friedman et al. [35], where the regression parameters are optimized with cyclical coordinate descent. The regularization parameters are typically selected so as to minimize the mean square cross-validation error, using a 10-fold cross-validation scheme. This is done automatically with the function `cv.glmnet()`. Absolute values of nonzero regression coefficients can then be used for ranking molecular interactions.

3.3 Tesla

For the results reported in Subheading 5, Tesla was run with a linear regression implementation in Matlab, where the regression parameters were optimized with convex programming, using the CVX MATLAB package (*see Note 13*). A 10-fold cross-validation scheme was applied to optimize the regularization parameters, minimizing the mean square cross-validation error. Tesla requires the prior specification of permissible change-points. For the application in Subheading 5, these were naturally chosen to reflect the light condition (i.e., light versus darkness).

3.4 Hierarchical Bayesian Regression (HBR)

The marginal posterior probabilities of molecular interactions are directly obtained from the MCMC trajectories, estimated from the relative frequency of inclusion of the corresponding edges in the sampled models. To test for convergence of the MCMC simulations, standard convergence diagnostics, based on correlation scatterplots and Gelman–Rubin potential scale reduction factors [36, 37], can be applied. For the study reported in Subheading 5, satisfactory convergence indications were obtained for 20,000 iterations, with a burn-in period of 10,000 iterations discarded. A software implementation in MATLAB is available from the first author upon request.

3.5 Sparse Bayesian Regression (SBR)

A MATLAB implementation of sparse Bayesian regression (SBR) is available from the supplementary material of [17]. For the study reported in Subheading 5, the default settings for the hyperparameters and the maximal number of iterations for the marginal likelihood maximization were used. We note that the method in [17] is a slightly modified version of the fast marginal likelihood algorithm from [38]; for the technical details, we refer the reader to the supplementary material of [17].

3.6 Bayesian Splines Autoregression (BSA)

For Bayesian splines autoregression, the MATLAB programs provided with the supplementary material of [6] can be used, with the following modification: for the target genes, the future gene expression values have to be replaced by the estimates of the time derivatives, y_g , as discussed at the end of Subheading 2.7. This implementation is particularly straightforward for the gene-specific hyperparameters, corresponding to Eqns. (2.8–2.9) in [6], which were used for the simulations in Subheading 5, as no difference between gene-specific and global hyperparameters was found by Morrissey et al. [6]. For the other model options, including the order of the splines, the number of knots, and the hyperparameters of the Bayesian model, the default settings in the MATLAB programs were used. For the MCMC simulations, standard convergence diagnostics, e.g., based on potential scale reduction factors, should be used, as described in Subheading 3.4.

3.7 State-Space Models (SSM)

In its multivariate formulation, the SSM method described in Subheading 2.8 can neither deal with target-specific potential regulator sets nor with target-specific exclusion of certain data in relation with mutagenesis experiments. However, this can easily be rectified by implementing a separate SSM for each target variable y_g . A Matlab implementation of approximate inference with the variational Bayesian EM algorithm is available from [21]. The outputs depend on the number of hidden nodes. However, Aderhold et al. [39] found that except for very low values, the network reconstruction scores are fairly stable with respect to a modification of the number of hidden nodes. The results reported in Subheading 5 were obtained with the default choice of $n = 8$ hidden nodes.

3.8 Gaussian Process (GP)

The Gaussian process approach described in Subheading 2.9 has been implemented in the GP4GRN software package, developed by Äijö and Lähdesmäki [23]. This software computes, for each target gene, the posterior probabilities of all potential sets of regulators. The posterior probabilities for individual molecular interactions are then obtained by marginalization, summing the posterior probabilities of all configurations of regulators that include the molecular interaction in question, as shown in Eq. (33). The hyperparameters θ, σ^2 in Eq. (33) are optimized with the Polack–Ribiere conjugate gradient method [22] to maximize the marginal likelihood of Eq. (31), while the hyperparameters \mathbf{b}, σ_b^2 are set fixed (Äijö and Lähdesmäki [23] provide default values).

3.9 Mixture Bayesian Network Models (MBN)

For the mixture Bayesian network (MBN) approach, one can apply the implementation of the EM algorithm for Gaussian mixture models from the “Pattern Analysis Toolbox” by I.T. Nabney; this Matlab toolbox has been made available as supplementary material of [40]. As the EM algorithm is a greedy optimization technique that converges to the nearest (local) maximum of the likelihood, it is advisable to repeat the optimization from multiple restarts. The results reported in Subheading 5 were obtained from 10 different initializations (*see Note 14*). This yields $H = 10$ regulator sets $\pi_g^{(1)}, \dots, \pi_g^{(10)}$ for each target variable y_g ($g = 1, \dots, G$). In imitation of the Bayesian approach, one can use the fraction of regulator sets that obtain the regulator x_j^g to rank the regulatory interactions $x_j^g \rightarrow y_g$ ($g = 1, \dots, G$ and $j = 1, \dots, G_g$).

3.10 Gaussian Bayesian Networks (BGe)

For the Gaussian Bayesian network model with the BGe scoring metric, the prior distribution of the unknown parameters is assumed to be a Gaussian–Wishart distribution with hyperparameters α , \mathbf{T}_0 , ν , and $\boldsymbol{\mu}_0$. In the absence of any genuine prior knowledge about the regulatory interactions, a natural choice is to set the parameter matrix of the Wishart prior to the identity matrix, symbolically $\mathbf{T}_0 = \mathbf{I}$, and the mean vector of the Gaussian prior to the zero vector, symbolically: $\boldsymbol{\mu}_0 = \mathbf{0}$ (*see Note 15*). The scalar hyperparameters α and ν can be interpreted as equivalent prior sample sizes [26]. The results reported in Subheading 5 followed [39] and set $\alpha = G_g + 4$ and $\nu = 1$. With this choice, the equivalent prior sample sizes are as uninformative as possible subject to the regulatory conditions discussed by Geiger and Heckerman [26]. To render the computation of the marginal interaction posterior probabilities in Eq. (46) computationally tractable, one can follow [39] and impose a maximal fan-in restriction, e.g., $\mathcal{F} = 3$.

3.11 Improved Chemical Model Averaging (iCheMA)

A pseudo code representation of the iCheMA method is available from Table 2. A software implementation can be found at <http://researchdata.gla.ac.uk/374/>.

4 Evaluation Methodology

4.1 Data

This section describes the data used for a critical comparative assessment of the method performance.

Real data have the advantage that they were obtained from real organisms using real assays. In the application described in Subheading 6, these are transcriptional concentration time courses from *Arabidopsis thaliana* seedlings obtained with quantitative reverse transcription polymerase chain reaction (qRT-PCR). The use of real data mimics the actual application a biologist is interested in. A disadvantage, however, is the absence of a ground truth, making it difficult to evaluate the prediction from the different methods.

Realistic data are simulated from a mathematical model of the molecular interactions occurring in the signaling pathways and regulatory networks. Since the data have been synthetically generated, the ground truth is known and can be used for an objective performance evaluation. A disadvantage is that the data generation process might make simplifying assumptions that render the data insufficiently representative of real biological systems studied in the laboratory. The challenge, hence, is to make the data generation process as realistic as possible, and we describe below how this objective can be accomplished.

4.2 Generation of Realistic Data: Methodology

Various mathematical models have been developed to describe the molecular interactions and signal transduction processes in the central circadian clock of *Arabidopsis thaliana*; see [30, 41, 42] and [43]. They are based on the systems of ordinary differential equations (ODEs) that describe the chemical kinetics of transcription initiation, translation, and posttranslational modification, using mass action kinetics and/or Michaelis–Menten kinetics. In principle, we could use these mathematical models together with the published values of the kinetic rate parameters to generate synthetic transcription profiles from the circadian regulatory networks published by Locke et al. [41] and Pokhilko et al. [30, 42], then use the latter as a gold standard for our method evaluation.

However, this approach would not generate data that are sufficiently biologically realistic. The solutions of ODEs typically converge to limit cycles with regular oscillations and constant amplitude, which fail to capture the stochastic amplitude variation observed in real qRT-PCR experiments. In addition, the damping of oscillations experimentally observed in constant light conditions is not correctly modeled. As discussed in Guerriero et al. [44], the problem of ODEs is that the intrinsic fluctuations of molecular processes in the cell are ignored, thereby not allowing for molecular noise that may have a significant impact on the behavior of the system; see also [45].

A more realistic approach is to model the individual molecular processes of transcription, translation, degradation, dimerization, etc. as individual discrete events, as shown in Table 3. Statistical mechanics arguments then lead to a Markov jump process in continuous time whose instantaneous reaction rates are directly proportional to the number of molecules of each reacting component [45, 46]. Such dynamics can be simulated exactly using standard discrete-event simulation techniques. A convenient procedure is to follow Guerriero et al. [44] and adopt the Bio-PEPA framework [47]. This procedure was used for the benchmark study of Subheading 5 to simulate gene expression profiles for the core circadian clock of *Arabidopsis thaliana*, using the Bio-PEPA Eclipse Plug-in (see Note 16). This framework is built on a stochastic process algebra implementation of chemical kinetics, and the stochastic simulations are run with the Gillespie algorithm [48].

In order to correctly quantify stochastic fluctuations, concentrations are represented as numbers of molecules per unit volume. This requires the unit volume size Ω to be defined, which scales the molecule amounts and kinetic laws such that a unit concentration in an ODE representation becomes a molecule count close to Ω ; see Guerriero et al. [44] for more details. The size of Ω has a strong influence on the stochasticity of the system. Since larger volumes entail a more pronounced averaging effect, the stochasticity decreases with increasing values of Ω , and the solutions from the equivalent deterministic ODEs are subsumed as a limiting case for $\Omega \rightarrow \infty$. Conversely, decreasing values of Ω increase the stochasticity. Guerriero et al. [44] showed that replacing the continuous deterministic dynamics of ODEs by the

Table 3
Illustration of elementary molecular reactions with discrete stochastic kinetics

Elementary molecular reaction	
$X_{\text{DNA}} + X_{\text{protein}} \xrightarrow{k_1} X_{\text{DNA}} + X_{\text{mRNA}} + X_{\text{protein}}$	Transcription
$X_{\text{mRNA}} \xrightarrow{k_2} X_{\text{mRNA}} + X_{\text{protein}}$	Translation
$X_{\text{mRNA}} \xrightarrow{k_3} \emptyset, X_{\text{protein}} \xrightarrow{k_4} \emptyset$	Degradation
$2X_{\text{protein}} \xrightarrow{k_5} X_{\text{dimer}}$	Dimerization

The letter “X” represents a single molecule of the type indicated by the subscript, and the symbol \emptyset indicates the disappearance of a molecule. Arrows indicate reactions, i.e., the transformation of the substrates on the left to the products on the right. The lower case letters above the arrows denote chemical kinetic parameters. The reactions are modeled mathematically with a Markov jump process. Reactions occur stochastically according to a Poisson process, whose intensity is the sum of the kinetic parameters; here: $\lambda = k_1 + \dots + k_5$. The propensity of a reaction is proportional to its kinetic parameter, i.e., given that a reaction has occurred, the probability that the nature of this reaction is of type i is k_i/λ .

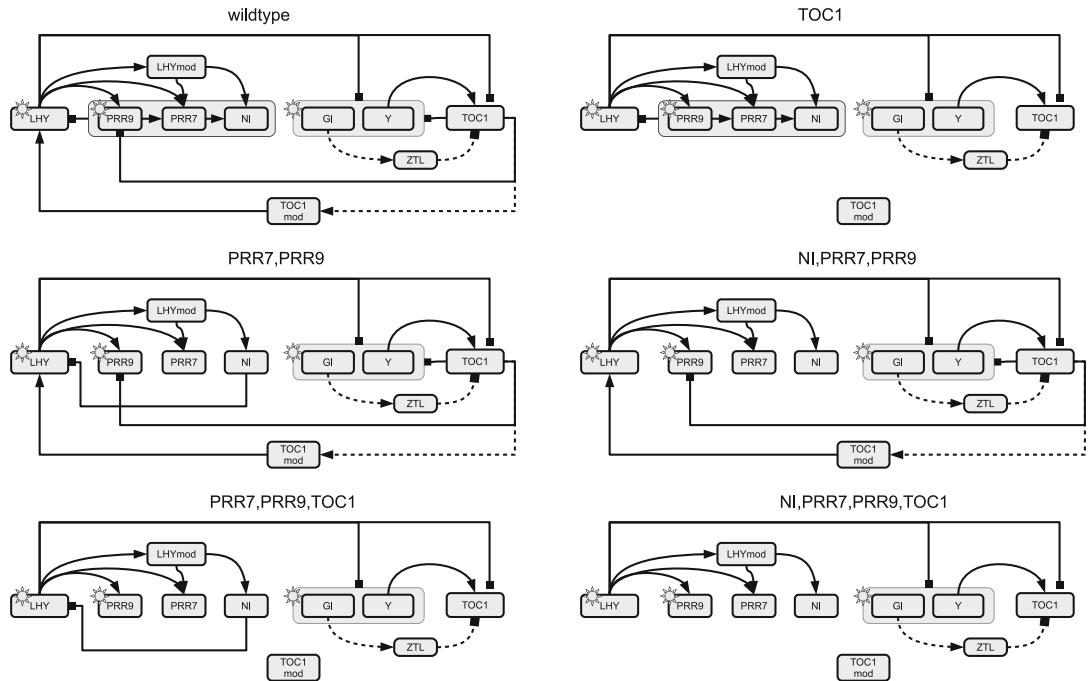


Fig. 3 Model network of the circadian clock in *Arabidopsis thaliana* and network modifications. Each graph shows interactions among core circadian clock genes. Solid lines show protein–gene interactions; dashed lines show protein modifications; and the regulatory influence of light is symbolized by a sun symbol. The top-left panel (“wild type”) shows the network structure published by Pokhilko et al. [29]. The remaining panels show modified network structures, corresponding to constant knockouts of the proteins shown above the corresponding network structure. Gray boxes group sets of regulators or regulated components. Arrows symbolize activations and bars inhibitions

discrete stochastic dynamics with an appropriate choice of Ω leads to a more accurate matching of the experimental data, including the damping of oscillations experimentally observed in constant light, better entrainment to light in several light patterns, better entrainment to changes in photo-period, and the correct modeling of secondary peaks experimentally observed for certain photo-periods.

For the study described in Subheading 5, mRNA concentration time series were simulated from the circadian clock regulatory network published in Guerriero et al. [44] and Pokhilko et al. [29], shown in Fig. 3 (top left, network “wild type”). This involves genetic regulatory reactions for mRNA transcription, protein translation, and mRNA and protein degradation for nine genes. A full list of reactions and their corresponding mathematical descriptions is available from the supplementary material of Guerriero et al. [44].

An additional advantage of this procedure is that it is straightforward to assess the effect of network structure modification on the performance of the network reconstruction methods. This

can easily be effected by inactivating certain reactions in the gold standard network, by setting the respective reaction rates to zero. Figure 3 shows the complete circadian regulatory network in *Arabidopsis thaliana*, as published by Guerriero et al. [44] and Pokhilko et al. [29] (“wild type”), and several modified sparser structures, which are used in the benchmark study described in Subheading 5. The exact setup of the data generation process is described in detail in Subheading 4.3.

4.3 Generation of Realistic Data: Practical Procedure

The Bio-PEPA framework [47] can be used to conveniently generate mRNA and protein concentration profiles with Markov jump processes. As discussed in Subheading 4.2, these profiles are sensitive to the choice of the unit volume parameter Ω . For values of $\Omega < 10$, the concentration profiles are dominated by stochasticity, whereas for $\Omega > 1000$ they become indistinguishable from the deterministic solutions of ODEs. For the study reported in Subheading 5, a value of $\Omega = 100$ was used, as suggested by Guerriero et al. [44], which gives the best match to the experimental qRT-PCR data, in particular with respect to the fluctuations of the qRT-PCR amplitudes. mRNA concentration time series were simulated from the circadian regulatory network of Guerriero et al. [44] and Pokhilko et al. [29], shown in the top-left panel of Fig. 3, named “wild type.” In addition, mRNA concentration time series were simulated from a series of modified network structures, in which various feedback loops and recurrent interactions had been removed (*see Note 17*); these networks are shown in the remaining panels of Fig. 3. For each of these network types, 11 interventions were carried out, in emulation of the biological protocols of Flis et al. [49] and Edwards et al. [50]. These interventions include knock-outs of the genes GI, LHY, TOC1, and the double knock-out PRR7/PRR9. The knock-outs were simulated by downregulating the transcription rates of the targeted genes, and replacing them by random noise, drawn from a truncated normal distribution (to ensure nonnegativity of the concentrations). Again, in emulation of the biological protocols of Flis et al. [49] and Edwards et al. [50], varying photo-periods were simulated, of 4, 6, 8, 12, and 18 h as well as a full dark (DD) and a full light (LL) cycle, each following a 12 h–12 h light–dark cycle entrainment phase over 5 days. For each type of intervention, concentration time series were generated to encompass a simulated epoch of 6 days, of which the first 5 days were used for entrainment. After entrainment, molecule counts of mRNA and proteins were recorded in 2-h intervals of simulated time, for 24 h, giving a total of 13 “observations.” However, for the network reconstruction task, only the gene expression time series were kept and the protein concentrations were discarded; this emulates the common problem of systematically missing values for certain types of molecular species (in this case: protein concentrations).

Combining these 13 observations for each intervention type yields 143 observations in total for each of the regulatory network structures shown in Fig. 3. Finally, to standardize the data, a widely established procedure is to rescale all molecule concentrations to zero mean and unit standard deviation.

4.4 Network Inference Scoring Scheme

All the methods reviewed in Subheading 2 provide network interaction scores, e.g., estimated marginal posterior probabilities or related measures, that allow ranking the network interactions in descending order. If the true regulatory network is known, as in Subheading 4.2 and Fig. 3, this ranking defines the Receiver Operating Characteristic (ROC) curve [51], where for all possible threshold values, the sensitivity (or recall) is plotted against the complementary specificity. By numerical integration, we then obtain the area under the curve (AUROC) as a global measure of network reconstruction accuracy, where larger values indicate a better performance, starting from AUROC = 0.5 to indicate random expectation, to AUROC = 1 for perfect network reconstruction. A second well-established measure that is closely related to the AUROC score is the area under the precision recall curve (AUPREC), which is the area enclosed by the curve defined by the precision plotted against the recall [52]. AUPREC scores have the advantage over AUROC scores in that the influence of large quantities in false positives can be identified better through the precision. These two scores (AUROC and AUPREC) are widely applied in the systems biology community to score the global network reconstructions accuracy [53]. For an illustration, see Fig. 4.

4.5 ANOVA

For the evaluation reported in Subheading 5, hundreds of simulations were run for a variety of different settings, related to the method for derivative (rate) estimation (described in Subheading 2.13), the regulatory network structure (shown in Fig. 3), and the method applied for learning this structure from data (reviewed in Subheading 2). The results are complex and render pattern and trend recognition by visual inspection challenging. In order to disentangle the different factors, and in particular distinguish the effect of the model from the other confounding factors, a powerful method to adopt is the DELVE evaluation procedure for comparative assessment of classification and regression methods in Machine Learning [54, 55], which is based on a multi-way analysis of variance (ANOVA) scheme [e.g., 56].

Let $score_{gnmk}$ denote the AUROC or AUPREC score obtained for gradient estimation method g , network topology n , network reconstruction method m , and data instantiation k . The range of these index parameters is as follows: $g \in \{0, 1\}$, where $g = 0$ denotes the numerical gradient approximation based on the finite difference quotient of Eq. (2), and $g = 1$ the gradient obtained

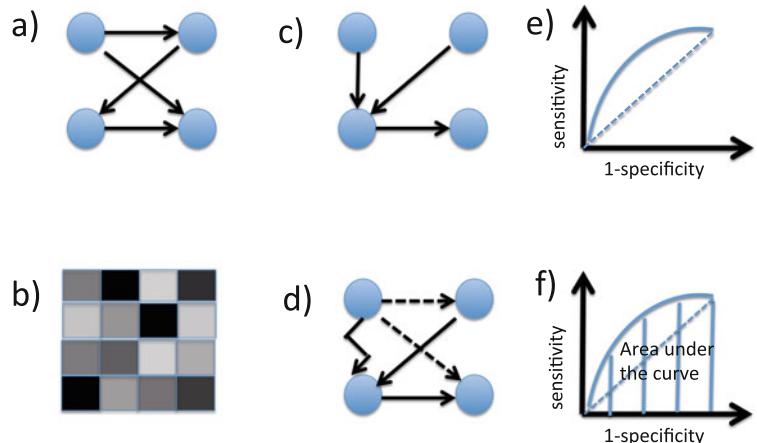


Fig. 4 Illustration of network reconstruction assessment. Panel (a) True network. Panel (b) Matrix of inferred marginal posterior probabilities of the edges, with gray shading indicating values between 0 (white) and 1 (black). Panel (c) Network reconstructed by applying a threshold to the interaction matrix of panel (b) and discarding all edges whose posterior probabilities are below the threshold. Panel (d) Network reconstruction scoring. The figure shows *true positive* edges (solid lines), *false negative* edges (dashed lines), and *false positive* edges (zig-zagging line). From this reconstruction, we can compute various network reconstruction assessment scores. The *sensitivity* or *recall* is the proportion of true edges that are detected. In this example, the sensitivity is $2/4 = 0.5$. The *precision* is the proportion of detected edges that are true. In this example, the precision is $2/3 = 0.\bar{6}$. The *specificity* is the proportion of non-edges that are detected, i.e., for which the erroneous detection of a false edge is avoided. In this example, the specificity is $11/12 = 0.91\bar{6}$. Panel (e) Rather than selecting a specific fixed threshold, we can plot the sensitivity against the complementary specificity (i.e., 1 minus the specificity) for all possible threshold values. This gives us the so-called *receiver operating characteristics (ROC)* curve. The diagonal dashed line is the expected ROC curve for random network reconstruction. Panel (f) To summarize the ROC curve with a single figure of merit, we can numerically integrate over it to obtain the *area under the curve* (AUC or AUROC). This number ranges from 0 to 1, with 1 indicating perfect prediction and 0.5 corresponding to random expectation (the area under the dashed line). A similar procedure, not illustrated in this figure, can be carried out for the *precision-recall* curve, leading to the AUPREC score

from a smooth interpolant; $m \in \{0, 1, 2, 3, 4, 5\}$, where $n = 0$ represents “wild type” (the published network topology), and $n \neq 0$ one of five network modifications shown in Fig. 3; $m \in \{1, 2, \dots, 11\}$, for the 11 network reconstruction methods reviewed in Subheading 2, and $k \in \{1, \dots, K\}$ for K independent data instantiations. We model the AUROC scores with the following ANOVA approach:

$$\text{score}_{gnmk} = G_g + N_n + M_m + \varepsilon_{gnmk} \quad (66)$$

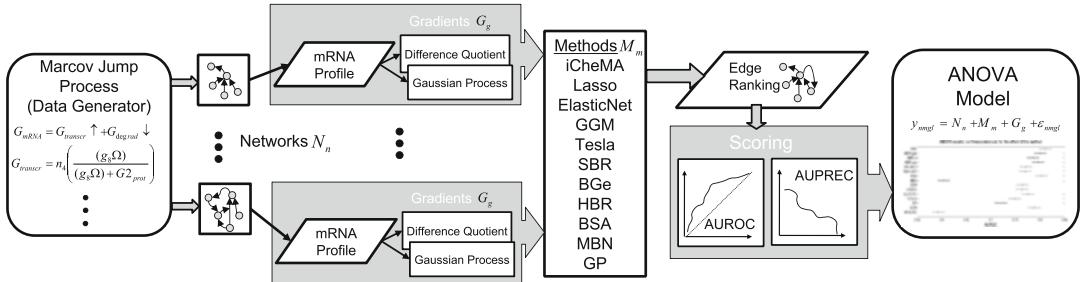


Fig. 5 Overview of the ANOVA pipeline. Based on a mathematical formulation in terms of Markov jump processes, realistic mRNA concentration time series are generated from a collection of gene regulatory networks, and the transcription rate (temporal gradient) is computed with two alternative methods: numerical differentiation versus GP regression. 11 state-of-the-art network reconstruction methods are compared, each predicting a ranking of the potential interactions (edges) in the network. From these rankings, the areas under the ROC (AUROC) and precision-recall curve (AUPREC) are computed, and provide a score for the accuracy of network reconstruction. An ANOVA scheme is applied to disentangle the effects of network topology, gradient computation, and reconstruction method, for clearer recognition of trends and patterns. Figure reproduced from Aderhold et al. [27]

where $\varepsilon_{gnmk} \sim N(0, \sigma^2)$ is zero-mean white additive Gaussian noise, and G_g , N_n , M_m are the main effects associated with the gradient computation, network topology, and network reconstruction method, respectively. A graphical illustration is provided in Fig. 5. As a sanity check, it is recommended to carry out a standard residual analysis, as discussed in more detail in Appendix A.1 of Aderhold et al. [39].

5 Results of the Benchmark Study

This section discusses the results of a comprehensive benchmark study, which is based on the work of Aderhold et al. [27].

5.1 Effect of Transcription Rate Estimation

As we discussed in Subheading 2, there are two ways to estimate the mRNA transcription rate or the gradient. We can either use the finite difference quotient of Eq. (2), or we can use the Gaussian process smoothing approach described in Subheading 2.13. In the latter case, we can compute the gradient analytically, and we therefore refer to these two schemes as the *numerical* and the *analytical* gradient, respectively. To quantify the difference in the accuracy of network inference, both gradient types were applied to all the methods reviewed in Subheading 2, and they were evaluated on the realistic data of Subheading 4.1. The difference quotient was calculated with a time difference of $\delta_t = 2\text{ h}$. This mimics typical time intervals in gene expression time series from recent studies in plant systems biology; see e.g., [49]. The analytical gradient was calculated with a Gaussian process using an RBF kernel, as described in Subheading 2.13. The network reconstruc-

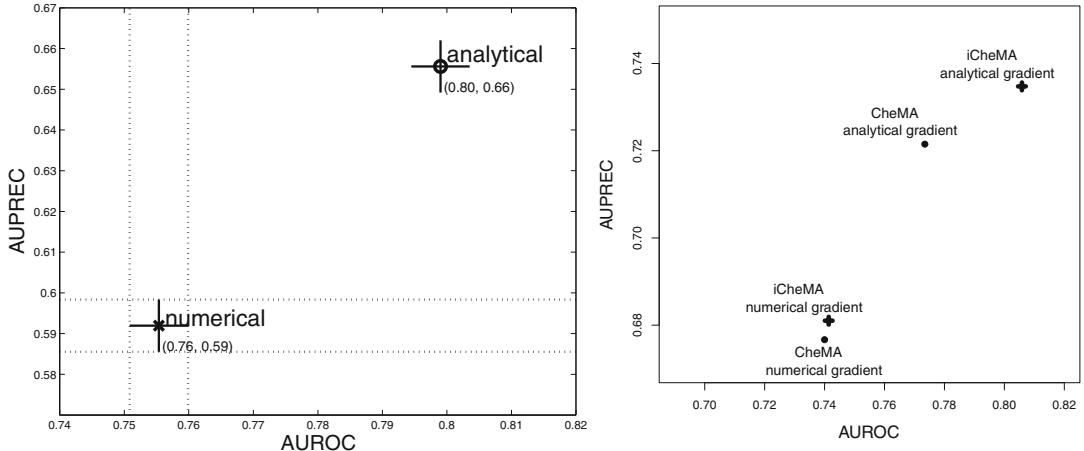


Fig. 6 Left Panel: Effect of the gradient type: numerical *versus* analytical. For the realistic network data from Subheading 4.1, we compared the network reconstruction accuracy of a numerical and an analytical gradient. The numerical gradient was computed with the difference quotient based on a time difference of $\delta_t = 2$ h. The analytical gradient was derived from the derivative of a Gaussian process (GP) using the radial basis function kernel with optimized parameters, as implemented in the Matlab library `gpstuff`. Both panels show the mean AUROC and AUPREC scores with confidence intervals for the effect of the gradient type derived from ANOVA models. All methods listed in Subheading 2 were included, and 10 independent data instantiations were generated from each of the networks shown in Fig. 3. The ANOVA model has thus three effects: gradient type G_g , network structure N_n , and method M_m : $\text{score}_{gnml} = G_g + N_n + M_m + \varepsilon_{gnml}$. Right panel: Comparison between CheMA and iCheMA. The scatterplot shows the network reconstruction accuracy for the iCheMA model, described in Subheading 2.12, compared with the earlier CheMA model proposed by Oates et al. [28]. The comparison is based on realistic data (see Subheading 4.1) generated from the wild-type network in the top-left panel of Fig. 3. Both methods, CheMA and iCheMA, were applied with both a numerical and an analytical gradient. Figures reproduced from Aderhold et al. [27]

tion accuracy was quantified with AUROC and AUPREC scores, as described in Subheading 4.4, for all the different conditions described before: the different network reconstruction methods, M_m , reviewed in Subheading 2; the different network structures, N_n , shown in Fig. 3; and the two gradient types, G_g , summarized above. For each combination, simulations were run on 10 independent data instantiations, generated from the method described in Subheading 4.1. This gives a total of $2 \times 6 \times 11 \times 10 = 1320$ scores, which are decomposed with an ANOVA analysis (see Subheading 4.4) that treats the different conditions and methods as distinct effects: $\text{score}_{gnml} = G_g + N_n + M_m + \varepsilon_{gnml}$. The left panel of Fig. 6 shows the effect of the gradient estimation, G_g . The figure shows that the results for the data with the analytical rate estimation significantly improve over the performance based on the numerical difference quotient. This suggests that the network reconstruction accuracy significantly improves when an analytically derived gradient using a Gaussian process is used instead of the numerical difference quotient.

Method comparison

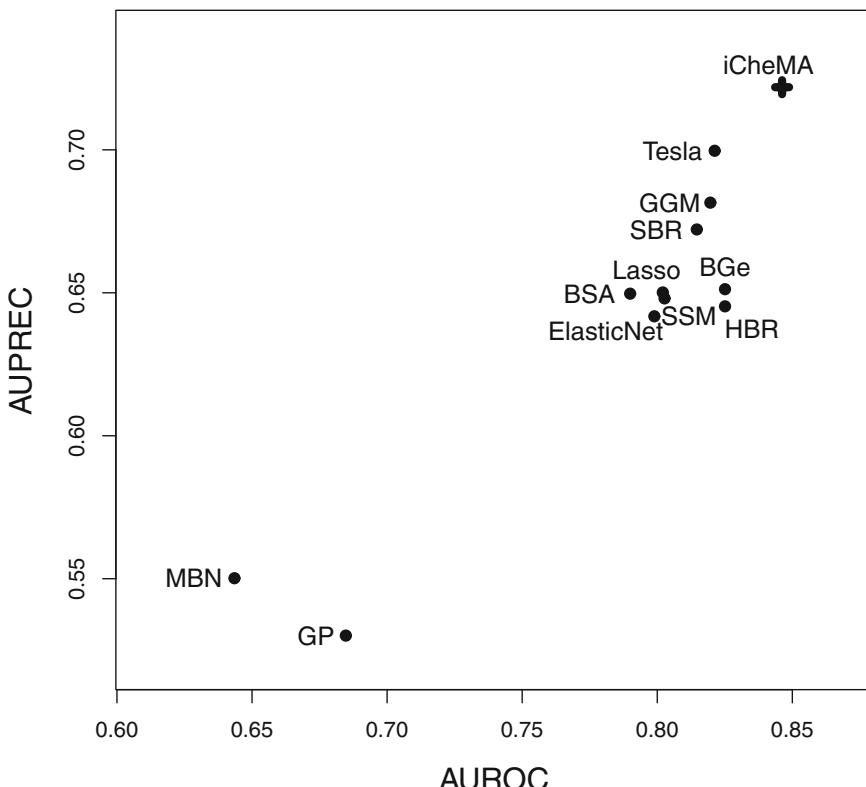


Fig. 7 Model comparison. The scatterplot shows a comparison of the network reconstruction accuracy scores obtained with the methods described in Subheading 2, using the realistic network data from Subheading 4.1, generated from the six network structures of Fig. 3. All network reconstruction methods used the analytical gradient, obtained from GP regression with an RBF kernel, and the displayed AUROC and AUPREC values are the effects of the method, derived from an ANOVA with two effects: **method** M_m and network structure N_n : $y_{mn} = M_m + N_n + \varepsilon_{mn}$. Figures reproduced from Aderhold et al. [27]

5.2 Network Reconstruction Accuracy

Following the same procedure as described in the previous section, but this time extracting the effect of the network reconstruction method, M_m , from the ANOVA analysis, leads to Fig. 7. The figure shows a comparison of the performance of all the methods reviewed in Subheading 2, separated into AUROC (horizontal axis) and AUPREC (vertical axis) scores. The first feature to notice is that all methods perform better than random expectation ($\text{AUROC} > 0.5$), but fall short of perfect prediction ($\text{AUROC} < 1.0$). The first finding is encouraging, and the latter finding reflects the fact that all reconstruction methods are based on a mathematical abstraction and simplification that inevitably misses some of the detailed biological effects.

The majority of the methods are grouped around an AUROC score of 0.82 and an AUPREC score of 0.65. Some methods

turn out to consistently outperform others. For instance, sparse Bayesian regression (SBR) is consistently outperformed by Gaussian graphical models (GGM), and outperforms itself Lasso, Elastic Nets, state-space models (SSM), and Bayesian spline autoregression (BSA). However, the differences are rather small. For some other methods, the comparison is less clear. For instance, hierarchical Bayesian regression (HBR) has lower AUPREC scores than GGMs and SBR, but higher AUROC scores. Hence, from this analysis it is not clear which method is to be preferred. This ambiguity is a consequence of the fact that both scores, AUROC and AUPREC, summarize the rich information contained in the ROC and precision-recall curves with a single figure, which inevitably incurs a loss of information. Hence, for methods where the ranking is insufficiently clear, an inspection of the entire curves would be advisable.

A striking feature is the strong performance of Tesla. This method takes change-point information, related to the light–dark cycle of the plant’s environment, into account. It is therefore closer to the biology than most other methods, which is reflected in its better performance.

The clear winner of the comparative evaluation is iCheMA, which consistently outperforms all other methods in terms of both AUROC and AUPREC scores. The explanation for the better performance is given by a comparison of Eqs. (1) and (47). All other methods are based on Eq. (1), which models the effect of transcriptional regulation by a generic function that has to be learned from the data. In contrast, iCheMA is the only method that is based on Eq. (47), where transcriptional regulation is assumed to follow Michaelis–Menten kinetics. In this way, the mathematical model of iCheMA is closer to the true biological processes than the competing models, which is rewarded with higher network reconstruction performance scores. To paraphrase this, while the competing models have to learn the functional form of Michaelis–Menten kinetics from the data from scratch, iCheMA uses the functional form of Michaelis–Menten kinetics as prior information, which is the source of the performance improvement.

Two methods were found to have significantly poorer performance than most of the other methods: the Gaussian process approach of Äijö and Lähdesmäki [23] (GP), and the Mixture Bayesian Network (MBN) model. The poor performance of the latter reflects the fact that learning conditional probabilities indirectly via joint probabilities is usually inferior to learning conditional probabilities directly [57]. The interesting observation regarding GP is that this method is similar to iCheMA, but without using the Michaelis–Menten kinetic term in the modeling of transcriptional regulation. This finding provides a stark example of the importance of incorporating biological prior knowledge into the model whenever it is available.

5.3 Comparison of CheMA and iCheMA

Having identified iCheMA as the most powerful of the methods reviewed in this chapter, the final question is how it compares with its predecessor, CheMA, proposed by Oates et al. [28]. CheMA estimates transcription rates via a numerical gradient, while iCheMA uses an analytical gradient, and we have already seen in Subheading 5.1 that the latter is the more powerful approach. However, there are two additional subtler differences between the two methods. As opposed to CheMA, iCheMA respects the positivity constraint of the reaction rates, Eq. (57), which prevents the analytical marginalization in Eq. (54). Hence, iCheMA replaces the approximate collapsed Gibbs sampler of CheMA by an exact uncollapsed Gibbs sampler. The second difference concerns the prior distribution, where iCheMA uses a proper conjugate prior, while CheMA uses an improper Jeffreys prior. We refer the reader to [27] for details.

The right panel of Fig. 6 shows a performance comparison. Due to the computational costs, the comparison was only carried out for transcription time series generated from the wild-type network, shown in the top-left panel of Fig. 3. To disentangle the effects of gradient computation and the other two factors, CheMA and iCheMA, were run with both the numerical and the analytical gradient. In confirmation of the findings in Subheading 5.1, the analytical gradient clearly achieves better results than the numerical gradient, but it is seen that iCheMA outperforms CheMA also due to the other two factors.

As an aside, we notice that the performance of iCheMA in Fig. 6, right panel, is slightly poorer than in Fig. 7. This is due to the fact that in the former case, only data from the wild-type network were considered, whereas the latter figure takes all networks into account. Note that all network variants in Fig. 3 are based on pruning and discarding interactions of the wild-type network. This finding thus confirms the well-known fact that densely connected networks are more difficult to learn than sparser ones, and the differences in the scores allow a quantification of this effect.

6 Real-World Application

There is substantial interest in understanding the molecular mechanism of circadian regulation, i.e., an organism's internal time keeping. For plants in particular, circadian regulation is essential to align the plant's metabolism to the diurnal rhythm of day and night. We finish this chapter with an application of the best method from the benchmark study (the iCheMA method) to the prediction of the circadian regulatory network in *Arabidopsis thaliana*. The data used for this study come from the DIURNAL database of the Mockler Lab [58]. The focus of the

DIURNAL database are diurnal and circadian regulated genes that have been previously identified from microarray data using the Affymetrix ATH1 GeneChip platform (TAIR) in conjunction with the HAYSTACK tool [58]. Each selected gene in DIURNAL provides a compilation of up to 20 different time courses determined by different experimental conditions, which were produced by different laboratories. Each time series has gene expression samples from 12 consecutive measurements taken in 4-h intervals that cover 48 h. The experimental conditions include different settings such as 12-h light days, short days with 6 h of light, long days with 18 h of light, total darkness, constant light, different temperatures, and over-expressed genes. The background strain is predominantly Col-0 (see Note 18). Gene expression profiles were extracted for the following *A. thaliana* genes: *CCA1*, *LHY*, *PRR5*, *PRR7*, *PRR9*, *GI*, *TOC1*, *LUX*, *ELF3*, *ELF4*, *RVE8-1*, and *RVE8-2*, where the last two correspond to two different Affymetrix probes of *RVE8*. Since all 20 conditions were available for these genes, measurements for $T = 240$ time points were collected for each gene. Data were normalized for each laboratory separately, since some of the time courses were found to be on a different scale. Furthermore, an artificial light variable was added to mimic a light spike at the beginning of the day. There are some subtleties related to data normalization, artificial light conditions, and transcriptional time delays, which are not relevant for gaining a conceptual methodological understanding and which we therefore do not discuss here. The interested reader is referred to the appendix of Aderhold et al. [27] for more details.

6.1 Results

Various hypotheses about the structure of the central circadian gene regulatory network in *A. thaliana* have been published in the biological literature. For the evaluation of the network inference from the gene expression time series, two recently published network structures were used, which we refer to as the P2013 [42] and the F2014 network [59]. Taking these structures as a gold standard, we can evaluate the network reconstruction accuracy with standard ROC curves. The results are shown in Fig. 8. The figure compares the performance of two different Gaussian process kernels: the RBF kernel and the periodic kernel (see chapter 4 in [34]). A further subtlety is the fact that the F2014 network includes a gene that is not included in the P2013 network (*RVE8*). For that reason, the reconstruction accuracy for the F2014 network was repeated twice: with *RVE8* included and excluded. Figure 8 also shows the area under the ROC curves (AUROC) as well as the area under the precision-recall curve (AUPREC) as an alternative figure of merit.

With AUROC values ranging between 0.58 and 0.77, the agreement with the literature is significantly better than random expectation. The results obtained with the periodic kernel for the

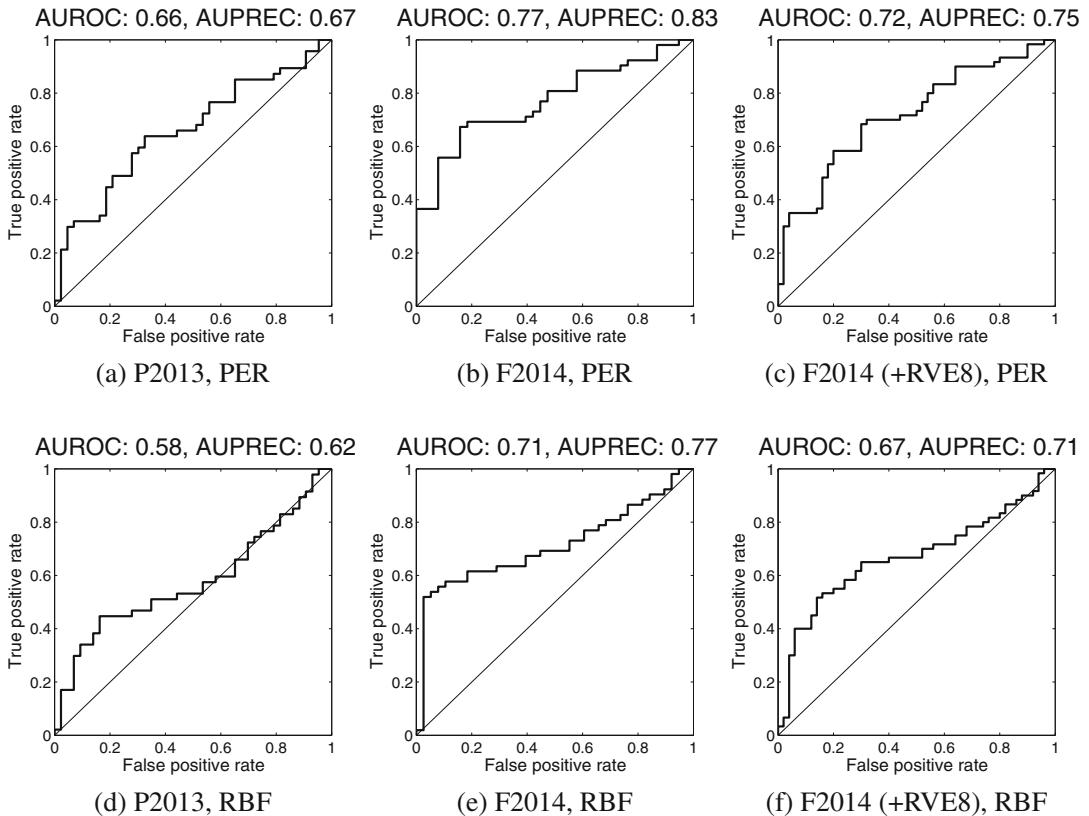


Fig. 8 ROC curves comparing iCheMA with the F2014 and P2013 networks for the gene expression data from *A. thaliana*. Panels (a–c) show ROC curves based on iCheMA with a gradient derived from a periodic (PER) kernel, panels (d–f) show ROC curves based on a gradient derived from an RBF kernel. The numbers at the top of each panel show the areas under the ROC curves (AUROC) and precision-recall curves (AUPREC). P2013 and F2014 refer to two different gold-standard networks; see the main text for details. Panels (c) and (f) display the comparison with the F2014 network and the additional RVE8 gene included, which is not present in the P2013 network. Figure adapted from Aderhold et al. [27]

Gaussian process are systematically better than those obtained with the RBF kernel. This is not surprising, given that the diurnal light-dark cycle and, consequently, the gene expression time series are intrinsically of a periodic nature. An interesting finding is that the agreement with the F2014 network is consistently better than with the P2013 network.

To arrive at a particular network structure prediction, the marginal posterior probabilities obtained with the periodic kernel were used, corresponding to the top row in Fig. 8, and all values above a selection threshold of 0.11 were interpreted as evidence of a gene interaction; this value was found in Grzegorczyk et al. [60] to lead to approximately the same number of gene interactions as in the P2013 network. The resulting network is shown in Fig. 9. To formally compare this prediction with the two networks from the

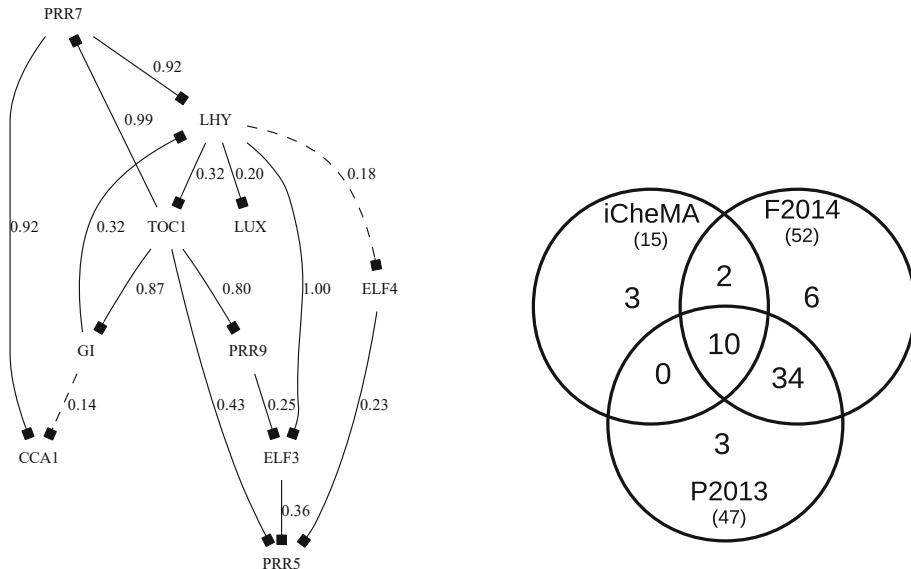


Fig. 9 Predicted clock gene interactions for *A. thaliana*. The *left panel* shows the clock gene network in *A. thaliana*, as predicted with iCheMA from the gene expression data described in Subheading 6. The numbers on the edges denote predicted marginal posterior probabilities as defined in Subheading 2.12.2. Following Grzegorczyk et al. [60], the edge inclusion threshold was set to 0.11. The gradients were derived from a Gaussian process with periodic kernel. We only considered repressive interaction types, symbolized by a small box at the end of a directed edge, as this is the most likely scenario for gene interactions inside the circadian clock. The *right panel* displays a Venn diagram with the number of matching interactions among the predicted network, the P2013 [42], and F2014 network [59]. The numbers in brackets correspond to the total amount of edges in each network. Figure reproduced from Aderhold et al. [27]

literature, we created a Venn diagram, also shown in Fig. 9, which displays the number of interactions in the various intersections, e.g., the number of interactions predicted with iCheMA and found in P2013, but not in F2014, etc. Of the 15 gene interactions predicted with iCheMA in the way described above, shown in Fig. 9, 10 can be found in both the P2013 and the F2014 network, 2 can be found in the F2014 network, and only 3 do not agree with any of the published networks. This result is significantly better than random expectation and suggests that iCheMA provides a powerful tool for hypothesis generation in molecular systems biology.

We note that several interactions of the networks from the literature fall below our selection threshold and are not included in the reconstructed network. On a closer inspection of the recent relevant literature in this field, we observe a clear trend towards ever more complex and densely connected networks: Locke et al. [61], Pokhilko et al. [29, 30, 42], and Fogelmark and Troein [59]. This is “justified” by improving goodness of fit. However, goodness of fit is not an appropriate model selection criteria,

as for nested models, the root mean square prediction error is monotonically decreasing with the number of parameters. A good fit to the data could be the result of overfitting, and mathematically sound model selection criteria, as reviewed in this chapter, are indispensable for valid model development. The mismatch between the data-driven network reconstruction discussed in this chapter and the models from the recent literature does not necessarily indicate that the former approach is overconservative, but could also indicate overfitting and spurious complexity of the latter. This is an active area of investigation in contemporary plant systems biology, and the answer to this question will undoubtedly come out of future research projects.

7 Summary

We have reviewed, in Subheading 2, 11 representative state-of-the-art methods for regulatory network reconstruction from postgenomic data, with a practical usage guidance in Subheading 3. We have described, in Subheading 4, a workflow for systematic model assessment, which was applied, in Subheading 5, to realistic data generated from a wild-type network of circadian regulation and a series of five gene knock-out modifications. The results allow a comparative assessment of the network reconstruction accuracy and its quantification in terms of AUROC and AUPREC scores. It has turned out that modeling the biological regulation processes more closely has an edge on competing approaches, and iCheMA has come out as a clear winner that systematically outperforms all other methods. The comparatively high computational costs restrict this method to small-to-medium size rather than genome-wide networks, though. The application of iCheMA to gene expression time series from a set of core circadian clock genes in *Arabidopsis thaliana* has revealed that reconstructed gene interactions with high posterior probabilities tend to be in very good agreement with the literature. However, several hypothesized gene interactions published in the recent plant systems biology literature have been found to have low posterior probabilities, which might indicate spurious complexity in recently published circadian clock gene regulatory networks and room for improvement in this very active field of research.

8 Notes

1. Note that the sets of potential regulators are defined for each gene \mathcal{g} specifically. That is, the potential regulators for two target variables $y_{\mathcal{g}}$ and $y_{\mathcal{g}'}$ can be different, e.g., if certain (biologically motivated) restrictions are imposed.

2. For consistency with the fundamental equation of transcription, Eq. (1), we will enforce that each regulator set π_g for y_g contains the concentration x_g of g , symbolically $x_g \in \pi_g$.
3. Note that vector $\mathbf{x}_{\cdot,t}$ includes every available regulator without any dependency on the target gene g .
4. Note that the repeated bipartitioning of the genes into targets and putative regulators renders Glasso equivalent to Lasso, as discussed on page 4 of [8]. Lasso will be discussed in the next subsection.
5. For consistency with scaling laws, the intercept should actually be excluded from the regularization term. A more detailed discussion is beyond the scope of this chapter; see Hastie et al. [11] for more details.
6. We set: $v = 0.005$, $A_\delta = 2$, and $B_\delta = 0.2$, as in Grzegorczyk and Husmeier [14].
7. We use the authors' terminology, although the model is not a proper Bayesian network.
8. More precisely, $\mu_{g,b}^*$ is obtained by deleting the element corresponding to the target variable $y_{g,t}$ in $\mu_{g,b}$, and $\Sigma_{g,b}^*$ is obtained by deleting the row and the column corresponding to $y_{g,t}$ in $\Sigma_{g,b}$.
9. In our applications, we had $n \approx 10$ potential regulators.
10. Note that the abbreviation “BGe” was introduced by Geiger and Heckerman [26] and stands for *Bayesian metric for Gaussian networks having score equivalence*; see [26] for more details.
11. Recall from Table 1 that in the notation $x_{i,t}$ the first index, i , represents the gene, and the second index, t , time.
12. Using the whole distribution, as on page 58 of [33], gives us additional indication of uncertainty akin to a distribution of measurement errors, albeit at increased computational costs (additional matrix operations).
13. Matlab software for *Disciplined Convex Programming*: <http://cvxr.com/cvx/>.
14. The results were obtained with the procedure described in [39], where the initializations were obtained from the k-means cluster algorithm. The initial \mathcal{K}_g centers of the k-means algorithms are sampled from a multivariate Gaussian $N(\boldsymbol{\mu}, \mathbf{I})$ distribution, where \mathbf{I} is the identity matrix and $\boldsymbol{\mu}$ is a random expectation vector with entries sampled independently from continuous uniform distributions on the interval $[-1, +1]$. To avoid that the EM algorithm is initialized with allocations that possess unoccupied (empty) mixture components, one resamples the initial centers and reruns the k-means algorithm whenever k-means leads to outputs with empty components.
15. Loosely speaking, this setting ($\boldsymbol{\mu}_0 = \mathbf{0}$ and $\mathbf{T}_0 = \mathbf{I}$) reflects our “prior belief” that all domain variables, i.e., the potential

regulators and the target variable, are i.i.d. standard normally distributed.

16. <http://www.bioppepa.org>.
17. The translation of those proteins contributing to interactions that one likes to suppress is turned off.
18. A full list of growing conditions and strains is available from the Mockler Lab [58] at ftp://www.mocklerlab.org/diurnal/expression_data/Arabidopsis_thaliana_conditions.xlsx.

References

1. Ptashne M, Gann A (2001) Genes and signals. Cold Spring Harbor Laboratory Press, Cold Spring Harbor
2. Barenco M, Tomescu D, Brewer D, Callard R, Stark J, Hubank M (2006) Ranked prediction of p53 targets using hidden variable dynamic modeling. *Genome Biol* 7(3):R25
3. Lawrence ND, Girolami M, Rattray M, Sanguinetti G (2010) Learning and inference in computational systems biology. MIT Press, Cambridge
4. Husmeier D (2003) Sensitivity and specificity of inferring genetic regulatory interactions from microarray experiments with dynamic Bayesian networks. *Bioinformatics* 19:2271–2282
5. Zoppoli P, Morganella S, Ceccarelli M (2010) TimeDelay-ARACNE: reverse engineering of gene networks from time-course data by an information theoretic approach. *BMC Bioinf* 11:154
6. Morrissey ER, Juárez MA, Denby KJ, Burroughs NJ (2011) Inferring the time-invariant topology of a nonlinear sparse gene regulatory network using fully Bayesian spline autoregression. *Biostatistics* 12(4):682–694
7. Schäfer J, Strimmer K (2005) A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics. *Stat Appl Genomics Mol Biol* 4(1). <https://doi.org/10.2202/1544-6115.1175>
8. Friedman J, Hastie T, Tibshirani R (2008) Sparse inverse covariance estimation with the graphical Lasso. *Biostatistics* 9:432–441
9. Opgen-Rhein R, Strimmer K (2007) From correlation to causation networks: a simple approximate learning algorithm and its application to high-dimensional plant gene expression data. *BMC Syst Biol* 1(37). <https://doi.org/10.1186/1752-0509-1-37>
10. Tibshirani R (1995) Regression shrinkage and selection via the Lasso. *J R Stat Soc Ser B (Methodol)* 58(1):267–288
11. Hastie T, Tibshirani R, Friedman JJH (2009) The elements of statistical learning. Springer, New York
12. Zou H, Hastie T (2005) Regularization and variable selection via the Elastic Net. *J R Stat Soc Ser B (Stat Methodol)* 67(2):301–320
13. Ahmed A, Xing EP (2009) Recovering time-varying networks of dependencies in social and biological studies. *Proc Natl Acad Sci* 106:11878–11883
14. Grzegorczyk M, Husmeier D (2012) A non-homogeneous dynamic Bayesian network with sequentially coupled interaction parameters for applications in systems and synthetic biology. *Stat Appl Genet Mol Biol* 11(4). Article 7
15. Bishop CM (2006) Pattern recognition and machine learning. Springer, Singapore
16. Tipping M (2001) Spare Bayesian learning and the relevance vector machine. *J Mach Learn Res* 1:211–244
17. Rogers S, Girolami M (2005) A Bayesian regression approach to the inference of regulatory networks from gene expression data. *Bioinformatics* 21(14):3131–3137
18. Murphy KP (2012) Machine learning: a probabilistic perspective. MIT Press, Cambridge
19. Smith M, Kohn R (1996) Nonparametric regression using Bayesian variable selection. *J Econom* 75:317–343
20. Beal M, Falciani F, Ghahramani Z, Rangel C, Wild D (2005) A Bayesian approach to reconstructing genetic regulatory networks with hidden factors. *Bioinformatics* 21(3):349–356
21. Beal M (2003) Variational algorithms for approximate Bayesian inference. PhD thesis, Gatsby Computational Neuroscience Unit, University College London, London
22. Rasmussen C, Williams C (2006) Gaussian processes for machine learning, vol 1. MIT Press, Cambridge

23. Äijö T, Lähdesmäki H (2009) Learning gene regulatory networks from gene expression measurements using non-parametric molecular kinetics. *Bioinformatics* 25(22):2937–2944
24. Ko Y, Zhai C, Rodriguez-Zas S (2007) Inference of gene pathways using Gaussian mixture models. In: International conference on bioinformatics and biomedicine, Fremont, pp 362–367
25. Ko Y, Zhai C, Rodriguez-Zas S (2009) Inference of gene pathways using mixture Bayesian networks. *BMC Syst Biol* 3:54
26. Geiger D, Heckerman D (1994) Learning Gaussian networks. In: International conference on uncertainty in artificial intelligence. Morgan Kaufmann Publishers, San Francisco, pp 235–243
27. Aderhold A, Husmeier D, Grzegorczyk M (2017) Approximate Bayesian inference in semi-mechanistic models. *Stat Comput* 27(4):1003–1040
28. Oates CJ, Dondelinger F, Bayani N, Korkola J, Gray JW, Mukherjee S (2014) Causal network inference using biochemical kinetics. *Bioinformatics* 30(17):i468–i474
29. Pokhilko A, Hodge S, Stratford K, Knox K, Edwards K, Thomson A, Mizuno T, Millar A (2010) Data assimilation constrains new connections and components in a complex, eukaryotic circadian clock model. *Mol Syst Biol* 6(1):416
30. Pokhilko A, Fernández A, Edwards K, Southern M, Halliday K, Millar A (2012) The clock gene circuit in *Arabidopsis* includes a repressor with additional feedback loops. *Mol Syst Biol* 8:574
31. Marin JM, Robert CP (2007) Bayesian core: a practical approach to computational Bayesian statistics. Springer, New York
32. Chib S, Jeliazkov I (2001) Marginal likelihood from the Metropolis–Hastings output. *J Am Stat Assoc* 96(453):270–281
33. Holsclaw T, Sansó B, Lee HK, Heitmann K, Habib S, Higdon D, Alam U (2013) Gaussian process modeling of derivative curves. *Technometrics* 55(1):57–67
34. Rasmussen CE, Williams CKI (2006) Gaussian processes for machine learning. MIT Press, Cambridge
35. Friedman J, Hastie T, Tibshirani R (2010) Regularization paths for generalized linear models via coordinate descent. *J Stat Softw* 33(1):1–22
36. Brooks S, Gelman A (1999) General methods for monitoring convergence of iterative simulations. *J Comput Graph Stat* 7:434–455
37. Gelman A, Rubin D (1992) Inference from iterative simulation using multiple sequences. *Stat Sci* 7:457–472
38. Tipping M, Faul A, et al (2003) Fast marginal likelihood maximisation for sparse Bayesian models. In: International workshop on artificial intelligence and statistics, vol 1, pp 3–6
39. Aderhold A, Husmeier D, Grzegorczyk M (2014) Statistical inference of regulatory networks for circadian regulation. *Stat Appl Genet Mol Biol* 13(3):227–273
40. Nabney I (2002) NETLAB: algorithms for pattern recognition. Springer, Berlin
41. Locke JCW, Kozma-Bognár L, Gould PD, Fehér B, Kevei E, Nagy F, Turner MS, Hall A, Millar AJ (2006) Experimental validation of a predicted feedback loop in the multi-oscillator clock of *Arabidopsis thaliana*. *Mol Syst Biol* 2(59). <https://doi.org/10.1038/msb4100102>
42. Pokhilko A, Mas P, Millar AJ, et al (2013) Modelling the widespread effects of TOC1 signalling on the plant circadian clock and its outputs. *BMC Syst Biol* 7(1):1–12
43. Trejo-Banos D, Millar AJ, Sanguinetti G (2015) A Bayesian approach for structure learning in oscillating regulatory networks. *Bioinformatics* 31:3617–3624
44. Guerriero M, Pokhilko A, Fernández A, Halliday K, Millar A, Hillston J (2012) Stochastic properties of the plant circadian clock. *J R Soc Interface* 9(69):744–756
45. Wilkinson DJ (2009) Stochastic modelling for quantitative description of heterogeneous biological systems. *Nat Rev Genet* 10(2):122–133
46. Wilkinson D (2011) Stochastic modelling for systems biology, vol 44. CRC Press, Boca Raton
47. Ciocchetta F, Hillston J (2009) Bio-PEPA: a framework for the modelling and analysis of biological systems. *Theor Comput Sci* 410(33):3065–3084
48. Gillespie D (1977) Exact stochastic simulation of coupled chemical reactions. *J Phys Chem* 81(25):2340–2361
49. Flis A, Fernández AP, Zielinski T, Mengin V, Sulpice R, Stratford K, Hume A, Pokhilko A, Southern MM, Seaton DD, McWatters HG, Stitt M, Halliday KJ, Millar AJ (2015) Defining the robust behaviour of the plant clock

- gene circuit with absolute RNA timeseries and open infrastructure. *Open Biol* 5(10):150042. <https://doi.org/10.1098/rsob.150042>
- 50. Edwards K, Akman O, Knox K, Lumsden P, Thomson A, Brown P, Pokhilko A, Kozma-Bognar L, Nagy F, Rand D, et al (2010) Quantitative analysis of regulatory flexibility under changing environmental conditions. *Mol Syst Biol* 6(1):424
 - 51. Hanley JA, McNeil BJ (1982) The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology* 143(1):29–36
 - 52. Davis J, Goadrich M (2006) The relationship between precision-recall and ROC curves. In: Proceedings of the 23rd international conference on machine learning (ICML). ACM, New York, pp 233–240
 - 53. Marbach D, Costello JC, Küffner R, Vega NM, Prill RJ, Camacho DM, Allison KR, Kellis M, Collins JJ, Stolovitzky G, et al (2012) Wisdom of crowds for robust gene network inference. *Nat Methods* 9(8): 796–804
 - 54. Rasmussen CE (1996) Evaluation of Gaussian processes and other methods for non-linear regression. PhD thesis, Citeseer
 - 55. Rasmussen CE, Neal RM, Hinton GE, van Camp D, Revow M, Ghahramani Z, Kustra R, Tibshirani R (1996) The DELVE repository was developed as part of a PhD thesis, which could be cited as an alternative to the technical report: Carl Edward Rasmussen Evaluation of Gaussian Processes and other Methods for Non-Linear Regression PhD thesis University of Toronto
 - 56. Brandt S (1999) Data analysis: statistical and computational methods for scientists and engineers. Springer, New York
 - 57. Neuneier R, Hergert F, Finnoff W, Ormoneit D (1994) Estimation of conditional densities: a comparison of neural network approaches. In: International conference on artificial neural networks. Springer, Berlin, pp 689–692
 - 58. Mockler T, Michael T, Priest H, Shen R, Sullivan C, Givan S, McEntee C, Kay S, Chory J (2007) The DIURNAL project: DIURNAL and circadian expression profiling, model-based pattern matching, and promoter analysis. In: Cold Spring Harbor symposia on quantitative biology, vol 72. Cold Spring Harbor Laboratory Press, Cold Spring Harbor, pp 353–363
 - 59. Fogelmark K, Troein C (2014) Rethinking transcriptional activation in the *Arabidopsis* circadian clock. *PLoS Comput Biol* 10(7):e1003705
 - 60. Grzegorczyk M, Aderhold A, Husmeier D (2015) Inferring bi-directional interactions between circadian clock genes and metabolism with model ensembles. *Stat Appl Genet Mol Biol* 14(2):143–167
 - 61. Locke JCW, Southern MM, Kozma-Bognár L, Hibberd V, Brown PE, Turner MS, Millar AJ (2005) Extension of a genetic network model by iterative experimentation and mathematical analysis. *Mol Syst Biol* 1(1)



Chapter 4

Whole-Transcriptome Causal Network Inference with Genomic and Transcriptomic Data

Lingfei Wang and Tom Michoel

Abstract

Reconstruction of causal gene networks can distinguish regulators from targets and reduce false positives by integrating genetic variations. Its recent developments in speed and accuracy have enabled whole-transcriptome causal network inference on a personal computer. Here, we demonstrate this technique with program Findr on 3000 genes from the Geuvadis dataset. Subsequent analysis reveals major hub genes in the reconstructed network.

Key words Causal gene network, Whole-transcriptome network, Causal inference, Genome–transcriptome variation

1 Introduction

Rapid developments in sequencing technologies have driven low the cost and high the throughput [1], with genomic and transcriptomic datasets from the same individuals increasingly publicly available (e.g., [2, 3]). The question now lies at the computational aspect, on how to fully exploit those datasets in order to address important biological and medical questions [4]. Network-based approaches have received strong interests, especially from the clinical domain, where disease-related hub genes present attractive candidates for drug targeting [5, 6].

In this chapter, we focus on the reconstruction of causal gene networks on genome and (whole-)transcriptome datasets (*see* [7] for a review, and also Chapter 5 for an alternative perspective). As opposed to co-expression networks, causal gene networks are directed, and can identify the regulator among two co-expressed genes, or the existence of a hidden confounder gene or a feedback loop. With more stringent statistical tests, causal inference can also reduce the notoriously high numbers of false positives in co-expression networks.

For this purpose, genomic variations, which are typically observed in cohort studies and recombinant inbred lines, can be integrated as causal anchors or instrumental variables. Similar to double-blinded randomized controlled trials, genomic variations define naturally randomized groupings of individuals that allow to infer the causal relations between quantitative traits, a principle also known as Mendelian randomization [8]. To test for a causal relation from a candidate regulator to a target, Mendelian randomization seeks a shared upstream causal anchor that is associated to both. By assuming that the causal anchor can only affect the target through the regulator, the interaction would then be identified.

However, this assumption does not always hold for gene regulations. For example, even if genomic variations are limited to lie in the *cis*-regulatory region of the regulator (i.e., *cis*-expression quantitative trait loci; *cis*-eQTLs), they may still also be associated to other nearby genes, which in turn control the target (Fig. 1a). Consequently, existing studies and public software tools, namely Trigger [9] and CIT [10], were proposed to test this assumption through a “conditional independence test.” As was revealed in other studies, the conditional independence test cannot consider the existences of hidden confounders (Fig. 1b) and technical variations (Fig. 1c, [7, 11–15]), which led to few discoveries of gene regulations. Additionally, neither software was efficient enough to handle the scale of modern datasets.

Recently in [14], we proposed alternative tests which are robust against confounders and technical variations. Implementational and statistical advances in the accompanying program Findr also resulted in almost 1,000,000 times of speedup compared to CIT. This makes possible the reconstruction of whole-transcriptome causal gene networks, which can detect novel interactions by avoiding any preselection of genes.

In this chapter, we present a detailed protocol for the application of Findr, through an example where causal gene networks are inferred among 3000 genes from downsampled Geuvadis study data [2]. This is supplemented with a brief outline of the methods implemented in Findr, and its future perspectives in method development and application domains.

2 Notations and Materials

In this section, we briefly formalize the network inference question and establish the necessary computational environment for Findr.

2.1 Question Formalization

Consider genome–transcriptome variation data from N unrelated individuals. After preprocessing and eQTL analysis, we have identified G expressed genes (see Note 1), $\mathcal{J}_1, \mathcal{J}_2, \dots, \mathcal{J}_G$, in which the first $E \leq G$ genes (i.e., $\mathcal{J}_1, \mathcal{J}_2, \dots, \mathcal{J}_E$) have *cis*-eQTLs.

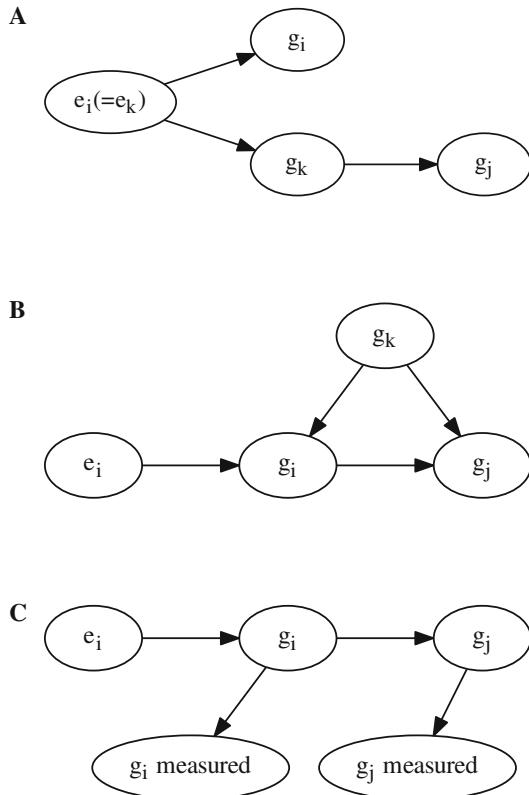


Fig. 1 Three “difficult” scenarios in causal network inference when testing the regulation from gene g_i to gene g_j , with the *cis*-eQTL e_i of g_i as the causal anchor, but also with interferences from another gene g_k or technical measurement errors. **(a)** Potential false positives, due to gene g_k that shares the same *cis*-eQTL with g_i , can be distinguished by the conditional independence test. **(b, c)** False negatives arise in conditional independence test, due to confounder gene g_k **(b)** or technical measurement errors on top of the unobservable true values of g_i **(c)**

We denote the expression level (FPKM) of gene g_i for individual j as $g_{i,j}$, whose matrix form is

$$\mathbf{G} \equiv (g_{i,j}), \quad i = 1, \dots, G, \quad j = 1, \dots, N. \quad (1)$$

Similarly, for gene g_i , the genotype of its *cis*-eQTL (see Note 2) for individual j is defined as $e_{i,j}$, with the matrix form:

$$\mathbf{E} \equiv (e_{i,j}), \quad i = 1, \dots, E, \quad j = 1, \dots, N, \quad (2)$$

where each genotype is limited by the number of alleles, N_a , as:

$$e_{i,j} \in \{0, 1, \dots, N_a\}. \quad (3)$$

The unknown gene regulation network can be represented as the posterior probability of regulation between every pair of genes, given the observed data, as:

$$w_{i,j} \equiv P(g_i \rightarrow g_j | \mathbf{E}, \mathbf{G}), \quad i, j = 1, \dots, G, \quad i \neq j. \quad (4)$$

Regulations are identified solely according to their expression and eQTL patterns, independent of the underlying mechanism or whether the regulation is direct (*see Note 3*).

Causal inference utilizes the *cis*-eQTL of every regulator gene to map the probability of regulation for all its possible targets. Therefore, genes without any *cis*-eQTL ($\mathcal{J}_{E+1}, \dots, \mathcal{J}_G$) are regarded as only target genes but not as regulators, with

$$w_{i,j} = 0, \quad \text{for } i = E + 1, \dots, G. \quad (5)$$

The expression levels of all possible regulators are also a sub-matrix of \mathbf{G} as:

$$\mathbf{G}_{\text{reg}} \equiv (g_{i,j}), \quad i = 1, \dots, E, \quad j = 1, \dots, N. \quad (6)$$

Given the expression levels \mathbf{G} and *cis*-eQTL genotypes \mathbf{E} , the question is to compute the probability of regulation matrix:

$$\mathbf{W} \equiv (w_{i,j}), \quad i = 1, \dots, E, \quad j = 1, \dots, G. \quad (7)$$

Since $w_{i,j}$ represents the probability of interaction $g_i \rightarrow g_j$, a larger value within the range 0 to 1 would indicate a higher significance.

2.2 The Findr Program

Findr (Fast Inference of Networks from Directed Regulations) is an efficient C library to reconstruct causal gene networks, whose methods can deal with the unique challenges in genomic and transcriptomic datasets, and are sketched in Subheading 4. Findr provides interfaces in python, R, and command line (*see Note 4*). Its efficient implementation and analytical calculation of null distributions (Subheading 4.2.2) are pivotal to the speedup of nearly one million times compared to existing programs [14]. This allows for whole-transcriptome causal network inference on modern datasets.

As a demonstrative example, we use the Findr R package with version 1.0.3 (<https://doi.org/10.5281/zenodo.1036757>) in this chapter (*see Note 5*).

2.3 Computing Environment

At the time of writing, the latest Findr R package (version 1.0.3) requires the following computing environment:

- A modern personal computer, or a high-performance computing environment (*see Note 6*).

- A modern Linux or Mac operating system.
- The GCC compiler (*see Note 7*).
- A command-line environment (to install Findr).
- A recent R language environment (R, RStudio, etc.).

3 Whole-Transcriptome Causal Network from the Geuvadis Dataset

The Geuvadis project [2] measured genome-wide genotypes and gene expression levels in 465 human lymphoblastoid cell line samples. Using this dataset as an example, here we reconstruct a causal gene network with Findr in R.

3.1 Install Findr

The latest version of Findr (*see Note 8*) can be downloaded and installed with the following lines in command-line environment (*see Note 9*):

```
#Comments above a command explain its function
#Comments below a command (if present) show its expected output
#Download Findr R package from github (see Note 10)
git clone https://github.com/lingfeiwang/findr-R.git
#Install Findr
cd findr-R && R CMD INSTALL findr
```

3.2 Prepare Data

Here, we reconstruct a causal gene regulation network among 3000 genes in which 1000 have *cis*-eQTLs. The dataset was downsampled from the Geuvadis project (*see Note 11*). In R, the Findr library and the downsampled Geuvadis dataset can be loaded with:

```
#Load Findr
library(findr)
#Load downsampled Geuvadis dataset
data(geuvadis)
```

3.3 Reconstruct Network

Network inference is performed with the function *findr::pij_gassist*, taking E, G_{reg}, and G as the input and returning W as the output (*see Note 12*):

```
#Reconstruct causal gene network
w=findr::pij_gassist(geuvadis$dgt,geuvadis$dt,geuvadis$dt2,nodiag=TRUE)
#Examine output dimension
print(dim(w))
#[1] 1000 3000
```

The computation takes about one second on a modern desktop computer and scales linearly with the numbers of regulators, targets, and individuals.

3.4 Analyze and Visualize Network

3.4.1 Regulation Probabilities

To demonstrate the properties of the reconstructed causal network for human lymphoblastoid cell lines, we briefly analyze and visualize it below:

The distribution of posterior regulation probability $P(g_i \rightarrow g_j | \mathbf{E}, \mathbf{G})$ is visualized in Fig. 2 with code (see Note 13):

```
#Drop self-regulation probabilities
wnd=w
diag(wnd)=NA
wnd=w[which(!is.na(wnd))]

#Histogram of regulation probabilities
hist(wnd,breaks=50,main='Histogram of regulation probability',
xlab='Regulation probability')
```

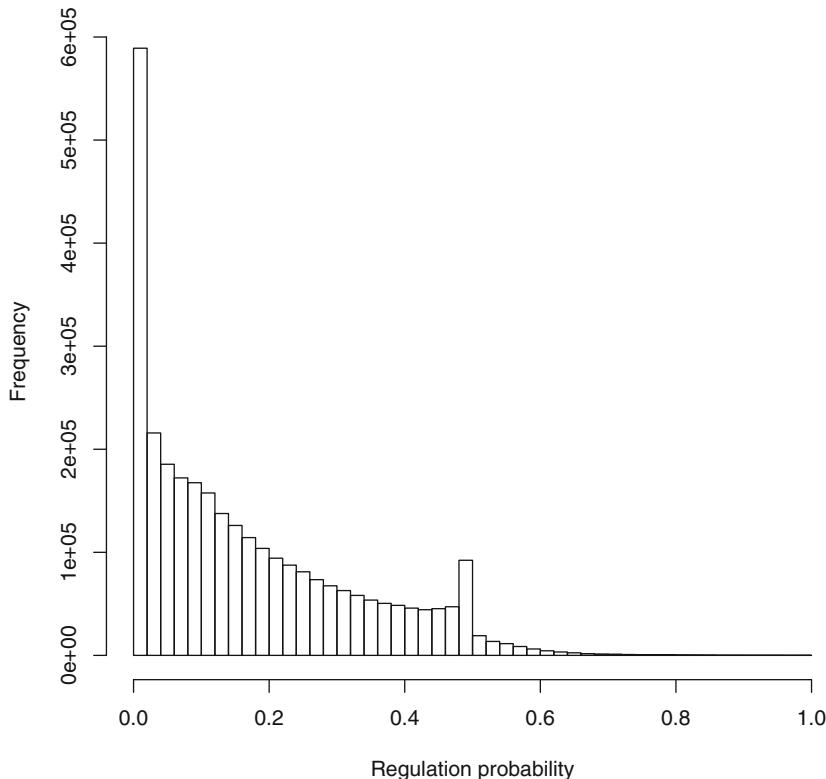


Fig. 2 The histogram of regulation probability (off-diagonal elements of \mathbf{W}) of the causal gene network reconstructed from the downsampled Geuvadis dataset

Distribution of out-degree

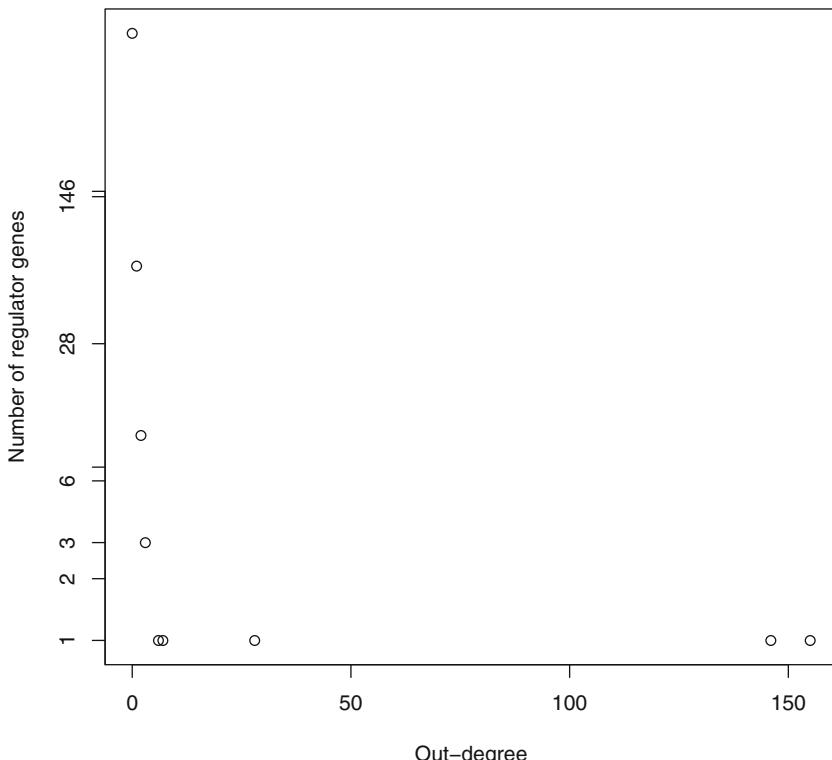


Fig. 3 The histogram of out-degree of regulator genes in the thresholded causal gene network reconstructed from the downsampled Geuvadis dataset

3.4.2 Out-Degree Distribution

The distribution of out-degree is visualized in Fig. 3 with code:

```
#Threshold network to 90% confidence level
threshold=0.9
net=w>=threshold

#Histogram of out-degree
odeg=rowSums(net)
vals=table(odeg)
plot(names(vals),vals,log='y',main='Distribution of out-degree',
     xlab='Out-degree',ylab='Number of regulator genes')
```

3.4.3 Top Hub Gene List

The list of top hub genes (by out-degree) is obtained below:

```
#Find top 5 hub genes, and their numbers of targets
ntop=5
odego=odeg[order(odeg,decreasing=TRUE)]
print(odego[1:ntop])
```

(continued)

#	ZNF77	SP2	PEA15	ZNF580	C6orf106
#	155	146	28	7	6

3.4.4 Network Visualization in Cytoscape

The reconstructed network can be exported to a csv file for visualization in Cytoscape (*see Note 14*), with the following code:

```
#Convert causal network to sparse format
dat=NULL
for (i in 1:dim(net)[1])
  if(any(net[i,]))
    dat=cbind(dat, rbind(rownames(net)[i], names(which(net[i,]))))
dat=t(dat)
colnames(dat)=c('source','target')
print(dat[1:2,])
  #           source          target
  #[1,] "ARHGEF35"      "OR2A1"
  #[2,] "ARHGEF35"      "OR2A42"
#Export sparse network to dat.csv
write.csv(dat,'dat.csv',row.names=FALSE,quote=FALSE)
```

Cytoscape can then import the network in dat.csv and visualize it. The largest connected component is shown in Fig. 4.

4 Statistical Methods

In this section, we outline the statistical methods used in Findr. (For details, *see [14]*.)

4.1 Data Normalization

To satisfy the assumptions of linear dependency and normal noise distribution, and also to remove outliers (*see Note 15*), the expression levels of each gene are transformed to follow the standard normal distribution, based on the expression level ranking across individuals. Each gene is normalized separately.

4.2 Causal Inference Subtests

Consider all possible regulatory relations between the triplet (A, B, C) , in which B is the regulator gene, A is its *cis*-eQTL, and C is a potential target gene. Causal inference performs three subtests in Table 1 (*see Note 16*) to narrow down their relations to contain $A \rightarrow B \rightarrow C$, but with a false-positive rate as low as possible. Each subtest compares a null and an alternative hypothesis, by first performing a likelihood ratio test, then converting the likelihood ratio into p -values, and finally computing the posterior probability of alternative hypotheses given the observed data, which is equivalent to the *local* False Discovery Rate (FDR). This is similar with Genome Wide Association

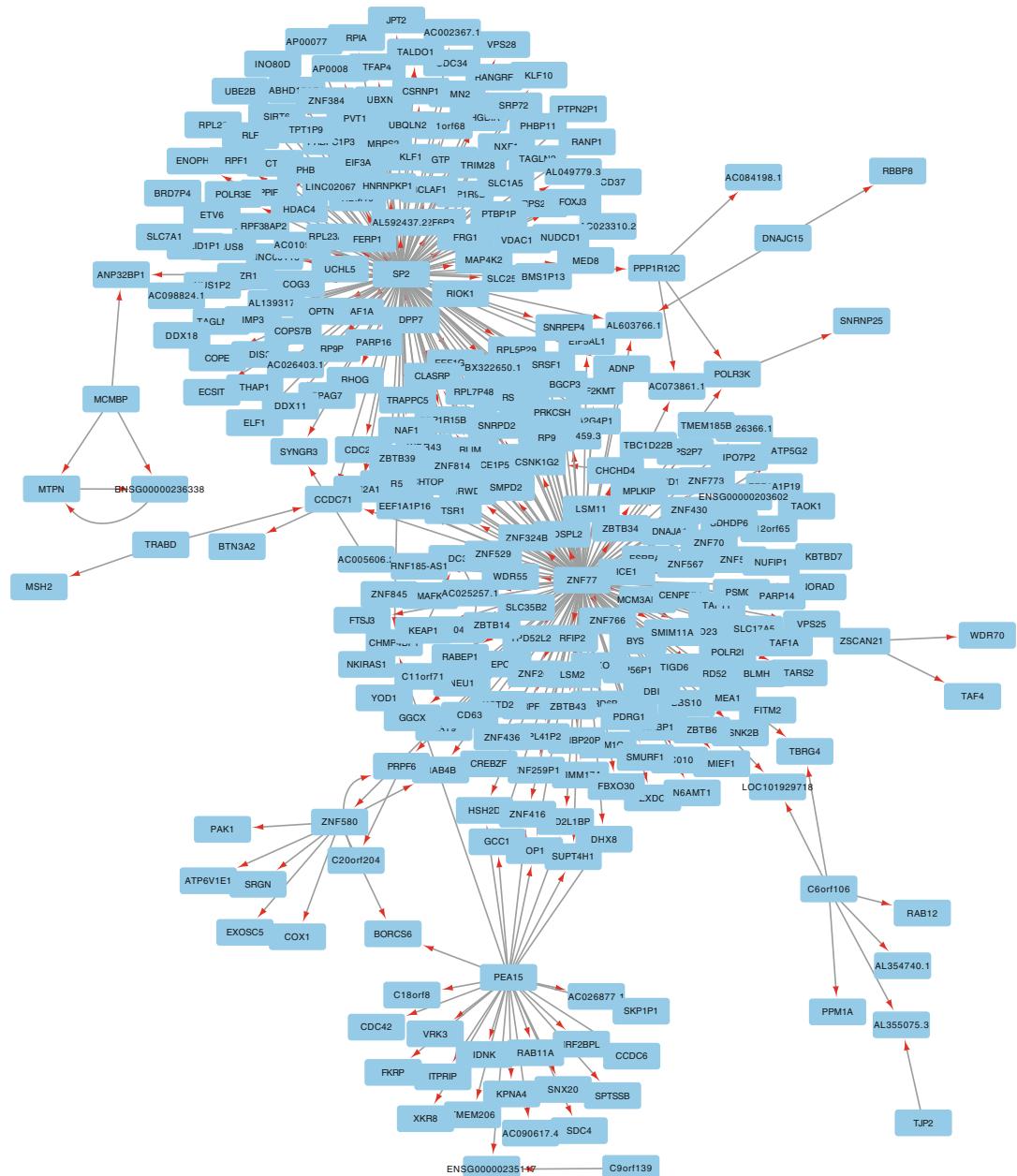
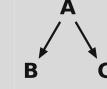


Fig. 4 The largest connected component of the reconstructed and thresholded regulation network from the downsampled Geuvadis dataset, as visualized in Cytoscape. Of the 3000 genes, 308 are in this connected component with 366 regulations

Studies, in which the Pearson correlation (equivalent to likelihood ratio) is first computed, and then converted into p -values and FDRs.

Table 1

Three subtests of causal inference performed to test the gene regulation $B \rightarrow C$, with the *cis*-eQTL of B as the causal anchor A

Test ID	Test name	Null hypothesis	Alternative hypothesis
1	Secondary (linkage)	$A \quad C$	$A \rightarrow C$
2	Controlled		
3	Relevance		

These tests are numbered, named, and defined in terms of null and alternative hypotheses as shown. Arrows in a hypothesis indicate directed regulatory relations (from [14])

4.2.1 Likelihood Ratio Tests

In Table 1, each graph represents a probabilistic dependency model among the (A, B, C) triplet. The expression level of each gene is modeled as following a normal distribution, whose mean depends additively on all its parents. The dependency is linear on other gene expressions, and categorical on genotypes. Based on the normally distributed models, the likelihood ratio between the alternative and the null hypotheses can be computed for each subtest.

As an example of mathematical expression of graphical models, in the secondary linkage test, the expression level of gene C for sample i is modeled in the alternative hypothesis with a normal distribution:

$$c_i \sim N(\mu_{a_i}, \sigma_c^2). \quad (8)$$

Model parameters include $\mu_j, j \in \{0, \dots, N_a\}$ and σ_c . The null hypothesis dissociates C from A , thereby additionally equating all $\mu_j = \mu$, leaving only μ and σ_c as its parameters. After deriving the mathematical models from graphical models, the likelihood ratio is the ratio of maximum likelihoods between the alternative and null hypotheses.

4.2.2 P-values

For each subtest, the null distribution of likelihood ratio may be obtained either by simulation or analytically. Regardless of the method, likelihood ratios can then be converted into p -values according to their rankings in the null distribution. In [14], we found that the null distribution can be computed analytically, therefore avoiding simulations and accelerating the computations in Findr by ~ 1000 times.

4.2.3 Posterior Probabilities of Alternative Hypotheses

P -values can be further reformulated into the posterior probabilities of alternative hypotheses, according to [9, 16]. This is achieved by regarding the computed p -values as a mixture of null p -values (following the standard uniform distribution) and alternative p -values (following an unknown distribution with a decreasing probability density function that vanishes at one), with

unknown proportions. These proportions can be inferred from the number of least significant p -values, which then allows us to estimate the posterior probability of alternative hypothesis at any p -value with Bayesian inference.

These posterior probabilities are termed “local FDRs,” as opposed to FDRs (see Note 17). Findr implements a simplified estimator (see Note 18), with the resulting posterior probabilities denoted as:

$$\begin{aligned} p_{i,j}^{(k)} &\equiv P(\text{alternative hypothesis in subtest } k \text{ for } A = e_i, \\ &B = g_i, C = g_j \mid \mathbf{E}, \mathbf{G}). \end{aligned} \quad (9)$$

4.3 Subtest Combination

Findr computes the final probability of regulation by combining the subtests as:

$$w_{i,j} \equiv \frac{1}{2} \left(p_{i,j}^{(1)} p_{i,j}^{(2)} + p_{i,j}^{(3)} \right). \quad (10)$$

By combining the secondary linkage and controlled tests, the first term verifies that the correlation between g_i and g_j is not entirely due to pleiotropy. By replacing the conditional independence test in [9] with the controlled test, this combination is robust against hidden confounders and technical variations.

On the other hand, the relevance test in the second term can identify interactions that arise from the indirect effect $e_i \rightarrow g_i \rightarrow g_j$ but are too weak to be detected by the secondary linkage test. In such cases, the data reveals nothing more than co-expression, and therefore the direction of regulation cannot be determined. Correspondingly, the coefficient $\frac{1}{2}$ assigns half of the probability to each direction.

5 Notes

1. The whole chapter is equally applicable on gene isoforms.
2. Findr only uses one *cis*-eQTL for each regulator gene. When multiple *cis*-eQTLs coexists, the most significant one is usually selected for the strongest test signal.
3. However, direct regulations tend to have stronger significance than the indirect regulations they form, if the relevant genes have similar levels of technical variations.
4. URLs for Findr library and interfaces:
 - Library: <https://github.com/lingfeiwang/findr>
 - Command-line interface: <https://github.com/lingfeiwang/findr-bin>
 - Python interface: <https://github.com/lingfeiwang/findr-python>
 - R package: <https://github.com/lingfeiwang/findr-R>

5. Findr's python and command-line interfaces have the same functionality with similar computing environment requirements.
6. The computer memory (RAM) limits the size of data Findr can process. Findr's python and command-line interfaces can automatically split the dataset according to the specified memory usage limit. The R package requires manual division. A modern computer with 4GB of RAM can handle whole-transcriptome network inference for hundreds of individuals, with thousands of genes having *cis*-eQTLs and tens of thousands having not.
7. GCC is already installed on most Linux machines. On Mac OS, Apple's compiler may pretend to be GCC, so Findr may fail to install. For the solution, see FAQ in <https://github.com/lingfeiwang/findr/blob/master/doc.pdf>. GCC can be downloaded from <https://gcc.gnu.org>.
8. To exactly reproduce the example, Findr 1.0.3 can be downloaded from <https://doi.org/10.5281/zenodo.1036757>.
9. Detailed installation instructions and FAQs are available in the full manual at <https://github.com/lingfeiwang/findr/blob/master/doc.pdf>, or the doc.pdf file of the corresponding version.
10. Without git, the latest version of Findr can also be downloaded from <https://github.com/lingfeiwang/findr-R/archive/master.zip>. The user then needs to uncompress it and enter the corresponding folder before installation.
11. The original, full Geuvadis dataset can be downloaded from ArrayExpress (<https://www.ebi.ac.uk/arrayexpress/>), in accessions E-GEUV-1 and E-GEUV-2. In this analysis, 360 unique European individuals of 465 in total are considered. Geuvadis reported 23,722 genes expressed (in $\geq 50\%$ individuals) after QC. Genetic loci without any variation were discarded. After that, the Geuvadis QTL analysis identified 3172 genes with at least one significant *cis*-eQTL.
Findr includes a random subset of these data for illustration purposes, totaling 1000 genes with *cis*-eQTLs and 2000 more without *cis*-eQTLs. For each of the 1000 genes, the genotypes of its strongest *cis*-eQTL are also included.
- The same network analysis can be performed on other datasets, such as on the full Geuvadis dataset to reconstruct the whole-transcriptome causal gene network among 23,722 genes. This requires an already performed eQTL analysis for the dataset, either from software tools such as matrixeQTL [17] or fastQTL [18] or from existing studies.
12. Function description for
`findr.pij_gassist(dg,dt,dt2,na=NULL,nodiag=FALSE)`:

- dg: Input integer matrix \mathbf{E} of *cis*-eQTL genotype data, as defined in Eq. (2). The element $[i, j]$ is the genotype value $(0, 1, \dots, N_a)$ of the *cis*-eQTL of regulator gene i for individual j .
- dt: Input double matrix \mathbf{G}_{reg} of regulator gene expression level data, as defined in Eq. (6). The element $[i, j]$ is the expression level of regulator i for individual j .
- dt2: Input double matrix \mathbf{G} of all gene expression level data, as defined in Eq. (1). The element $[i, j]$ is the expression level of gene i for individual j .
- na: The number of alleles N_a . If unspecified, it is automatically detected as the maximum value of dg.
- nodiag: This function can infer networks for regulators that either should or should not be regarded as targets. In the earlier case, the regulators should also appear before other genes as targets, and nodiag should be TRUE. In the latter case, there should be no overlap between regulators and targets, with nodiag=FALSE.
- Return value: Output double matrix \mathbf{W} of inferred probability of regulation, as defined in Eq. (7). The element $[i, j]$ is the probability of regulation from regulator i to target j after observing the input data.

13. The peak at regulation probability around 0.5 is due to correlated regulator and target genes, whose regulation direction cannot be determined with the *cis*-eQTL of the regulator. In such cases, the novel combination of causal inference tests assumes a half probability for each direction.

14. <http://www.cytoscape.org/>.

15. Although the data normalization step attempts to transform gene expression levels into the standard normal distribution, ill-distributed datasets may still cause underperformance from the method and should therefore be analyzed with extra care. Examples may include a large proportion of ties in gene expression levels, from single-cell transcriptomics or sparsely expressed genes.

16. Here, we omit the primary linkage test, comparing $A \rightarrow B$ against $A \leftarrow B$, because the *cis*-eQTL A is assumed to be significant. These tests are also numbered differently with [14].

17. For the difference between FDR and local FDR, see [19].

18. Findr also skips the computation of P -values when deriving the posterior probabilities using the null distribution. For more stringent local FDR conversion with Grenander estimator, the user can first compute p -values within Findr (findr.pijs_gassist_pv) and then use other software tools (e.g., fdrtool in R from [19]) to obtain local FDRs.

6 Future Perspectives

The reconstruction of causal networks can be potentially extended to various data types. For example, other causal anchors to infer causal gene networks may include epigenetic markers, copy number variations, and perturbation screens. By targeting tissue- or species-specific genes, Findr may also reconstruct cross-tissue or host-pathogen/microbiota causal gene networks. The same analysis may also apply on gene isoforms, proteome, etc., to reconstruct multi-omics causal networks. By considering different distributions of technical variation, the very same causal inference may also infer cell type-specific causal networks from single-cell datasets. Each of those perspectives contains its unique challenges that are worth addressing in the future. However, the statistical and computational frameworks have already been laid down.

Acknowledgements

Development of Findr was supported by grants from the BBSRC [BB/J004235/1, BB/M020053/1].

References

- Goodwin S, McPherson JD, McCombie WR (2016) Coming of age: ten years of next-generation sequencing technologies. *Nat Rev Genet* 17(6):nrg.2016.49
- Lappalainen T, Sammeth M, Friedländer MR, 't Hoen PA, Monlong J, Rivas MA, González-Porta M, Kurbatova N, Griebel T, Ferreira PG, Barann M, Wieland T, Greger L, van Iterson M, Almlöf J, Ribeca P, Pulyakhina I, Esser D, Giger T, Tikhonov A, Sultan M, Bertier G, MacArthur DG, Lek M, Lizano E, Buermans HPJ, Padialou I, Schwarzmayr T, Karlberg O, Ongen H, Kilpinen H, Beltran S, Gut M, Kahlem K, Amstislavskiy V, Stegle O, Pirinen M, Montgomery SB, Donnelly P, McCarthy MI, Flück P, Strom TM, The Geuvadis Consortium, Lehrach H, Schreiber S, Sudbrak R, Carracedo A, Antonarakis SE, Häslar R, Syvänen A-C, van Ommen G-J, Brazma A, Meitinger T, Rosenstiel P, Guigó R, Gut IG, Estivill X, Dermitzakis ET (2013) Transcriptome and genome sequencing uncovers functional variation in humans. *Nature* 501(7468):506–511
- The GTEx Consortium (2015) The Genotype-Tissue Expression (GTEx) pilot analysis: multitissue gene regulation in humans. *Science* 348(6235):648–660
- Ashley EA (2016) Towards precision medicine. *Nat Rev Genet* 17(9):507–522
- Schadt EE, Friend SH, Shaywitz DA (2009) A network view of disease and compound screening. *Nat Rev Drug Discov* 8(4):286–295
- Talukdar HA, Foroughi Asl H, Jain RK, Ermel R, Ruusalepp A, Franzén O, Kidd BA, Readhead B, Giannarelli C, Kovacic JC, Ivert T, Dudley JT, Civelek M, Lusis AJ, Schadt EE, Skogsberg J, Michoel T, Björkegren JL (2016) Cross-tissue regulatory gene networks in coronary artery disease. *Cell Syst* 2(3):196–208
- Rockman MV (2008) Reverse engineering the genotype–phenotype map with natural genetic variation. *Nature* 456(7223):738–744
- Lawlor DA, Harbord RM, Sterne JAC, Timpson N, Smith GD (2008) Mendelian random-

- ization: using genes as instruments for making causal inferences in epidemiology. *Stat Med* 27(8):1133–1163
9. Chen LS, Emmert-Streib F, Storey JD (2007) Harnessing naturally randomized transcription to infer regulatory relationships among genes. *Genome Biol* 8:R219
 10. Millstein J, Chen GK, Breton CV (2016) cit: hypothesis testing software for mediation analysis in genomic applications. *Bioinformatics* 32(15):2364–2365
 11. Greenland S (1980) The effect of misclassification in the presence of covariates. *Am J Epidemiol* 112(4):564–569
 12. Li Y, Tesson BM, Churchill GA, Jansen RC (2010) Critical reasoning on causal inference in genome-wide linkage and association studies. *Trends Genet* 26(12):493–498
 13. Cole SR, Platt RW, Schisterman EF, Chu H, Westreich D, Richardson D, Poole C (2010) Illustrating bias due to conditioning on a collider. *Int J Epidemiol* 39(2):417–420
 14. Wang L, Michoel T (2017) Efficient and accurate causal inference with hidden confounders from genome-transcriptome variation data. *PLoS Comput Biol* 13(8):e1005703
 15. Hemani G, Tilling K, Smith GD (2017) Orienting the causal relationship between imprecisely measured traits using genetic instruments. *bioRxiv*, pp 117101
 16. Storey JD, Tibshirani R (2003) Statistical significance for genomewide studies. *Proc Natl Acad Sci* 100(16):9440–9445
 17. Shabalin AA (2012) Matrix eQTL: ultra fast eQTL analysis via large matrix operations. *Bioinformatics* 28(10):1353–1358
 18. Ongen H, Buil A, Brown AA, Dermitzakis ET, Delaneau O (2016) Fast and efficient QTL mapper for thousands of molecular phenotypes. *Bioinformatics* 32(10):1479–1485
 19. Strimmer K (2008) fdrtool: a versatile R package for estimating local and tail area-based false discovery rates. *Bioinformatics* 24(12):1461–1462



Chapter 5

Causal Queries from Observational Data in Biological Systems via Bayesian Networks: An Empirical Study in Small Networks

Alex White and Matthieu Vignes

Abstract

Biological networks are a very convenient modeling and visualization tool to discover knowledge from modern high-throughput genomics and post-genomics data sets. Indeed, biological entities are not isolated but are components of complex multilevel systems. We go one step further and advocate for the consideration of causal representations of the interactions in living systems. We present the causal formalism and bring it out in the context of biological networks, when the data is observational. We also discuss its ability to decipher the causal information flow as observed in gene expression. We also illustrate our exploration by experiments on small simulated networks as well as on a real biological data set.

Key words Causal biological networks, Gene regulatory network reconstruction, Direct Acyclic Graph inference, Bayesian networks

1 Introduction

Throughout their lifetime, organisms express their genetic program, i.e., the instruction manual for molecular actions in every cell. The products of the expression of this program are messenger RNAs (mRNAs); the blueprints to produce proteins, the cornerstones of the living world. The diversity of shapes and the fate of cells is a result of different readings of the genetic material, probably because of environmental factors, but also because of epigenetic organizational capacities. The genetic material appears regulated to produce what the organism needs in a specific situation. We now have access to rich genomics data sets. We see them as instantaneous images of cell activity from varied angles, through different filters. Patterns of action of living organisms can be deciphered using adequate methods from these data. Probabilistic

(in)dependence relationships can most certainly be extracted from observational data, but we want to go one step further. If the data is generated from a causal structure, what clues can be found in the data to help uncover this hidden structure without resorting to intervention? Intervention is not always practical, nor ethical. In our biological network study context, even though modern techniques allow scientists to intervene on the system, e.g., knocking out one gene or blocking a molecular pathway, they are still expensive experiments. We aim to investigate which kinds of causal links can (and which cannot) be discovered from observational data, defined by biologists as wild type and steady state.

In short, this chapter is reviewing causal reconstruction in gene network from observational data. More specifically, it blends a presentation of Systems Biology elements, methods for gene network reconstruction, and concepts in statistical causality in Subheading 2. This section narrows down the type of system under scrutiny, namely gene regulatory networks, although our work could apply to more detailed representations of biological systems if the data were available. Subheading 3 introduces, in detail, the class of methods we chose to focus on: Bayesian networks. Despite being designed to decipher probabilistic (in)dependencies in the data, we test its capacity to learn causality from observational data. This is the purpose of the preliminary experiments on small networks and an application on a real network in Subheading 4. All simulations were performed with the aid of the R package `bnlearn` [1, 2]. We conclude with a short discussion in Subheading 5.

2 Biological Networks and Causality

Biological system inner organization is at the origin and is the essence of the diversity of life we observe around us, while allowing organisms to adapt to their environments. They have received growing attention for the last 20 years [3–5], and are complex both in terms of their many different kinds of interacting elements that they constitute and in terms of the diversity of the responses that they drive. They can be understood at different levels: from the interactions of species within an ecosystem to the finer molecular grain through interactions between cells or the communications between organs. We focus on the latter situation in the present work.

2.1 Networks for Biological Systems

Biological networks offer an integrated and interpretable tool for biological systems knowledge modeling and visualization. Nodes in these networks stand for biological entities (e.g., genes and proteins), while edges encode interactions between these entities [6]. The definition of the biological entities is far from trivial—for example, the definition of a gene is not unique [7, 8]. In our “small world” modeling representation of the reality [9], we use simplified definitions of these biological elements, and coin them variables in our system. Some nodes can represent a phenotype of interest: for example, the yield or level of resistance to a disease or to an adverse environmental factor [10]. An edge, on the other hand, has a more diffuse meaning: ranging from a vague association of the measurements linked to the nodes to a functional dependency between them driven by an explicit mechanism. We will be primarily interested in directed edges; we aim to go beyond co-expression or association networks [11–13]. In the real biological world, mechanisms exist which trigger pathways within the system of a living organism [14] and sometimes between living organisms [15, 16]. On the analytical side of this, we aim to providing a meaningful representation of these interactions, not to say regulations between the components in the system under study. We will use *directed acyclic graphs* [17], although it is known that gene networks do indeed contain feedback cycles [18–20]. Solutions exist to consider feedback cycles in the Bayesian network framework. For example, time can be used to unfold the successive regulations taking place between the elements of a cycle [21–23]. When a time series is not available, some works detail the existence of feedback cycles as a super graph associated to a family of distributions instead of a unique factorized distribution [24]. This is essentially inspired by an inversion of the technique used to condense (aka contract) all strongly connected components (which contain cycles) into a single composite node [25]. Notice that each distribution would then correspond to different causal assumptions which would then need to be tested using additional data. In [26], the authors give consistency results for an algorithm to recover the true distribution of non-Gaussian observational data generated from a cyclic structure.

If one is interested in representing a fine grain of biological details, interconnected networks depicting different types of biological entities can be used. A small example can be found in Fig. 1 of Chapter 1 of the present book. Some apparent relationships between genes and phenotypes are often the result of direct regulatory links between different sorts of biological entities. We focus on this gene regulatory network (GRN) with the assumption that it can be deciphered from transcript levels.

2.2 Data and Reconstruction Methods

Modern high-throughput technologies such as RNA sequencing allow the researcher to have access to the simultaneous levels of all transcripts in a biological sample. They provide a partial snapshot of the state of the cell. A first hindrance in analyzing such data set is related to their inherent noise and high-dimensionality [27]; much more variables are observed and the low number of samples makes the reconstruction task very challenging. A vast number of methods were developed to circumvent this difficulty with difference performance guarantees [28–30]. A second obstacle is concerned with the nonlinearity of the relationships [31], sometimes immersed in intricate temporal responses [32–34]. Another feature of modern biological data sets is that of missing observations, either due to technical fault or because all relevant variables cannot be monitored [35]; adequate techniques need be implemented to deal with this [36–39]. We will explicitly note the difficulty of modeling feedback cycles with traditional (yet advanced) analysis methods [40].

Many review papers can now be found in the bioinformatics literature on gene network reconstruction [41–44], and also in the statistical literature [37, 45–47]. Other papers generally compare their relative merits and pitfalls [48–51]. In this vein, the Dialogue for Reverse Engineering Assessment and Methods (DREAM) project, over the last decade, has also ran several challenges related to network reconstruction [52–54]. Lessons were learned about the power of combining complementary reconstruction methods [52, 55], and such methods showed great performance in reconstructing real networks from genuine data sets related to cancer [56, 57], protein signaling [58, 59], or to fitness-related data [60]. When only observational data is available, bounds on causal effects can be retrieved (see the discussion on EoC or CoE below as well) and verified by means of intervention (knock-out) experiments [61, 62]. In no instance are we pretending to be exhaustive, since this research area is vast and constantly expanding. Our focus is to test the ability of methods to decipher causal links from steady-state observational data. For example, we ignore single-cell data [58] and gene expression time-series modeling [31, 63]. Neither do we consider the addition/integration of biological knowledge or of complimentary data sets [64] or a supervised framework [65]. Lastly, we do not cover purely interventional designs [66–68]. In some sense, our work is related to the work of [69] or that of [70], but we explore beyond the cause–effect pair of variables or the treatment effect. Borrowing ideas developed in any of these paradigms would certainly be a winning strategy, if they revealed useful information in our transcript level steady-state context.

The reader interested in a classified list of gene network reconstruction methods is encouraged to read through Subheadings 3–5 of Chapter 1 of the present book. We note here that correlation-

based methods do not lead to causal links between variables. An asymmetry (and hence directions on edges) can only be inferred by considering transcription factor genes as likely sources and other genes as likely targets (see also [71] as a generalization of [72] in that it identified transcription factors). In a Gaussian graphical model setting, an attempt to infer directions was made in [73] using comparisons between (standardized) partial variances and the notions of relative endogeneous/exogeneous nodes. The rationale of this distinction is that nodes with a higher level of remaining variance are more likely to be source of a directed edge, while those nodes with lower partial variance are more likely to be targets. The simple counterexample in the Box below and in Fig. 1 shows how directions can be created artificially. En route to causality, structural equation modeling (SEM [74]) assumes that the expression of each gene can be “programmed” as a linear combination of every other gene expression (for nonzero path coefficients), all other factors, and of a noise term. The last method we mention here is a class of probabilistic graphical model, namely *Bayesian Networks*. We postpone their detailed presentation until Subheading 3. To the best of our knowledge, Bayesian networks were first used for GRN reconstruction by Friedman et al. [75], and their ability to orient some edges and not others is well established [17, 76, 77].

We consider here the graph in Fig. 1a, which corresponds to a covariance matrix defined as:

$$M = \begin{pmatrix} 1 & \rho & \rho^2 \\ \rho & 1 & \rho \\ \rho^2 & \rho & 1 \end{pmatrix}$$

Now, assuming that enough data is available, the maximum likelihood estimate of the concentration matrix (i.e., the inverse of the covariance matrix), which specifies direct relationships in a Gaussian graphical setting, is attained:

$$K = \frac{1}{1 - \rho^2} \begin{pmatrix} 1 & -\rho & 0 \\ -\rho & 1 + \rho^2 & -\rho \\ 0 & -\rho & 1 \end{pmatrix}$$

If we follow the direction rule of the approximate causal algorithm of [73], we obtain the DAG in Fig. 1b. In fact, the partial variances of variables A and C are then equal to $1 - \rho^2$ and hence larger than that of node B , equal to $\frac{1-\rho^2}{1+\rho^2}$. Notice that this is conditional on having the edges and the directions declared significant, so it depends on the value of the correlation ρ and on the sample size.

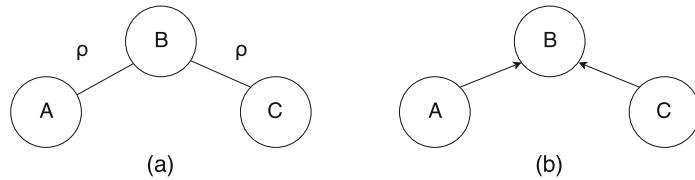


Fig. 1 (a) Simple three-node network, and (b) reconstructed network from ideal data using the method in [73]

2.3 Causality

Causality is the intrinsic connection through which one event (the cause) triggers another one (the effect) under certain given conditions. Issues about causality and inductive learning are often ascribed to have originated with Hume's work on deciphering human cognitive reasoning as a scientific empirical study [78]: that causation exists between our own experiments rather than between material facts; something can be learned from data. Major statisticians introduced such concepts in statistical data analysis at the beginning of the twentieth century [79–81]. The modern era of causality started with considerations of randomized vs. nonrandomized experiments in [82] working with observational data, a review paper linking counterfactual and philosophical considerations [83], and [77] aimed at unifying the different mainstream causal approaches. Science dived into seeking the Effects of Causes (EoC): what is the difference in outcome between the result of applying a treatment, and what would have happened if the treatment had not been applied. The latter of the two questions is referred to as a counterfactual statement. In this framework, the cause is (partly) responsible for the effect, while the effect is (partly) dependent on the cause. Note the addition of the word partly here to clearly stress the likely, but not absolutely certain, aspect of the causation mechanism: statistics are concerned with stochastic events, not deterministic laws. Moreover, one cause is not exclusive of other causes for one consequent effect—several causal factors can be the cause of the effect and can have additive effects or different interactions with each other. A causal sequence is generated and determined by natural mechanisms, generally several of them. These can be read from a temporal relationship. However, when nature does not explicitly reveal its mechanisms, learning about causality from data is extremely challenging. Wainer [84] took a simple example so as to disentangle the research question; whether pupils achieving good results at school are happy because of their performance, or it is their intrinsic happiness which makes them good at school. In such cases, we need some sort of intervention achieved via randomization of subjects to test hypotheses.

Historically, the language of counterfactual reasoning [85] has been used to define causal relationships from conditional

statements (“what if something was different in the past”): if event A had have occurred, then B would have also occurred. Although the use of the subjunctive mood suggests that the antecedent is false, the theory encompasses true or false antecedents. Though not universal, e.g., in a decision theoretic framework [86], the counterfactual inference mechanism is natural to compute the EoC. [17] essentially formalized the use of counterfactuals for statistical causation inference. The “ B would have been b , had A been a ” statement means that the $A \rightarrow B$ relationship can be read as “if we replace the equation determining A by a constant value $A = a$, then the solution of the equations for variable B will be $B = b$ ” (formally $A = a \rightarrow B = b$). This allows the author to define the do operator and state the property of the modification of the joint distribution under intervention (see Subheading 3.10). Among the tools used in causal inference, we mention Instrumental Variables [87], Causal Trees [70], Path Analysis [79], Structural Equation Modeling [88, 89], G-computation [90], Bayesian Decision-Making inspired approaches [86], more recent Kernel Methods [91], and Probabilistic Causality [92, 93] (Pearl argued that a confusion was first made with the statement that cause increases probability [17], whereas some assumptions need to be checked to link the two notions). Applications of statistical causality can be found in Psychology [94], Epidemiology [95], Social Research [96], Economics [76], Project Management [97], Marketing [98], Human [99], and Veterinary [100] Medicine.

We now integrate the concepts of causality into systems biology and gene networks. We will not address the relationships from marker to phenotype, although the kind of relationship can be similar, the causality necessarily originates from the genome [101]. [102] introduces high-dimensional techniques for Quantitative Trait Locus (QTL) detection in recent data sets in which gene-gene interactions are accounted for. The perspective is more a pragmatical one for breeding research to improve traits of interest, rather than to dissect the molecular regulations. eQTL causal networks could be considered if we had genomic data in addition to transcriptomics [103] (see also Chapter 4 of the present book).

2.4 Causality in Networks

The concept of causality in gene network reconstruction is about distinguishing “direct” regulatory relationships [104] between biological components of the network from “association.” Causal inference methods for gene network reconstruction are at the cornerstone of biological knowledge discovery, and may indeed be useful in solving biomedical problems. Randomized trials [105] in this context are highly impractical, very costly, and even unethical in most instances [106]. Gene co-expression is arguably not enough [107], and most of the time methods rely on time series [32, 33, 108–111], perturbation [110, 112] (sometimes combined with observational data [113, 114]), or genetical genomics

[55, 115, 116] experiments. We will instead focus purely on observational (aka steady-state wild-type) data. As an example, an application of predicting the phenotype accounting for gene interaction and obtaining ranges of causal effects from a theoretical point of view can be found in [61] with the R `pcalg` companion package in [117].

Our choice is to test Bayesian Networks (BNs) for their inherent ease of representation of a complex system, with an associated probabilistic distribution under some mild assumptions. [98] proposed that Bayesian networks have a limited applicability in the reconstruction of causal relationships. Methods for BN inference are reviewed in [118] and in [2]. Arrows in these graphical networks are more like lines on the sketch of a construction plan—they encode conditional independence relationships, which by definition are symmetrical. Bayesian network arrows need not represent causal relationships, in fact, not even “influential” relationships. It is for this reason that it is perfectly valid to use prognostic or diagnostic models; the former estimates the chances of developing an outcome, and the latter estimates the chances of having this outcome [119]. This distinction is also that between Effect of Cause (EoC, e.g., “ Y is in state y . Will a change in X have an effect on Y ?”) or Cause of Effect (CoE, e.g., “ Y has changed its value. Is it because X was altered?”) [120]. With the exception of courthouse business, most causal questions—at least in science—are concerned with EoC via adequate designs of experiments [121]. CoE questions are hindered by situations where appropriate confounding effects are difficult to disentangle, perhaps due to missing information (for example, missing variables [39]). In this case, typically, one would need strong prior knowledge, or to make strong assumptions to allow the access of bounds on “Probability of Cause” to be estimated. To return to similar problems in gene network reconstruction, the reader is directed to [122].

2.5 Causal Interpretation of Bayesian Networks via Node Intervention

Directed Acyclic Graphs (DAGs) like the one in Fig. 3 encode conditional independence relationships between the variables at the vertices. For instance, in this graph, S and W are independent, conditional on G and R . G and R , however, are not independent nor conditionally independent on any other subset of nodes. In a pure graph theoretic approach, directed edges (parent/child relations) should not go through any interpretation beyond these independence relationships. Score-based or independence learning algorithms seek graph structures and conditional probabilities which optimally fit the data. On the frequentist side, there is no reason so as to prefer a member of the equivalence class of a learned graph (see Subheading 3). On the Bayesian side, prior beliefs can steer our confidence towards one structure or another. Regardless, one is often tempted to say that a directed edge

$X \rightarrow Y$ stands for some kind of causal dependence of X on Y . [123] asserted that one can retrieve causal relationships from observational data by resorting to considering directed edges of the essential graph. On the other hand, [118] claimed that taking the reasoning from immoralities to a causal interpretation is a fallacy. Unless there is clear knowledge in the system, causal statements are often acceptably left vague, and they can only be clarified with the vocabulary of interventions in DAGs [77]. This extends the meaning of the independencies encoded in classical graphical models to an enriched set of distributions under interventions on nodes. An *intervention* is defined as an operator setting a variable to a specified distribution. The most classical intervention one can think of is to impose a Dirac distribution to one variable—thereby setting it to some fixed value. Other distributions can be “set” to any node in a DAG.

From these considerations, we aim to verify whether learning algorithms could infer simulated causal relationships, and in which settings. In other words: what can and cannot be learned from observational data in terms of causality in a “typical” biological system?

3 Bayesian Networks as a Framework for DAG Inference

Bayesian networks are a now widely used and powerful tool for probabilistic modeling. In this section, we introduce this kind of graphical models and discuss their important features.

3.1 Graphs and DAGs

A *graph*, G , is a mathematical construct consisting of a pair, $\langle V, E \rangle$, of a set of *nodes* (also known as *vertices*), V , and a set of *edges*, E . Each edge itself consists of a pair of nodes representing the endpoints of that particular edge. For example, an edge from node a to node b would be denoted (a, b) .

A *DAG*, or *Directed Acyclic Graph*, is a graph whose edges are directed; that is, an edge (a, b) goes *from a to b*, not both ways; and which contains no cycles, i.e., for any node $a \in V$, there exists no nontrivial path which both begins and ends at a .

If we consider each node as representing a particular event, and each edge as a conditional dependence, we begin to see how a DAG may be used as a graphical model. For example, consider a DAG with node set $\{(F = \text{You run out of fuel}), (L = \text{You're late})\}$, and edge set $\{(F, L)\}$. We then have a two-node DAG with one edge going from F to L.

3.2 d-Separation

The importance and use of d-separation will become obvious later, for now we simply provide the definition. Judea Pearl, considered by some to be the forefather of modern causal ideologies, defines d-separation as follows [17, p. 16–17]:

If X , Υ , and Z are disjoint subsets in a DAG D , then Z is said to *d-separate* X from Υ ...if along every path between a node in X and a node in Υ there is a node w satisfying one of the following conditions;

1. w has converging arrows, and none of w or its descendants are in Z
2. w does not have converging arrows and is in Z .

3.3 Probabilistic Independence

Two variables, X and Υ , are said to be *probabilistically independent* given a third variable Z , denoted $(X \perp\!\!\!\perp \Upsilon | Z)_P$, if

$$P(X, \Upsilon | Z) = P(X|Z)P(\Upsilon|Z) \quad (1)$$

Where

$$P(A|B) := \frac{P(A, B)}{P(B)}, P(B) \neq 0 \quad (2)$$

3.4 D-maps, i-maps, and Perfect Maps

A DAG, D , is an independency map, or *I-map*, of some probability distribution P if and only if d-separation of nodes in D implies probabilistic independence of the corresponding variables in P , that is:

$$(X \perp\!\!\!\perp \Upsilon | Z)_D \rightarrow (X \perp\!\!\!\perp \Upsilon | Z)_P \quad (3)$$

Conversely, a DAG, D is a dependency map, or *d-map*, if and only if

$$(X \perp\!\!\!\perp \Upsilon | Z)_D \leftarrow (X \perp\!\!\!\perp \Upsilon | Z)_P \quad (4)$$

A DAG, D , is a *perfect map* of P if and only if it is both a D-map and an I-map of P .

3.5 The Markov Blanket

The Markov blanket of a node, X , in a DAG, D , denoted $MB(X)$, is the minimal set of nodes conditioned on which X becomes independent of the rest of the nodes in D .

It can be shown [124, p. 120–121] that if D satisfies the Markov condition—each node of the DAG is conditionally independent of its non-descendants given its parent variables, $MB(X)$ consists of the parents, children, and children’s parents of X . $MB(D)$ is illustrated in Fig. 2.

3.6 Bayesian Network Definition

A Bayesian Network (BN) is a minimal I-map of some probability distribution P ; that is, a directed acyclic graph, D , whose node-wise graphical separations imply probabilistic independencies in P , and where any further edge removal from D will result in loss of its I-mapness [124, p. 119].

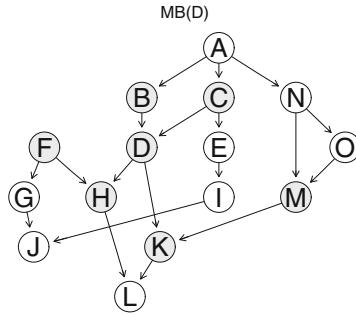


Fig. 2 The Markov blanket of node D

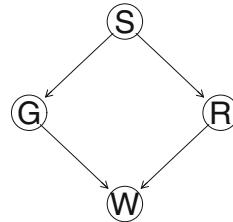


Fig. 3 An example Bayesian network

A Bayesian network, B , may be expressed as a pair $B = \langle D, \theta \rangle$ consisting of a DAG, D , and a set of parameters for the *conditional* probabilities of the nodes in D , denoted θ . Many extensions to the Bayesian network exist; for example, object-oriented Bayesian networks [125] can be used to form compound models for more complex situations when some elementary blocks are repeated and hierarchically assembled.

3.6.1 Why Are They Useful?

To begin, notice that in the definition for a BN, we only required θ to be the set of parameters for the *conditional* probabilities for each node, not the entire joint distribution. This in itself simplifies the expression of the full system. We also have that a BN satisfies the Markov Condition, and as such the maximum set of nodes that a node needs to be conditioned on is simply its Markov Blanket. This can drastically reduce the number of values required to fully specify the model.

3.7 Bayesian Network Learning: An Overview

A sample from a Bayesian network, B , consists of a realization of each node in B . For example, consider the Bayesian network in Fig. 3 as a classic example.

Let node S denote the event that it is summer, node R denote the event that rain is falling, node G that your garden sprinkler is active, and W that the grass is wet. A sample from this network could look something like:

{summer, rain, \neg sprinkler, wet}

Many algorithms exist to reproduce either the structure or probability distribution associated with a BN from a set of such samples, and can be summarized as in the following sections.

3.8 Structure Learning

Learning the structure of a BN is a complex task. The number of possible acceptable edge configurations grows super-exponentially as the number of nodes increases; for two-node networks, there are 3 possible valid DAG structures. For four nodes there are 543 structures, and for eight nodes there are $\approx 7.8 \times 10^{11}$ structures. For a reasonable 20-node network, there are $\approx 2.345 \times 10^{72}$ possible valid edge configurations [126]. This makes exploring the entire search space of plausible (and large enough to be useful) BNs an impractical exercise. Following are overviews, and examples, of the three main heuristic approaches intended to deal with this problem.

3.8.1 Constraint-Based Algorithms

The first general approach is using *constraint-based* algorithms, which are usually based on Pearl's *IC* (Inductive Causation) algorithm [17, p. 50]. With these, we start with a fully saturated, undirected graph and use conditional independence tests such as the log-likelihood G^2 or a modified Pearson's χ^2 test [2, p. 99] to remove edges one-by-one. Another pass then adds direction to edges where adequate information is available (see Pearl's algorithm [17, p. 50]). Examples of constraint-based algorithms include *IC* [17, p. 50], *Incremental Association Markov blanket (IAMB)* [127], and *PC* [123, p. 84].

3.8.2 Score-Based Algorithms

The other common approach is one which considers and scores the network as a whole, rather than one edge at a time. Initialize any network of your choosing, then possible future networks are evaluated by providing them with a score, such as the *Bayesian Dirichlet Equivalent uniform* (BDE), or the *Bayesian Information Criterion* (BIC) score [2, p. 106].

3.8.3 Hybrid Algorithms

Perhaps most commonly used are hybrid algorithms, which, as the name would suggest, are a combination of the two previous approaches. For example, the Sparse Candidate algorithm [128] uses tests for independence to restrict the search space for the subsequent score-based search. This general approach is known as the *restrict-maximize* heuristic and is implemented in bnlearn as `rsmax2`.

3.9 Parameter Learning

Once the structure of a network is known, the parameters to fit can be estimated relatively easily, using either; a frequentist maximum likelihood approach, whereby you view the global distribution as a function of the model parameters and do some optimization over the data; or by Bayesian methods [2, §1.4], where you update a prior belief about the parameters in light of the data [2, p. 11]. The first method, in the case of discrete Bayesian networks, reduces

to simply counting the occurrences of the data points relevant to each node and using the sample proportions to estimate the parameters. The Bayesian approach assigns a uniform prior over the entire conditional probability table, and then calculates the posterior distribution using the data present. Both methods are implemented in bnlearn in the `bn.fit` function.

3.9.1 Two Notes on Using Bayesian Methods

Firstly, the use of Bayesian methods may prove valuable when there is expert information available, as the uniform prior mentioned above does not necessarily need to be uniform—it may just as well represent the belief of an expert in the relevant field. To use the Bayesian approach in parameter learning, `method = "bayes"` must be included in the call to `bn.fit`. This allows for specification of the prior distribution. The user has some control over the weighting of the prior distribution with respect to the data when calculating the posterior distribution by means of the `iss` (imaginary sample size) argument.

Secondly, a discussing remark must be made on the scalability of the Bayesian network approach. Continuous efforts are made to improve the network size which can be handled. Just over a decade ago, one could tackle a few hundred nodes in a sparse network [129]. In the case of discrete data, the algorithm complexity is exponentially linked to the number of classes to represent factor variable levels. When the data is continuous, several choices are possible [130], from prior discretization to direct modeling, e.g., relying on kernels. In conjunction with discretization considerations, the maximum number of parents allowed per node constrains the capacity of the algorithm to run on a given data set [131, 132]. Beyond computational smart decisions, theoretical studies characterize algorithm complexity [133, 134] since [135]. This leads to the development of more efficient algorithms [136–138]. Sometimes, developments rely on parallel computation [139, 140]. Bayesian networks can be learnt with thousands of variables and few tens of thousands of edges using most current statistical packages. More specific applications can allow practitioners to perform computations with hundreds of thousands of nodes, e.g., to perform variable selection in databases [141], but this is not yet the case nor a need for biological networks, where a few tens of thousands of nodes is amply enough, and the limiting factor is then rather sample sizes.

3.10 Intervention

Intervening on a network and setting the value of a node has a notably different effect than simply “observing” it. Consider again the DAG in Fig. 3, with semantics as defined in Subheading 3.7.

Were an external deity to intervene on the scenario and turn on the sprinkler, whether or not the sprinkler is on becomes independent of what season it is (as it has been forced on), and the DAG structure changes to that in Fig. 4.

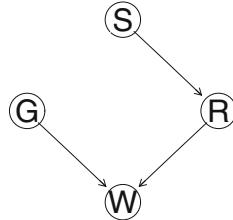


Fig. 4 The Bayesian network from Fig. 3 under the intervention $do(G=g)$

In this way, intervention renders a node independent of its parents, and the probability distribution underlying the DAG also gets altered. The modification (1) deletes the conditional probability of the nodes intervened on and (2) sets their value when they are parents of other nodes in the relevant conditional probabilities. Notice that interventions do not necessarily fix the value of a node to a given value, but could impose a particular distribution. It should be fairly clear, then, that you cannot intervene on a set of samples, as they have already been drawn from a particular distribution which it is now too late to alter. For example, if we have a set of samples from the DAG in Fig. 3, and we now want to know what would happen if we turn the sprinkler on (regardless of the season), we cannot do so without further experiment or more information about causality in the system.

Causal discovery in observational studies is mainly based on three axioms which bind the causality terminology to probabilistic distributions. The first one is the causal Markov condition. It states that once all direct causes of an event are known, the event is (conditionally) independent of its causal non-descendants. The second one is the faithfulness condition; it ensures that all dependencies observed in the data are structural, i.e., were generated by the structure of an underlying causal graph. These (in)dependencies are not the result of some canceling combination of model parameters, an event which often has a zero probability measure with commonly used distributions [123]. The last condition is termed causal sufficiency assumption; all the common causes of the measured variables are observed. When this is not the case, hidden variable methods can help [142].

3.11 Markov Equivalence

Consider a two-node network, D , with nodes A and B , and one edge from A to B , i.e., $A \rightarrow B$. With a trivial application of Bayes' rule, we observe:

$$P(A, B) = P(A)P(B|A) \quad (5)$$

$$= P(A) \frac{P(A|B)P(B)}{P(A)} \quad (6)$$

$$= P(B)P(A|B) \quad (7)$$

By (5) and (7), $P(A)P(B|A) = P(B)P(A|B)$ and the network is equivalent to $B \rightarrow A$.

Clearly, some visually nonidentical networks can encode the same information and dependency structures, and samples from them will be indistinguishable. Such networks are known as *Markov Equivalent* networks. For example, [143] used an MCMC computational approach to identify the equivalence class of a DAG. Calculating the values of $P(B)$ and $P(A|B)$ given $P(A)$ and $P(B|A)$ may be done as follows:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \quad (8)$$

$$P(B) = \sum_A P(B|A)P(A) \quad (9)$$

4 Experimental Setup: Quality of a Causal Network Reconstructed from Observational Data

The aim of our experiment is to investigate the similarity between the BN used to generate samples, and the BN learned from the data, as we varied the number of samples used in the process. This similarity was measured using the Structural Intervention Distance (SID) [144]. We provide the code for our experiments at <https://github.com/alexW335/BNCausalExperiments/blob/master/SID.R>.

4.1 Method

To be able to more easily investigate the problem at hand, we decided to use a small, 7-node BN as shown in Fig. 5.

This particular network was chosen as it possesses many substructures of particular interest when investigating causality, for example, the v-structure (or collider) in Fig. 6a, the (causal) chain as shown in Fig. 6b, and the fork (or common cause) seen in Fig. 6c.

The conditional probabilities associated with each node were chosen to show distinct separation in the data, e.g., $P(B = b|A = a)$ should be sufficiently different to $P(B = b|A = \neg a)$ for ease of

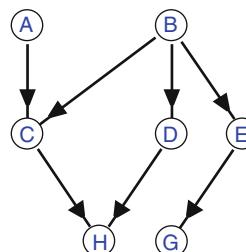


Fig. 5 Data generating 7-node DAG structure

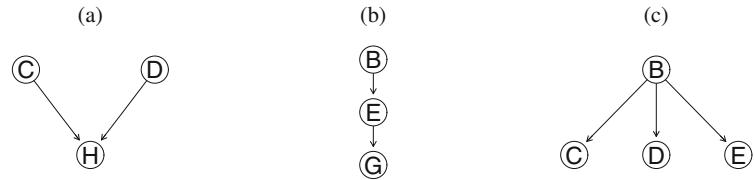


Fig. 6 Common structures of interest. **(a)** V-Structure, **(b)** Chain, **(c)** Fork

DAG-edge recovery, i.e., we seek high model identifiability (related to faithfulness of the model). These conditional probabilities are available in the source code.

4.1.1 Data Generation

Samples were generated on demand with the R function `rbn` [2] from the package `bnlearn`.

4.1.2 Network Reconstruction

The two-phase restrict–maximize heuristic was used to recover the DAG structure from the generated data. The semi-parametric χ^2 test was used to check for conditional independencies, the Inter-Associative Markov Blanket (IAMB) algorithm [127] was used in the restriction phase for locally minimizing the search space, a straightforward hill-climbing algorithm was used for exploring the restricted space, and the Bayesian Information Criterion (BIC) score [145] was used as the score by which to compare the possible networks during the search.

4.1.3 Initial Notes on BN Comparison

Initially, two measures were considered for comparing the quality and similarity of the reconstructed Bayesian network, $B^* = \langle D^*, \theta^* \rangle$, to the original Bayesian network $B = \langle D, \theta \rangle$:

1. **Graph Edit Distance** The graph edit distance between D and D^* , where one edit is either an edge *addition*, *deletion*, or *reversal*. This compares the structural similarities of the DAGs but fails to account for any differences in the underlying distribution.
2. **Kullback–Leibler (KL) Divergence** KL divergence from B to B^* may be used to compare the underlying probability distributions of the two BNs. As prior information about the structure of the original network is available in an experimental setting such as this, it is possible to condition the calculation of the KL divergence on a particular node, say D ; that is, enumerate

$$\sum_A \sum_B \sum_C (P(D|A, B, C)) \quad (10)$$

rather than

$$\sum_A \sum_B \sum_C \sum_D (P(A, B, C, D)) \quad (11)$$

Where \sum_X means sum over all possible states of X , e.g., $\{x, \neg x\}$.

This would give an idea of how accurately the probability distribution was being reproduced, and can be used to give some insight into the causal structure underlying the BN in question.

However, a few issues arise from using these measures. The KL divergence implementation proves particularly difficult to generalize, and the conditioning on some node D requires prior knowledge that the specific node selected would be the best node to look for downflow effects.

4.1.4 Bayesian Network Comparison with SID

One metric was used to compare the two models—the Structural Intervention Distance [144]. This does not consider the probability distributions underlying the BNs in such a way that, for example, the Kullback–Leibler divergence does; it rather contains information about the difference between both structures under the same intervention. This allows some information about the causal structure to be inferred, such as the location of v-structures and forks (see Fig. 6). In a nutshell, SID measures the number of interventions which lead to different graphical structures between two DAGs. The smaller the SID, the closer the two DAG structures.

4.2 Results

4.2.1 Seven-Node Network

As expected, the average SID appears to decrease as the number of samples used to generate the network increased, as per Fig. 7.

The data shown in Fig. 7 can be well modeled using a Generalized Additive Model (GAM), though the usefulness of such a model is debatable. It is assumed that the SID should asymptotically approach zero (or is at least nonnegative), so any model fitted should also exhibit this behavior to prove itself a useful predictive tool. This aside, the diagnostic plots for the GAM may be found in Fig. 8 to convince the reader that the model is at least naively fitting the observed data well.

Another factor of interest is how the confidence in the edges (both true edges, and those which should not appear) grows with the number of samples used to generate the network. This is plotted in Fig. 9 where the confidence at each level of sample count is expressed as the proportion of the 1000 BNs generated in which each edge appears. As one would expect, the 7 true edges tend to increase in confidence as a function of the number of samples, and the remaining 35 edges which are not present in the true BN appear to fluctuate randomly close to zero.

The same data used to produce Fig. 9 can be used to construct the average adjacency matrix, and thus the average DAG, produced at each sample count. The average DAG for 100 samples and 2500 samples are shown in Fig. 10a and b, respectively.

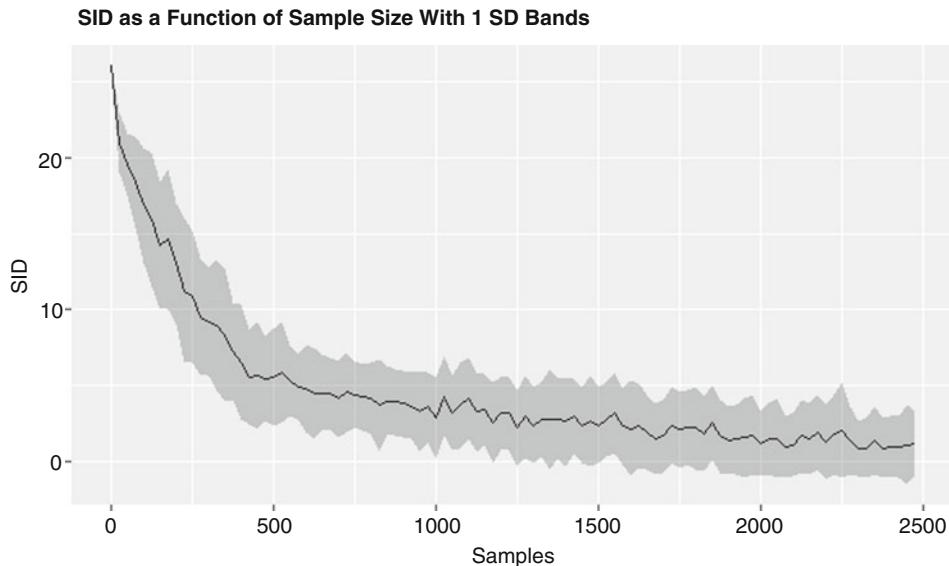


Fig. 7 Mean SID between the data generating Bayesian network and a 1000 BNs reconstructed from the samples, plotted as a function of the number of samples used to reconstruct the BN. A gray band of ± 1 standard deviation for the mean SID is shown

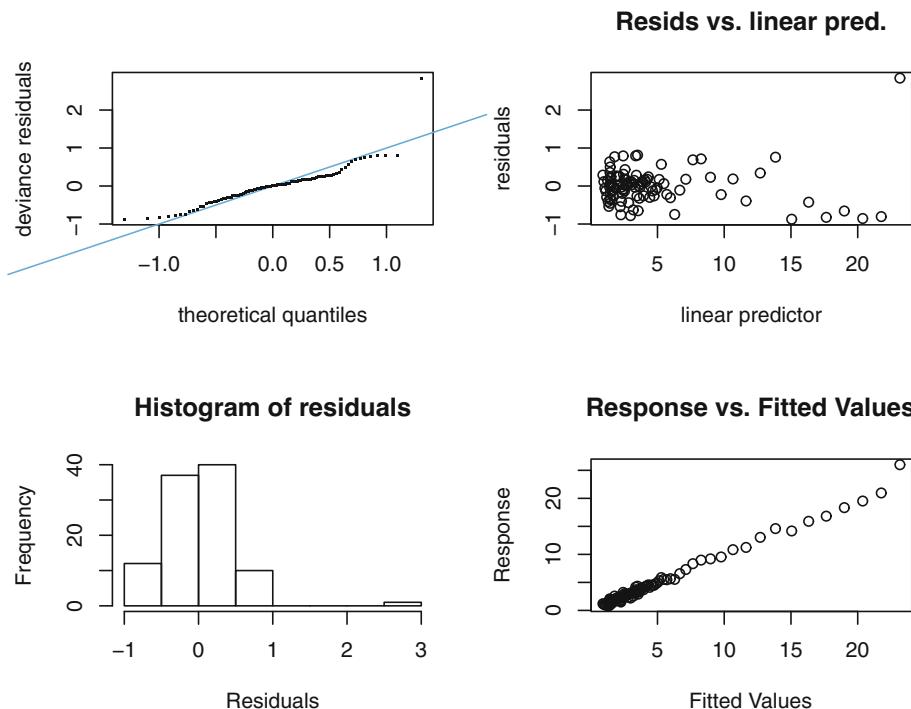


Fig. 8 Diagnostic plots for the GAM

Confidence in Edges by Number of Samples

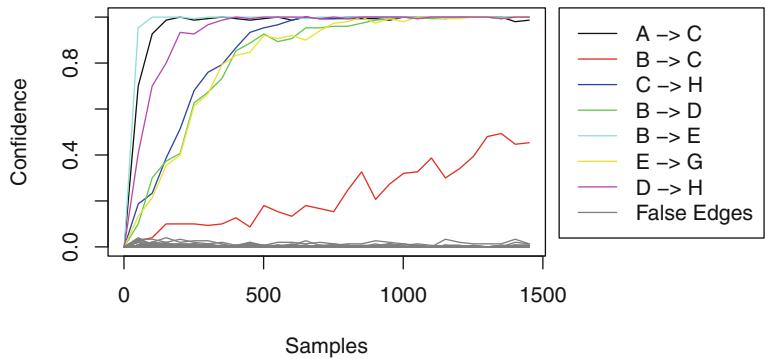
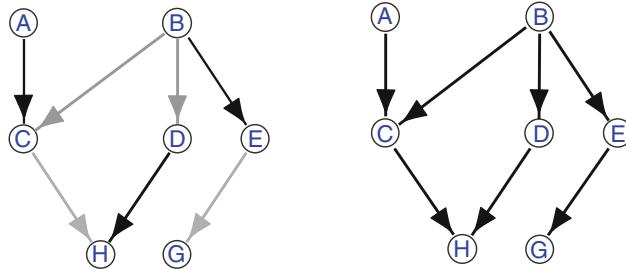


Fig. 9 Proportion of graphs generated from each number of samples which contain specific edges. The labeled edges are the ones which are known to be present in the actual BN



(a) Average DAG Generated from 100 Samples (b) Average DAG Generated from 2500 Samples

Fig. 10 Average graphs generated from different numbers of samples. Note that the edge coloring has been passed through a three-stage threshold: edges appearing in less than 5% of DAGs were not shown, those appearing in 5–50% were shown in gray, and those in over 50% were shown in black. (a) Average DAG generated from 100 samples. (b) Average DAG generated from 2500 samples

The adjacency matrices for the average DAGs provide more information than the simple graphical view; however, they can be a little harder to decode as is evident in Fig. 11.

4.2.2 Four-Node Network

As well as the illustrative seven-node network, the experiments were run on the Bayesian network whose topology is shown in Fig. 3, with conditional probabilities as represented in Fig. 12. Note that this network is unrelated to the similarly structured weather network of Subheading 3.7.

This yielded a curve of SID by samples as shown in Fig. 13, displaying markedly more variability than the seven-node case.

This variability is echoed in the edge confidence graph as shown in Fig. 14, where we can see that an incorrect edge was

	A	B	C	D	E	G	H
A	0.000	0.010	0.918	0.003	0.001	0.008	0.005
B	0.000	0.000	0.054	0.232	0.998	0.017	0.004
C	0.004	0.001	0.000	0.007	0.001	0.009	0.243
D	0.000	0.000	0.005	0.000	0.000	0.011	0.680
E	0.004	0.001	0.004	0.010	0.000	0.226	0.001
G	0.001	0.000	0.002	0.001	0.025	0.000	0.007
H	0.000	0.001	0.017	0.018	0.001	0.004	0.000

Fig. 11 The average adjacency matrix for DAGs generated from 100 samples. This corresponds to Fig. 10

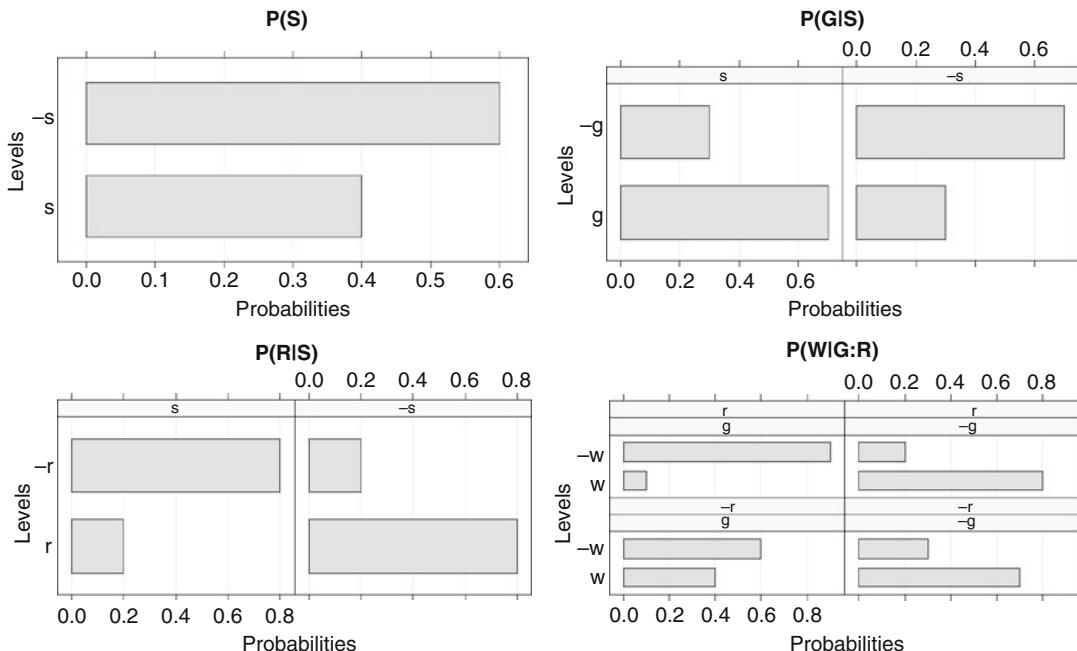


Fig. 12 The conditional probability tables for the nodes of the BN used in the second experiment

predicted with higher frequency than one of the actual edges for low to medium sample counts.

An interesting observation is that the collider

$$G \rightarrow W \leftarrow R$$

is not picked up easily. If incorrectly constructed, any alterations to the collider would have relatively high effects on the SID, as intervention on the central node has a large impact on the interventional distribution. This could explain the variability seen in Fig. 13.

4.2.3 A Real Signaling and Regulatory Gene Network

As an illustration, we applied the presented methodology to a real biological network. We chose the Sachs et al. [58] 11-node network. We keep the same nomenclature for genes as the one used

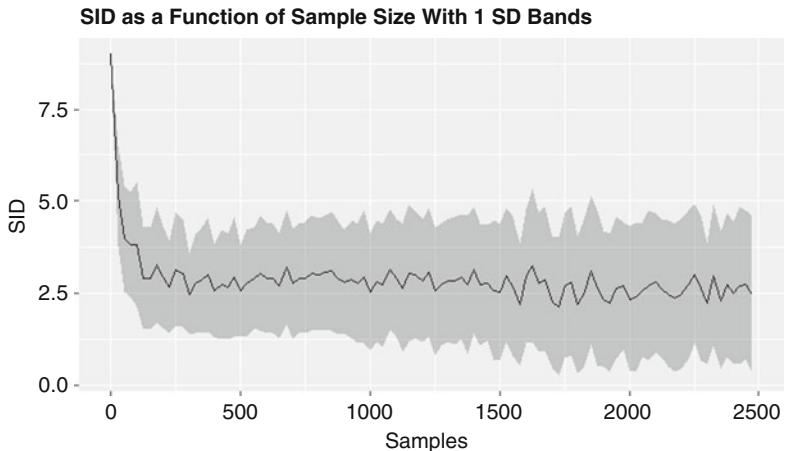


Fig. 13 Mean SID between the data generating Bayesian network and a 1000 BNs reconstructed from the samples, plotted as a function of the number of samples used to reconstruct the BN. A gray band of ± 1 standard deviation for the mean SID is shown

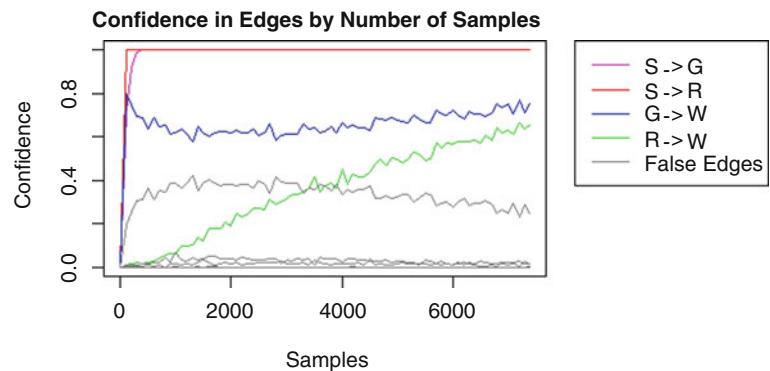


Fig. 14 Edge confidence for four-node network as a function of the number of samples used to generate the BN

in the initial paper, except for $\text{Plc}\gamma$, which is denoted Plcg here. We only used the 853 samples without intervention. The network which we considered as the reference network for SID comparison is depicted in Fig. 17. This network is ideal for our purpose as it has a large number of available samples, and does not include cycles. In addition, it has intervention data sets which could be used to refine the model. Results are presented in Figs. 15, 16, and 17.

The SID performance evolution in Fig. 15 can seem a bit disappointing. This seemingly poor performance stems from the fact that as many edges are missing from the reconstructed network, the resulting graph finally reaches three disconnected components (PIP2-PIP3-Plcg, Raf-Mek, and PKC-PKA-Jnk-P38-Akt-Erk). These disconnected components leave many pairs for which the interventional distribution is not adequately estimated,

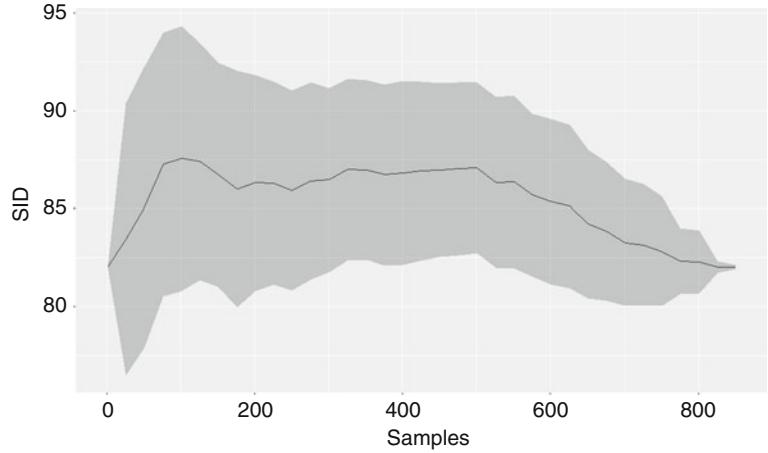


Fig. 15 Mean SID between the reference Sachs network and 300 BNs reconstructed from the samples, plotted as a function of the number of samples used to reconstruct the BN. 853 is the maximum available sample size. A gray band of ± 1 standard deviation for the mean SID is shown

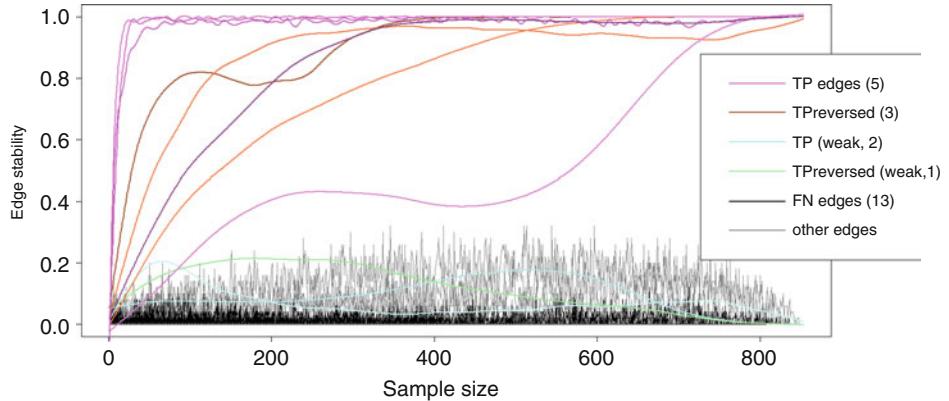


Fig. 16 Sachs edge confidence as a function of sample size. Edge colors match those of Fig. 17

and hence increase SID [144]. We see that adding the first edges even increases the computed SID. Forming a more complete sketch of the reference network helps to decrease the SID with increasing sample size. Notice that the empty network (predicted with no data) and the final 8-node network have the same SID measure, 84, although the predicted network is much more satisfying as it comprises 8 edges: five true positives and 3 reversed edges.

Figure 16 presents the edge confidence as a function of sample size. All true positive edges are identified very quickly (with less than 50 samples), except one ($\text{PIP3} \rightarrow \text{Plcg}$) which requires more samples (more than 100) to stand out, becoming stable with more than 500 samples. This may be because it competes with another reversed edge (the green lines in Fig. 16). Furthermore,

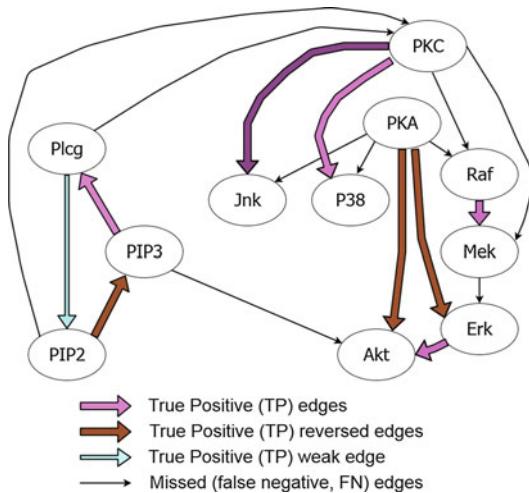


Fig. 17 Final predicted network with our causal Bayesian network approach. Violet edges are correctly identified, pale blue are those with weak confidence, brown edges are those which were learned in the reverse direction compared to our reference network, and the thin black edges are those which were missed. Notice that one weak true positive edge and one weak, reversed, true positive edge are not represented (see Fig. 16) as they overlap edges represented in the opposite direction

it is not clear whether this edge is indeed in this direction or reversed (see discussion in [58]). Another predicted edge assigned the wrong direction is also competing with the correct edge as per the reference network ($\text{PIP3} \rightarrow \text{PIP2}$), and this was reported as bidirectional in the literature study of [58]. Note that the local $\text{PIP2} \rightarrow \text{PIP3} \rightarrow \text{Plcg} \rightarrow \text{PIP2}$ loop does not contradict the DAG assumption, as this representation is that of the average graph created at that number of samples. The true positive edge with weak confidence is not part of the finally inferred network. When it is in the model, $\text{PIP2} \rightarrow \text{PIP3}$ is either not present or reversed (hence in the correct direction, as per discussion above). The confidence of missed edges does not stand out from the background noise, i.e., the nonrelevant edges or true negatives. Either the local encoded dependency is already captured by the 8 edges, or it requires interventional data to precisely identify them.

Five edges were correctly retrieved, and three additional edges were retrieved in the opposite direction compared to our reference network. This leads to a directed precision of 62.5% (undirected precision 100%) and a recall of 28% (undirected recall 44%). To be noted is that no false positive edge is predicted. No interventional data was used, while [58] used all the 5400 available samples with several subsets (6) of interventions.

5 Conclusion/Discussion

In this chapter, we discussed networks as a powerful approach to modeling complex living systems. We focused on gene regulatory networks, and their reconstruction from observational (aka wild-type) gene transcript data. We reviewed some methods dealing with the concept of causality, and we specifically focused on Bayesian networks. We introduced BNs in detail, and tested their capacity to retrieve causal links on two toy network examples. Unexpectedly, the two experiments lead to different conclusions. While the learning inference went quite well from a causal relationship perspective for the seven-node network, it showed mixed performance in the four-node network. We were not able to see any trend in the classification of directed edges which are correctly predicted versus those which are not correctly predicted. In the four-node network, edges originating from the source node S were detected with small sample size without any ambiguity, while it would not have been surprising if the learning algorithm had inverted one and not the other (to avoid creating a new V-structure). In addition, the converging V-structure to the sink node W was much more difficult to retrieve despite relatively large sample sizes (up to 5000 samples) and contrasting simulated distributions. The conclusion is quite different in our seven-node network, with all edges but one retrieved without ambiguity with more than 500 samples. Again, the most difficult directed edge to retrieve was one involved in a V-structure, this time from a source node (B). It also required fewer samples than the four-node network, while much more structures are possible with 7 nodes. At the same time, this seven-node network contains more conditional independence than the four-node network. Another surprising result is that edges which could have been inferred in one direction or the other ($B \rightarrow D$ could be inverted, or the directed path $B \rightarrow E \rightarrow G$ could be inverted; not both at the same time) were assigned a direction without ambiguity. The most surprising conclusion so far in our experiments is that increasing the sample size seemed to always improve the identification of edge, including its direction, whether it is imposed in the equivalence graph or not. This could be due to not studying enough graphical configurations. This conclusion must also be qualified in that obtaining a few hundred samples for a small 10-node example is not very realistic from an experimental point of view.

5.1 Future Work

Obviously, we only showed very limited experiments in very special cases. For our conclusions to be valid, we would need to extend our experimental setup to: (1) a bigger variety of conditional distributions, (2) more local network motifs, e.g., feed-forward loops, (3) question the binary variable framework, and (4) test

other learning inference methods. It would be interesting to test point (1) by checking some well-chosen edges for their prediction confidence as a function of the conditional probabilities, with a similar representation to that in Fig. 9, but a variation on the conditional distribution(s) on the x -axis instead of the sample size. However, varying a conditional distribution with one parameter in one dimension is technically challenging and necessarily not perfect. For point (2), a next step would be to build a dictionary gathering the typology of local structures which can be encountered in networks and how well a BN learning algorithm could perform at retrieving directed edges on those structures. For point (4), we and other colleagues (Geurts P and Huynh-Thu VA, 2013, personal communication) noted, for example, that random forests seem to give relevant directions to edges when used to infer GRNs (*see* [146]).

When performing tests to be presented in this chapter, we also used the graph edit and the KL divergence as potential measure of closeness between the true network and the predicted network. It is known that the graph edit distance is not ideal as two graphs with only one edge difference can lead to quite different causal conclusions and/or independence relationships. The KL divergence proved to be useful to compare probability distributions, yet prohibitively difficult to generalize. Combining KL-divergence with the SID metric could provide an alternative representation of the ability of the learning algorithm to retrieve the distribution—if not the causal relationships. An easily generalized metric combining both, or some variation of the two, could be of particular interest.

Acknowledgements

Alex White was partly supported by a Summer Scholarship grant from the Institute of Fundamental Sciences at Massey University (NZ). This work was also supported by a visiting professor scholarship from Aix-Marseille University granted to Matthieu Vignes in 2017. The material in this chapter slowly hatched after many discussions with a multitude of quantitative field colleagues and biologists during some works we were involved in, e.g., [34, 131, 147], but most of our inspiration stems from other works we read about and discussed (e.g., [60, 113, 146, 148, 149] to cite only a few). We are very appreciative of the many discussions with Stephen Marsland (Victoria University of Wellington, NZ). Lastly, we are very grateful to the reviewers of this chapter for their insightful comments.

References

1. Scutari M (2010) Learning Bayesian Networks with the bnlearn R Package. *J Stat Softw* 35(3):1–22
2. Scutari M, Denis JB (2014) Bayesian networks: with examples in R. Texts in statistical science. CRC Press: Taylor & Francis Group, Boca Raton
3. Edwards JS, Palsson BO (1999) Systems properties of the *Haemophilus influenzae* Rd metabolic genotype. *J Biol Chem* 274(25):17410–17416
4. Kitano H (2002) Systems biology: a brief overview. *Science* 295(2):1662–1664
5. Noble D (2006) The music of life: biology beyond genes. Oxford University Press, Oxford
6. Barabási AL, Oltvai ZN (2004) Network biology: understanding the cell's functional organization. *Nat Rev Genet* 5(2):101–113
7. Gericke NM, Hagberg M (2007) Definition of historical models of gene function and their relation to students' understanding of genetics. *Sci Educ* 16(7):849–881
8. Pennisi E (2007) DNA study forces rethink of what it means to be a gene. *Science* 316(5831):1556–1557
9. McElreath R (2015) Statistical rethinking: a Bayesian course with examples in R and Stan. Chapman and Hall/CRC, Boca Raton
10. Scutari M, Howell P, Balding DJ, Mackay I (2014) Multiple quantitative trait analysis using Bayesian networks. *Genetics* 198(1):129–137
11. Zhang B, Horvath S (2005) A general framework for weighted gene co-expression network analysis. *Stat Appl Genet Mol Biol* 4:17
12. Tenenhaus A, Guillemot V, Gidrol X, Frouin V (2010) Gene association networks from microarray data using a regularized estimation of partial correlation based on PLS regression. *IEEE/ACM Trans Comput Biol Bioinform* 7(2):251–262
13. Rau A, Maugis-Rabusseau C, Martin-Magniette ML, Celeux G (2015) Co-expression analysis of high-throughput transcriptome sequencing data with Poisson mixture models. *Bioinformatics* 31(9):1420–1427
14. Jacob F, Monod J (1961) Genetic regulatory mechanisms in the synthesis of proteins. *J Mol Biol* 3(3):318–356
15. Hentschel U, Steinert M, Hacker J (2000) Common molecular mechanisms of symbiosis and pathogenesis. *Trends Microbiol* 8(5):226–231
16. Dupont PY, Eaton CJ, Wargent JJ, Fechner S, Solomon P, Schmid J, Day RC, Scott B, Cox MP (2015) Fungal endophyte infection of ryegrass reprograms host metabolism and alters development. *New Phytol* 208(4):1227–1240
17. Pearl J (2009) Causality: models, reasoning and inference, 2nd edn. Cambridge University Press, New York
18. Mangan S, Alon U (2003) Structure and function of the feed-forward loop network motif. *Proc Natl Acad Sci* 100(21):11980–11985
19. Zabet NR (2011) Negative feedback and physical limits of genes. *J Theor Biol* 284(1):82–91
20. Shojaie A, Jauhainen A, Kallitsis M, Michailidis G (2014) Inferring regulatory networks by combining perturbation screens and steady state gene expression profiles. *PLoS ONE* 9:1–16
21. Ghahramani Z (1998) Learning dynamic Bayesian networks. In: Adaptive processing of sequences and data structures. Lecture notes in computer sciences. Springer, New York, pp 168–197
22. Friedman N, Murphy K, Russell S (1998) Learning the structure of dynamic probabilistic networks. In: Proceedings of the fourteenth conference on uncertainty in artificial intelligence, UAI'98. Morgan Kaufmann, San Francisco, pp 139–147
23. Husmeier D (2003) Sensitivity and specificity of inferring genetic regulatory interactions from microarray experiments with dynamic Bayesian networks. *Bioinformatics* 19(17):2271–2282
24. Tulupiyev AL, Nikolenko SI (2005) Directed cycles in Bayesian belief networks: probabilistic semantics and consistency checking complexity. In: Gelbukh A, de Albornoz Á, Terashima-Marín H (eds) MICAI 2005: Advances in artificial intelligence. Springer, Berlin, pp 214–223
25. Harary F, Norman R, Cartwright D (1965) Structural models: an introduction to the theory of directed graphs. Wiley, New York

26. Lacerda G, Spirtes P, Ramsey J, Hoyer P (2008) Discovering cyclic causal models by independent components analysis. In: Proceedings of the twenty-fourth conference annual conference on uncertainty in artificial intelligence (UAI-08). AUAI Press, Corvallis, pp 366–374
27. Quackenbush J (2007) Extracting biology from high-dimensional biological data. *J Exp Biol* 210(9):1507–1517
28. Bühlmann P, van de Geer S (2011) Statistics for high-dimensional data – methods, theory and applications. Springer, Berlin
29. Verzelen N (2012) Minimax risks for sparse regressions: ultra-high dimensional phenomenons. *Electron J Stat.* 6:38–90
30. Giraud C (2014) Introduction to high-dimensional statistics. Chapman & Hall/CRC, New York
31. Oates CJ, Dondelinger F, Bayani N, Korkola J, Gray JW, Mukherjee S (2014) Causal network inference using biochemical kinetics. *Bioinformatics* 30(17):i468–i474
32. Shojaie A, Michailidis G (2010) Discovering graphical Granger causality using the truncating lasso penalty. *Bioinformatics* 26(18):i517–i523
33. Rau A, Jaffrézic F, Foulley JL, Doerge RW (2010) An empirical Bayesian method for estimating biological networks from temporal microarray data. *Stat Appl Genet Mol Biol* 9:1
34. Marchand G, Huynh-Thu VA, Kane NC, Arribat S, Varès D, Rengel D, Balzergue S, Rieseberg LH, Vincourt P, Geurts P, Vignes M, Langlade NB (2014) Bridging physiological and evolutionary time-scales in a gene regulatory network. *New Phytol* 203(2):685–696
35. Chandrasekaran V, Parrilo PA, Willsky AS (2012) Latent variable graphical model selection via convex optimization. *Ann Stat* 40(4):1935–1967
36. Blanchet J, Vignes M (2009) A model-based approach to gene clustering with missing observation reconstruction in a Markov random field framework. *J Comput Biol* 16(3):475–486
37. Colombo D, Maathuis MH, Kalisch M, Richardson TS (2012) Learning high-dimensional directed acyclic graphs with latent and selection variables. *Ann Stat* 40(1):294–321
38. Fusi N, Stegle O, Lawrence ND (2012) Joint modelling of confounding factors and prominent genetic regulators provides increased accuracy in genetical genomics studies. *PLoS Comput Biol* 8(1):1–9
39. Sadeh MJ, Moffa G, Spang R (2013) Considering unknown unknowns: reconstruction of nonconfoundable causal relations in biological networks. *Bayesian Anal* 11(20):920–932
40. Mooij JM, Janzing D, Heskes T, Schölkopf B (2011) On causal discovery with cyclic additive noise models. In: Shawe-Taylor J, Zemel RS, Bartlett PL, Pereira F, Weinberger KQ (eds) Advances in neural information processing systems, vol 24. Curran Associates Inc., Red Hook, pp 639–647
41. de Jong H (2004) Modeling and simulation of genetic regulatory systems: a literature review. *J Comput Biol* 9(1):67–103
42. Markowetz F, Spang R (2007) Inferring cellular networks – a review. *BMC Bioinf* 8(6):S5
43. Lee WP, Tzou WS (2009) Computational methods for discovering gene networks from expression data. *Brief Bioinform* 10(4):408–423
44. Emmert-Streib F, Glazko G, Göökmen A, De Matos Simoes R (2012) Statistical inference and reverse engineering of gene regulatory networks from observational expression data. *Front Genet* 3:8
45. Maathuis MH, Kalisch M, Bühlmann P (2009) Estimating high-dimensional intervention effects from observational data. *Ann Stat* 37(6A):3133–3164
46. Oates CJ, Mukherjee S (2012) Network inference and biological dynamics. *Ann Appl Stat* 6(3):1209–1235
47. Fu F, Zhou Q (2013) Learning sparse causal Gaussian networks with experimental intervention: regularization and coordinate descent. *J Am Stat Assoc* 108(501):288–300
48. Werhli AV, Grzegorczyk M, Husmeier D (2006) Comparative evaluation of reverse engineering gene regulatory networks with relevance networks, graphical Gaussian models and Bayesian networks. *Bioinformatics* 22(20):2523–2531
49. Altay G, Emmert-Streib F (2010) Revealing differences in gene network inference algorithms on the network level by ensemble methods. *Bioinformatics* 26(14):1738–1744

50. Emmert-Streib F, Altay G (2010) Local network-based measures to assess the inferability of different regulatory networks. *IET Syst Biol* 4:277–288
51. Marbach D, Prill RJ, Schaffter T, Mattiussi C, Floreano D, Stolovitzky G (2010) Revealing strengths and weaknesses of methods for gene network inference. *Proc Natl Acad Sci* 107(14):6286–6291
52. Marbach D, Costello JC, Küffner R, Vega NM, Prill RJ, Camacho DM, Allison KR, Consortium TD, Kellis M, Collins JJ, Stolovitzky G (2014) Wisdom of crowds for robust gene network inference. *Nat Methods* 9(8):796–804
53. Meyer P, Cokelaer T, Chandran D, Kim KH, Loh PR, Tucker G, Lipson M, Berger B, Kreutz C, Raue A, Steiert B, Timmer J, Bilal E, Sauro HM, Stolovitzky G, Saez-Rodriguez J (2014) Network topology and parameter estimation: from experimental design methods to gene regulatory network kinetics using a community based approach. *BMC Syst Biol* 8(1):13
54. Hill SM, Heiser LM, Cokelaer T, Unger M, Nesser NK, Carlin DE, Zhang Y, Sokolov A, Paull EO, Wong CK, Graim K, Bivol A, Wang H, Zhu F, Afsari B, Danilova LV, Favorov AV, Lee WS, Taylor D, Hu CW, Long BL, Noren DP, Bisberg AJ, HPN-DREAM Consortium, Mills GB, Gray JW, Kellen M, Norman T, Friend S, Qutub AA, Fertig EJ, Guan Y, Song M, Stuart JM, Spellman PT, Koepl H, Stolovitzky G, Saez-Rodriguez J, Mukherjee S (2016) Inferring causal molecular networks: empirical assessment through a community-based effort. *Nat Methods* 13(4):310–318
55. Allouche D, Cierco-Ayrolles C, de Givry S, Guillermín G, Mangin B, Schiex T, Vandel J, Vignes M (2013) A panel of learning methods for the reconstruction of gene regulatory networks in a systems genetics context. Springer, Berlin, pp 9–31
56. Bontempi G, Haibe-Kains B, Desmedt C, Sotiriou C, Quackenbush J (2011) Multiple-input multiple-output causal strategies for gene selection. *BMC Bioinf* 12(1):458
57. Engelmann JC, Amann T, Ott-Rtzter B, Ntzel M, Reinders Y, Reinders J, Thasler WE, Kristl T, Teufel A, Huber CG, Oefner PJ, Spang R, Hellerbrand C (2015) Causal modeling of cancer-stromal communication identifies PAPPA as a novel stroma-secreted factor activating NF κ B signaling in hepatocellular carcinoma. *PLoS Comput Biol* 11(5):1–22
58. Sachs K, Perez O, Pe'er D, Lauffenburger DA, Nolan GP (2005) Causal protein-signaling networks derived from multiparameter single-cell data. *Science* 308(5721):523–529
59. Ness RO, Sachs K, Vitek O (2016) From correlation to causality: Statistical approaches to learning regulatory relationships in large-scale biomolecular investigations. *J Proteome Res* 15(3):683–690
60. Gagneur J, Stegle O, Zhu C, Jakob P, Tekkedil MM, Aiyar RS, Schuon AK, Pe'er D, Steinmetz LM (2013) Genotype-environment interactions reveal causal pathways that mediate genetic effects on phenotype. *PLoS Genet* 9(9):e1003803
61. Maathuis MH, Colombo D, Kalisch M, Bühlmann P (2012) Predicting causal effects in large-scale systems from observational data. *Nat Methods* 7:47–48
62. Taruttis F, Spang R, Engelmann JC (2015) A statistical approach to virtual cellular experiments: improved causal discovery using accumulation IDA (aida). *Bioinformatics* 31(23):3807–3814
63. Michailidis G, d'Alché Buc F (2013) Autoregressive models for gene regulatory network inference: sparsity, stability and causality issues. *Math Biosci* 246(2):326–334
64. Werhli AV, Husmeier D (2007) Reconstructing gene regulatory networks with Bayesian networks by combining expression data with multiple sources of prior knowledge. *Stat Appl Genet Mol Biol* 6:1
65. Mordelet F, Vert JP (2008) SIRENE: supervised inference of regulatory networks. *Bioinformatics* 24(16):i76–i82
66. Eberhardt F, Glymour C, Scheines R (2005) On the number of experiments sufficient and in the worst case necessary to identify all causal relations among n variables. In: Proceedings of the twenty-first conference on uncertainty in artificial intelligence, UAI'05. AUAI Press, Arlington, pp 178–184
67. Hauser A, Bühlmann P (2014) Two optimal strategies for active learning of causal models from interventional data. *Int J Approx Reason* 55(4):926–939. Special issue on the sixth European Workshop on Probabilistic Graphical Models
68. Meinshausen N, Hauser A, Mooij JM, Peters J, Versteeg P, Bühlmann P (2016) Methods

- for causal inference from gene perturbation experiments and validation. *Proc Natl Acad Sci* 113(27):7361–7368
69. Mooij JM, Peters J, Janzing D, Zscheischler J, Schölkopf B (2016) Distinguishing cause from effect using observational data: methods and benchmarks. *J Mach Learn Res* 17(32):1–102
70. Athey S, Imbens G (2016) Recursive partitioning for heterogeneous causal effects. *Proc Natl Acad Sci USA* 113(27):7353–7360
71. Chen G, Larsen P, Almasri E, Dai Y (2008) Rank-based edge reconstruction for scale-free genetic regulatory networks. *BMC Bioinf* 9(1):75
72. Agrawal H (2002) Extreme self-organization in networks constructed from gene expression data. *Phys Rev Lett* 89:268702
73. Opgen-Rhein R, Strimmer K (2007) From correlation to causation networks: a simple approximate learning algorithm and its application to high-dimensional plant gene expression data. *BMC Syst Biol* 1(1):37
74. Xiong M, Li J, Fang X (2004) Identification of genetic networks. *Genetics* 166(2):1037–1052
75. Friedman N, Linial M, Nachman I, Pe'er D (2000) Using Bayesian networks to analyze expression data. *J Comput Biol* 7(3–4):601–620
76. Spirtes P (2005) Graphical models, causal inference, and econometric models. *J Econ Methodol* 12(1):3–34
77. Pearl J (2009) Causal inference in statistics: an overview. *Stat Surv* 3(1):96–146
78. Hume D (1738–1740) A treatise of human nature. John Noon, London
79. Wright S (1921) Correlation and causation. *J Agric Res* 20(7):557–585
80. Neyman J (1990) On the application of probability theory to agricultural experiments. Essay on principles. Section 9 (translated and edited by d. m. dabrowska and t. p. speed from the polish original, which appeared in roczniki nauk rolniczych tom x (1923) 1–51 (annals of agricultural science)). *Stat Sci* 5(4):465–472
81. Fisher RA (1925) Statistical methods for research workers. Oliver & Boyd, Edinburgh,
82. Rubin D (1974) Estimating causal effects of treatments in randomized and non-randomized studies. *J Educ Psychol* 66:688–701
83. Holland PW (1986) Statistics and causal inference. *J Am Stat Assoc* 81(396):945–960
84. Wainer H (2014) Visual revelations: happiness and causal inference. *Chance* 27(4):61–64
85. Bottou L, Peters J, Quiñonero Candela J, Charles DX, Chickering DM, Portugaly E, Ray D, Simard P, Snelson E (2013) Counterfactual reasoning and learning systems: the example of computational advertising. *J Mach Learn Res* 14:3207–3260
86. Dawid AP (2000) Causal inference without counterfactuals. *J Am Stat Assoc* 95(450):407–424
87. Tan Z (2006) Regression and weighting methods for causal inference using instrumental variables. *J Am Stat Assoc* 101(476):1607–1618
88. Bollen KA (1989) Structural equations with latent variables. Wiley, New York
89. Tarka P (2017) An overview of structural equation modeling: its beginnings, historical development, usefulness and controversies in the social sciences. *Qual Quant* 52:313–354
90. Robins JM (1987) A graphical approach to the identification and estimation of causal parameters in mortality studies with sustained exposure periods. *J Chronic Dis* 40(Suppl 2):139S–161S
91. Lopez-Paz D, Muandet K, Schölkopf B, Tolstikhin I (2015) Towards a learning theory of cause-effect inference. In: Bach F, Blei D (eds) Proceedings of the 32nd international conference on machine learning, Lille, Proceedings of machine learning research, vol 37, pp 1452–1461
92. Suppes P (1970) A Probabilistic theory of causality. North-Holland, Amsterdam
93. Eells E (1970) Probabilistic causality. Cambridge University Press, Cambridge
94. Buchsbaum D, Bridgers S, Skolnick Weisberg D, Gopnik A (2012) The power of possibility: causal learning, counterfactual reasoning, and pretend play. *Philos Trans R Soc B Biol Sci* 367(1599):2202–2212
95. Greenland S, Pearl J, Robins JM (1999) Causal diagrams for epidemiologic research. *Epidemiology* 10(1):37–45
96. Verkuyten M, Thijs J (2002) School satisfaction of elementary school children: the role of performance, peer relations, ethnicity and gender. *Soc Indic Res* 59(2):203–228
97. Cardenas IC, Voordijk H, Dewulf G (2017) Beyond theory: towards a probabilistic causation model to support project governance in infrastructure projects. *Int J Proj Manag* 35(3):432–450

98. Gupta S, Kim HW (2008) Linking structural equation modeling to Bayesian networks: decision support for customer retention in virtual communities. *Eur J Oper Res* 190(3):818–833
99. Kleinberg S, Hripcsak G (2011) A review of causal inference for biomedical informatics. *J Biomed Inform* 44(6):1102–1112
100. Martin W (2014) Making valid causal inferences from observational data. *Prev Vet Med* 113(3):281–297. Special Issue: Schwabe Symposium 2012
101. Wu R, Casella G (2010) Statistical genetics - associating genotypic differences with measurable outcomes. In: Tanur J (ed) *Statistics: a guide to the unknown*, pp 243–254. Holden-Day, San Francisco
102. Frommlet F, Bogdan M, Ramsey D (2016) *Phenotypes and genotypes*. Springer, Berlin
103. Rakitsch B, Stegle O (2016) Modelling local gene networks increases power to detect trans-acting genetic effects on gene expression. *Genome Biol* 17(1):33
104. Brazhnik P, de la Fuente A, Mendes P (2002) Gene networks: how to put the function in genomics. *Trends Biotechnol* 20(11):467–472
105. Hu H, Li Z, Vetta AR (2014) Randomized experimental design for causal graph discovery. In: Ghahramani Z, Welling M, Cortes C, Lawrence ND, Weinberger KQ (eds) *Advances in neural information processing systems*, vol 27. Curran Associates, Inc., Red Hook, pp 2339–2347
106. Isabelle Guyon I, Janzing D, Schölkopf B (2010) Causality: Objectives and assessment. In: Guyon I, Janzing D, Schölkopf B (eds) *Proceedings of workshop on causality: objectives and assessment at NIPS 2008*, Whistler. *Proceedings of machine learning research*, vol 6, pp 1–42
107. Djordjevic D, Yang A, Zadoorian A, Rungruengecharoen K, Ho JW (2014) How difficult is inference of mammalian causal gene regulatory networks? *PLoS ONE* 9(11):1–10
108. Anjum S, Doucet A, Holmes CC (2009) A boosting approach to structure learning of graphs with and without prior knowledge. *Bioinformatics* 25(22):2929–2936
109. Deng M, Emad A, Milenkovic O (2012) Causal compressive sensing for gene network inference. In: 2012 IEEE statistical signal processing workshop, SSP 2012, pp 696–699
110. Krouk G, Lingeman J, Colon AM, Coruzzi G, Denis S (2013) Gene regulatory networks in plants: learning causality from time and perturbation. *Genome Biol* 14(6):123
111. Dondelinger F, Lèbre S, Husmeier D (2013) Non-homogeneous dynamic Bayesian networks with Bayesian regularization for inferring gene regulatory networks with gradually time-varying structure. *Mach Learn* 90(2):191–230
112. Cai X, Bazerque JA, Giannakis GB (2013) Inference of gene regulatory networks with sparse structural equation models exploiting genetic perturbations. *PLoS Comput Biol* 9(5):1–13
113. Rau A, Jaffrézic F, Nuel G (2013) Joint estimation of causal effects from observational and intervention gene expression data. *BMC Syst Biol* 7(1):111
114. Monneret G, Jaffrézic F, Rau A, Zerjal T, Nuel G (2017) Identification of marginal causal relationships in gene networks from observational and interventional expression data. *PLoS ONE* 12(3):1–13
115. Liu B, de la Fuente A, Hoeschele I (2008) Gene network inference via structural equation modeling in genetical genomics experiments. *Genetics* 178(3):1763–1776
116. Tasaki S, Sauerwine B, Hoff B, Toyoshiba H, Gaiteri C, Chaibub Neto E (2015) Bayesian network reconstruction using systems genetics data: comparison of MCMC methods. *Genetics* 199(4):973–989
117. Kalisch M, Mächler M, Colombo D, Maathuis MH, Bühlmann P (2012) Causal inference using graphical models with the R package *pcalg*. *J Stat Softw* 47:11
118. Koski TJ, Noble JM (2012) A review of Bayesian networks and structure learning. *Math Appl* 40(1):53–103
119. Hendriksen JMT, Geersing GJ, Moons KGM, de Groot JAH (2013) Diagnostic and prognostic prediction models. *J Thromb Haemost* 11:129–141
120. Dawid AP, Musio M, Fienberg SE (2016) From statistical evidence to evidence of causality. *Bayesian Anal* 11(3):725–752
121. Sebastiani P, Milton J, Wang L (2011) Designing microarray experiments. Springer, Boston, pp 271–290
122. Bühlmann P, Kalisch M, Meier L (2014) High-dimensional statistics with a view toward applications in biology. *Ann Rev Stat Appl* 1(1):255–278

123. Spirtes P, Glymour CN, Scheines R (2000) Causation, prediction, and search, adaptive computation and machine learning, 2nd edn. The MIT Press, Cambridge. With additional material by David Heckerman. A Bradford Book
124. Pearl J (1988) Probabilistic reasoning in intelligent systems: networks of plausible inference. The Morgan Kaufmann series in representation and reasoning. Morgan Kaufmann, San Francisco
125. Koller D, Pfeffer A (1997) Object-oriented Bayesian networks. In: Proceedings of the thirteenth conference on uncertainty in artificial intelligence, UAI'97. Morgan Kaufmann, San Francisco, pp 302–313
126. Marsland S (2015) Machine learning: an algorithmic perspective, 2nd edn. Chapman & Hall/CRC machine learning & pattern recognition series. CRC Press, Boca Raton
127. Tsamardinos I, Aliferis CF, Statnikov AR, Statnikov E (2003) Algorithms for large scale markov blanket discovery. In: FLAIRS conference, vol 2, pp 376–380
128. Friedman N, Nachman I, Peér D (1999) Learning Bayesian network structure from massive datasets: the sparse candidate algorithm. In: Proceedings of the fifteenth conference on Uncertainty in artificial intelligence. Morgan Kaufmann, San Francisco, pp 206–215
129. Brown LE, Tsamardinos I, Aliferis CF (2004) A novel algorithm for scalable and accurate Bayesian network learning. In: Proceedings of 11th World Congress in Medical Informatics (MEDINFO 04), vol 107, pp 711–715
130. Fu LD, Tsamardinos I (2005) A comparison of Bayesian network learning algorithms from continuous data. In: AMIA annual symposium proceedings, vol 960
131. Vignes M, Vandel J, Allouche D, Ramadan-Alban N, Cierco-Ayrolles C, Schiex T, Mangin B, de Givry S (2011) Gene regulatory network reconstruction using Bayesian networks, the Dantzig selector, the Lasso and their meta-analysis. *PLoS ONE* 6(12):1–15
132. Qi X, Shi Y, Wang H, Gao Y (2016) Grouping parallel Bayesian network structure learning algorithm based on variable ordering. In: Yin H, Gao Y, Li B, Zhang D, Yang M, Li Y, Klawonn F, Tallón-Ballesteros AJ (eds) Intelligent data engineering and automated learning – IDEAL 2016. Springer International Publishing, Cham, pp 405–415
133. Mengshoel OJ (2010) Understanding the scalability of Bayesian network inference using clique tree growth curves. *Artif Intell* 174(12):984–1006
134. De Campos CP (2011) New complexity results for map in Bayesian networks. In: Proceedings of the twenty-second international joint conference on artificial intelligence, vol 3, IJCAI'11. AAAI Press, Menlo Park, pp 2100–2106
135. Cooper GF (1990) The computational complexity of probabilistic inference using Bayesian belief networks. *Artif Intell* 42(2):393–405
136. Handa H, Katai O (2003) Estimation of Bayesian network algorithm with GA searching for better network structure. In: Proceedings of the 2003 international conference on neural networks and signal processing, vol 1, pp 436–439
137. Malone B, Yuan C, Hansen EA, Bridges S (2011) Improving the scalability of optimal Bayesian network learning with external-memory frontier breadth-first branch and bound search. In: Proceedings of the twenty-seventh conference on uncertainty in artificial intelligence, UAI'11. AUAI Press, Arlington, pp 479–488
138. Adabior ES, Acquaah-Mensah GK, Oduro FT (2015) SAGA: a hybrid search algorithm for Bayesian network structure learning of transcriptional regulatory networks. *J Biomed Inform* 53:27–35
139. Nikolova O, Aluru S (2012) Parallel Bayesian network structure learning with application to gene networks. In: 2012 International conference for high performance computing, networking, storage and analysis (SC), pp 1–9
140. Madsen AL, Jensen F, Salmer A, Langseth H, Nielsen TD (2017) A parallel algorithm for Bayesian network structure learning from large data sets. *Knowl Based Syst* 117:46–55
141. Thibault G, Aussem A, Bonnevay S (2009) Incremental Bayesian network learning for scalable feature selection. In: Adams NM, Robardet C, Siebes A, Boulicaut JF (eds) Advances in intelligent data analysis VIII. Springer, Berlin, pp 202–212
142. Stegle O, Janzing D, Zhang K, Mooij JM, Schölkopf B (2010) Probabilistic latent variable models for distinguishing between cause and effect. In: Lafferty JD, Williams CKI, Shawe-Taylor J, Zemel RS, Culotta A (eds)

- Advances in neural information processing systems, vol 23. Curran Associates, Inc., Red Hook, pp 1687–1695
- 143. He Y, Jia J, Yu B (2013) Reversible MCMC on Markov equivalence classes of sparse directed acyclic graphs. *Ann Stat* 41(4): 1742–1779
 - 144. Peters J, Bhlmann P (2015) Structural intervention distance for evaluating causal graphs. *Neural Comput* 27(3):771–779
 - 145. Schwarz G (1978) Estimating the dimension of a model. *Ann Stat* 6(2):461–464
 - 146. Huynh-Thu VA, Irrthum A, Wehenkel L, Geurts P (2010) Inferring regulatory networks from expression data using tree-based methods. *PLoS ONE* 5:1–10
 - 147. Vandel J, Mangin B, Vignes M, Leroux D, Loudet O, Martin-Magniette ML, De Givry S (2012) Gene regulatory network inference with extended scores for Bayesian networks. *Revue d'Intelligence Artificielle* 26(6):679–708
 - 148. Chiquet J, Smith A, Grasseau G, Matias C, Ambroise C (2009) SIMoNe: Statistical Inference for MOdular NEtworks. *Bioinformatics* 25(3):417–418
 - 149. Vallat L, Kemper CA, Jung N, Maumy-Bertrand M, Bertrand F, Meyer N, Pocheville A, Fisher JW, Gribben JG, Bahram S (2013) Reverse-engineering the genetic circuitry of a cancer cell with predicted intervention in chronic lymphocytic leukemia. *Proc Natl Acad Sci* 110(2):459–464



Chapter 6

A Multiattribute Gaussian Graphical Model for Inferring Multiscale Regulatory Networks: An Application in Breast Cancer

Julien Chiquet, Guillem Rigaill, and Martina Sundqvist

Abstract

This chapter addresses the problem of reconstructing regulatory networks in molecular biology by integrating multiple sources of data. We consider data sets measured from diverse technologies all related to the same set of variables and individuals. This situation is becoming more and more common in molecular biology, for instance, when both proteomic and transcriptomic data related to the same set of “genes” are available on a given cohort of patients.

To infer a consensus network that integrates both proteomic and transcriptomic data, we introduce a multivariate extension of Gaussian graphical models (GGM), which we refer to as *multiattribute* GGM. Indeed, the GGM framework offers a good proxy for modeling direct links between biological entities. We perform the inference of our multivariate GGM with a neighborhood selection procedure that operates at a multiscale level. This procedure employs a group-Lasso penalty in order to select interactions which operate both at the proteomic and at the transcriptomic level between two genes. We end up with a *consensus* network embedding information shared at multiple scales of the cell. We illustrate this method on two breast cancer data sets. An R-package is publicly available on github at <https://github.com/jchiquet/multivarNetwork> to promote reproducibility.

Key words Multiscale regulatory network, Gaussian graphical model, Group-Lasso, Proteomic data, Multi-omic data

1 Introduction

Gaussian Graphical Models (GGMs): A Canonical Framework for Network Inference Modeling Gaussian Graphical Models (GGMs) [1, 2] are a very convenient tool for describing the patterns at play in complex data sets. Indeed, through the notion of partial correlation, they provide a well-studied framework for spotting direct relationships between variables and thus reveal the latent structure in a way that can be easily interpreted. Application areas are very broad and include, for instance, gene regulatory

network inference in biology (using gene expression data) as well as spectroscopy, climate studies, functional magnetic resonance imaging, etc. Estimation of GGMs in a sparse, high-dimensional setting has thus received much attention in the last decade, especially in the case of a single homogeneous data set [3–7].

However, this simple canonical setup is uncommon in real-world data sets, because of both the complexity of the mechanism at play in biology and the multiplicity of the sources of data in omic. Hence, the need for variants of sparse GGM more adapted to recent omic data set is huge.

As a first example, the work developed in [8, 9] addresses the introduction of a possible special organization of the network itself to drive the reconstruction process. Indeed, while sparsity is necessary to solve the problem when few observations are available, biasing the estimation of the network towards a given topology can help us find the correct graph in a more robust way, by preventing the algorithm from looking for solutions in regions where the correct graph is less likely to reside. As a second example, [10] addresses the problem of sample heterogeneity which typically occurs when several assays are performed in different experimental conditions that potentially affect the regulations but are still merged together to perform network inference as data is very scarce. We remedy heterogeneity among sample experiments by estimating multiple GGMs, each of which matches different modalities of the same set of variables, which correspond here to the different experimental conditions. This idea, coupled with the integration of biological knowledge, was further explored for application in cancer in [11].

A deeper generalization of GGM comes by integrating multiple types of data measured from diverse platforms, what is sometimes referred to as *horizontal* integration: not only does this mean a better treatment of the heterogeneity of the data, but it also makes the network reconstruction more robust. The model presented in this chapter gives an answer to this question by offering a solution to reconstruct a sparse, multiattribute GGM, recoursing on both proteomic and transcriptomic data to infer a consensus network. Our main motivating application is a better understanding of heterogeneity in breast cancer, as detailed below.

Proteomic and Transcriptomic Data Integration in Cancer
Protein deregulations, leading to abnormal activation of signaling pathways, contribute to tumorigenesis [12]. Knowing the level of activation of the signaling pathways in any subgroup of tumors could therefore be a key indication to understand the biological mechanisms involved in tumorigenesis, and to identify some therapeutic targets. Therefore, the analysis of proteins is essential. However, measuring the expression of proteins is more difficult to implement than the measure of transcriptome (RNA) or genome (DNA). Several technologies have been developed to measure the

proteome, but the number of samples and the number of proteins that can be studied simultaneously is, up to now, limited. A useful technique for this task is the RPPA (Reverse-Phase Protein Arrays). It allows studying protein expression levels and the activity status of a few hundred proteins by analyzing their phosphorylated state, in several hundreds of samples [13].

To summarize, better understanding the proteome of tumors is essential to further our knowledge of cancer cells but proteome data are still small and rare. Integration of proteomic and transcriptomic data is a promising avenue to get the most of available proteomic data sets and better understand the relative roles of transcriptome and proteome. To take into account the different levels of information, a solution is to use multivariate GGM. Since it is probable that the proteomic and transcriptomic heterogeneity in cancers is caused by some few major underlying alterations, the hypothesis of proximity in between networks seems reasonable. Identifying commonalities between the transcriptome and proteome networks should help the prediction of the proteome using the transcriptome for which several large public cancer data sets are available.

Chapter Outline In the next section, we give a quick overview of the literature of sparse GGM (models, basic theoretical results, inference, and software). This provides the reader with the necessary material to approach the model at play in this chapter, dedicated to multiattribute GGM, introduced in Subheading 3. In Subheading 4, we perform some numerical studies: we demonstrate on simulated data the superiority of our approach in several scenarios. Then, two breast cancer data sets are used to illustrate the reconstruction of multiscale regulatory networks. See also Chapter 7 for a different perspective on the problem of multiple data integration.

2 Background

This section provides an overview on the state-of-the-art ℓ_1 -regularization methods for sparse GGM inference and their most recent striking variants, insisting on their computational and statistical properties.

2.1 Basics on Gaussian Graphical Models

Let $\mathcal{P} = \{1, \dots, p\}$ be a set of fixed vertices and $X = (X_1, \dots, X_p)^\top$ a random vector describing a signal over this set. The vector $X \in \mathbb{R}^p$ is assumed to be multivariate Gaussian with unknown mean and unknown covariance matrix $\Sigma = (\Sigma_{ij})_{(i,j) \in \mathcal{P}^2}$. No loss of generality is involved when centering X , so we may assume that $X \sim \mathcal{N}(\mathbf{0}_p, \Sigma)$. The covariance matrix Σ , equal to $\mathbb{E}(XX^\top)$ under the assumption that X is centered, belongs to the set \mathcal{S}_p^+ of positive-definite symmetric matrices of size $p \times p$.

Graph of Conditional Dependencies GGMs endow Gaussian random vectors with a graphical representation \mathcal{G} of their *conditional dependency structure*: two variables i and j are linked by an undirected edge (i, j) if, conditional on all other variables indexed by $\mathcal{P} \setminus \{i, j\}$, random variables X_i and X_j remain or become dependent. Thanks to the Gaussian assumption, conditional independence actually boils down to a zero conditional covariance $\text{cov}(X_i, X_j | X_{\mathcal{P} \setminus \{i, j\}})$, or equivalently to a zero partial correlation which we denote by ρ_{ij} , the latter being a normalized expression of the former.

Concretely, the inference of a GGM is based upon a classical result originally emphasized in [14] stating that partial correlations ρ_{ij} are actually proportional to the corresponding entries in the *inverse* of the covariance matrix $\boldsymbol{\Sigma}^{-1} = \boldsymbol{\Theta}$, also known as the *concentration* (or *precision*) matrix. More precisely, we have

$$\rho_{ij} = -\boldsymbol{\Theta}_{ij}/\sqrt{\boldsymbol{\Theta}_{ii}\boldsymbol{\Theta}_{jj}}, \quad \boldsymbol{\Theta}_{ii} = \text{Var}(X_i | X_{\mathcal{P} \setminus i})^{-1}; \quad (1)$$

thus $\boldsymbol{\Theta}$ directly describes the conditional dependency structure of X . Indeed, after a simple rescaling, $\boldsymbol{\Theta}$ can be interpreted as the adjacency matrix of an undirected weighted graph representing the partial covariance (or correlation) structure between variables X_1, \dots, X_p . Formally, we denote by $\mathcal{G} = (\mathcal{P}, \mathcal{E})$ this graph, the edges of which are characterized by:

$$(i, j) \in \mathcal{E} \Leftrightarrow \boldsymbol{\Theta}_{ij} \neq 0, \quad \forall (i, j) \in \mathcal{P}^2 \text{ such that } i \neq j.$$

In words, \mathcal{G} has no self-loop and contains all edges (i, j) such that $\boldsymbol{\Theta}_{ij}$ is nonzero. Therefore, recovering nonzero entries of $\boldsymbol{\Theta}$ is equivalent to inferring the graph of conditional dependencies \mathcal{G} , and the correct identification of nonzero entries is the main issue in this framework.

Maximum Likelihood Inference GGMs fall into the family of exponential models for which the whole range of classical statistical tools applies. As soon as the sample size n is greater than the number p of variables, the likelihood admits a unique maximum over \mathcal{S}_p^+ , defining a maximum likelihood estimator (MLE): suppose we observe a sample $\{X^1, \dots, X^n\}$ composed of n i.i.d. copies of X , stored row-wise once centered in a matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$ such that $(X^i)^\top$ is the i th row of \mathbf{X} . The empirical covariance matrix is denoted by $\mathbf{S}_n = \mathbf{X}^\top \mathbf{X} / n$. Let $\det(M)$ and $\text{Tr}(M)$ be the determinant and the trace of a matrix M . Maximizing the likelihood is equivalent to:

$$\hat{\boldsymbol{\Theta}}^{\text{mle}} = \arg \max_{\boldsymbol{\Theta} \in \mathcal{S}_p^+} \log \det(\boldsymbol{\Theta}) - \text{Tr}(\boldsymbol{\Theta} \mathbf{S}_n), \quad (2)$$

When $n > p$, Problem (2) admits a unique solution equal to \mathbf{S}_n . The scaled empirical covariance matrix \mathbf{S}_n follows a Wishart distribution, while its inverse \mathbf{S}_n^{-1} follows an inverse Wishart distribution with computable parameters.

There are two major limitations with the MLE regarding the objective of graph reconstruction by recovering the pattern of zeroes in $\boldsymbol{\Theta}$. First, it provides an estimate of the saturated graph: all variables are connected to each other; second, we need n to be larger than p to be able to even define this estimator, which is rarely the case in genomics. In any case, the need for regularization and feature selection is huge. A natural assumption is that the true set of direct relationships between the variables remains small, that is, the true underlying graph is sparse (say, of the order of p rather than the order of p^2). Sparsity makes estimation feasible in the case where $n < p$ since we can concentrate on sparse or shrinkage estimators with fewer degrees of freedom than in the original problem. Henceforth, the question of selecting the correct set of edges in the graph is treated as a question of variable selection.

High-Dimensional Inference of GGM The different methods for the inference of sparse GGMs in high-dimensional settings fall into roughly three categories. The first contains constraint-based methods, performing statistical tests [15–19]. However, they either suffer from the excessive computational burden [15, 19] or strong assumptions [16, 17] that correspond to regimes never attained in real situations. The second of these categories is composed of Bayesian approaches, see, for instance, [20–23]. However, constructing priors on the set of concentration matrices is not a trivial task and the use of MCMC procedures limits the range of applications to moderate-sized networks. The third category contains regularized estimators, which add a penalty term to the likelihood in order to reduce the complexity or degrees of freedom of the estimator and more generally regularize the problem: throughout this chapter, we focus on ℓ_1 -regularized procedures, which are freed from any test procedure—and thus multiple testing issues—since they directly perform estimation and selection of the most significant edges by zeroing entries in the estimator of $\boldsymbol{\Theta}$. The remainder of this section is dedicated to a quick review of the state-of-the-art methods of this kind.

2.2 Sparse Methods for GGM Inference

The idea underlying sparse methods for GGM is the same as for the Lasso in linear regression [24]: it basically uses ℓ_1 -regularization as a convex surrogate of the ideal but computationally intensive ℓ_0 -regularized problem:

$$\arg \max_{\boldsymbol{\Theta} \in \mathcal{S}_p^+} \log \det(\boldsymbol{\Theta}) - \text{Tr}(\boldsymbol{\Theta} \mathbf{S}_n) - \lambda \|\boldsymbol{\Theta}\|_{\ell_0}. \quad (3)$$

Problem (3) achieves a trade-off between the maximization of the likelihood and the sparsity of the graph within a single optimization problem. The penalty term can also be interpreted as a log prior on the coefficients in a Bayesian perspective. BIC or AIC criteria are special cases of such ℓ_0 regularized problems, except that the maximization is made upon a restricted subset of candidates $\{\tilde{\Theta}_1, \dots, \tilde{\Theta}_m\}$ and the choice of λ is fixed ($\log(n)$ for BIC and 2 for AIC). Actually, solving (3) would require the exploration of all possible $2^{p(p-1)/2}$ graphs. On the contrary, by preserving the convexity of the optimization problem, ℓ_1 -regularization paves the way to fast algorithms. For the price of a little bias on all the coefficients, we get to shrink some coefficients to exactly 0, operating selection and estimation in one single step as hoped in Problem (3).

Graphical-Lasso The criterion optimized by the graphical-Lasso was simultaneously proposed in [25] and [5]. It corresponds to the estimator obtained by fitting the ℓ_1 -penalized Gaussian log-likelihood, i.e., the tightest convex relaxation of (3):

$$\widehat{\Theta}_\lambda^{\text{glasso}} = \arg \max_{\Theta \in \mathcal{S}_p^+} \log \det(\Theta) - \text{Tr}(\Theta S_n) - \lambda \|\Theta\|_{\ell_1}. \quad (4)$$

In this regularized problem, the ℓ_1 -norm drives some coefficients of Θ to zero. The nonnegative parameter λ tunes the global amount of sparsity: the larger the λ , the fewer edges in the graph. A large enough penalty level produces an empty graph. As λ decreases towards zero, the estimated graph tends towards the saturated graph and the estimated concentration matrix tends towards the usual MLE (2). By construction, this approach guarantees a well-behaved estimator of the concentration matrix, i.e., sparse, symmetric, and positive definite, which is a great advantage of this method.

Ever since Criterion (4) was proposed, many efforts have been dedicated to developing efficient algorithms for its optimization. In the original proposal of [5], it is shown that solving for one row of matrix Θ in (4) while keeping other rows fixed boils down to a Lasso problem. The global problem is solved by cycling over the matrix rows until convergence. Thus, if one considers that L passes over the whole matrix are needed to reach convergence, a rough estimation of the overall cost is of the order of $Lp \times (\text{cost for solving for one row})$. With a block-coordinate update each iteration over a row has $\mathcal{O}(p^3)$ complexity and their implementation is $\mathcal{O}(Lp^4)$ for L sweeps over the whole matrix $\widehat{\Theta}$. In [5] again, a rigorous analysis is conducted in Nesterov's framework [26] showing that the complexity for a single λ reaches $\mathcal{O}(p^{4.5}/\varepsilon)$ where ε is the desired accuracy of the final estimate.

The *Graphical-Lasso* algorithm of [4] follows the same line but builds on a coordinate descent algorithm to solve each underlying

Lasso problem. While no precise complexity analysis is possible with these methods, empirical results tend to show that this algorithm is faster than the original proposal of [5]. Additional insights on the convergence of the graphical-Lasso are provided in [27], simultaneously with [28], showing how to take advantage of the problem sparsity by decomposing (4) into block diagonal problems depending on λ : this considerably reduces the computational burden in practice. Implementations of the graphical-Lasso algorithm are available in the R-packages **glasso**, **huge** [29], or **simone** [9]. The most recent notable efforts related to the optimization of (4) are due to [30, 31] and the QUIC (then BIG&QUIC) algorithm, a quadratic approximation which allows (4) to be solved up to $p = 1,000,000$ with a superlinear rate of convergence and with bounded memory. The R-package **quic** implements the first version of this algorithm.

On the statistical side, the most striking results are due to [32]: they show that selection consistency of the estimator defined by (4)—that is, recovery of the true underlying graphical structure—is met in the sub-Gaussian case when, for an appropriate choice of λ , the sample size n is of the same order as $\mathcal{O}(d^2 \log(p))$, where d is the highest degree in the target graph. Additional conditions on the empirical covariance between relevant and irrelevant features are required, known as the “irrepresentability conditions” in the Lasso case. Such statistical results are important since they provide insights on the “data” situations where such methods may either be successful or completely hopeless. More on this is discussed in [33]. For instance, this should prevent blindly applying the graphical-Lasso in situations where the sample size n is too small compared to p . Similarly, when the presence of hub nodes with high degree is suspected, the estimated graph should be interpreted with care.

Neighborhood Selection This approach, proposed in [3], determines the graph of conditional dependencies by solving a series of p independent Lasso problems, successively estimating the neighborhoods of each variable and then applying a final reconciliation step as posttreatment to recover a symmetric adjacency matrix. Concretely, a given column \mathbf{X}_j of the data matrix is “explained” by the remaining columns $\mathbf{X}_{\setminus j}$ corresponding to the remaining variables: the set $\text{ne}(j)$ of neighbors of variable j in the graph \mathcal{G} is estimated by the support of the vector solving

$$\hat{\boldsymbol{\beta}}_j = \arg \min_{\boldsymbol{\beta} \in \mathbb{R}^{p-1}} \frac{1}{2n} \|\mathbf{X}_j - \mathbf{X}_{\setminus j} \boldsymbol{\beta}\|_{\ell_2}^2 + \lambda \|\boldsymbol{\beta}\|_{\ell_1}. \quad (5)$$

Indeed, if each row of \mathbf{X} is drawn from a multivariate Gaussian $\mathcal{N}(\mathbf{0}, \boldsymbol{\Theta}^{-1})$, then the best linear approximation of \mathbf{X}_j by $\mathbf{X}_{\setminus j}$ is given by:

$$\mathbf{X}_j = \sum_{k \in \text{ne}(j)} \beta_{jk} \mathbf{X}_k = - \sum_{k \in \text{ne}(j)} \frac{\Theta_{jk}}{\Theta_{jj}} \mathbf{X}_k, \quad (6)$$

thus coefficients $\boldsymbol{\beta}_j$ and column $\boldsymbol{\Theta}_j$ —once its diagonal elements are removed—share the same support. By support, we mean the set of nonzero coefficients. Adjusting (5) for each $j = 1, \dots, p$ allows us to reconstruct the full graph \mathcal{G} . Because the neighborhoods of the p variables are selected separately, a post-symmetrization must be applied to manage inconsistencies between edge selections; [3] suggests AND or OR rules.

Let us fill the gap with Criterion (4). First, note that the p regression problems can be rewritten as a unique matrix problem, where \mathbf{B} contains p vectors $\boldsymbol{\beta}_j, j = 1, \dots, p$:

$$\hat{\mathbf{B}}^{\text{ns}} = \arg \min_{\mathbf{B} \in \mathbb{R}^{p \times p}, \text{diag}(\mathbf{B}) = \mathbf{0}_p} \frac{1}{2} \text{Tr}(\mathbf{B}^\top \mathbf{S}_n \mathbf{B}) - \text{Tr}(\mathbf{B}^\top \mathbf{S}_n) + \lambda \|\mathbf{B}\|_{\ell_1}. \quad (7)$$

In fact, it can be shown [8, 34, 35] that the optimization problem (7) corresponds to the minimization of a penalized, negative *pseudo*-likelihood: the joint distribution of X is approximated by the product of the p distributions of the p variables conditional on the other ones, that is:

$$\log \mathbb{P}(\mathbf{X}; \boldsymbol{\Theta}) = \sum_{j=1}^p \sum_{i=1}^n \log \mathbb{P}(X_j^i | X_{\setminus j}^i; \boldsymbol{\Theta}_j).$$

This pseudo-likelihood is based upon the (false) assumption that conditional distributions are independent. Moreover, all variables are assumed to share the same variance in this formulation. Building on these remarks, [34] amend criterion (7) by the adjunction of an additional symmetry constraint and introduce additional parameters to account for different variances between the variables.

Concerning the computational aspect, this approach has very efficient implementation as it basically boils down to solving p Lasso problems. Suppose, for instance, that the target neighborhood size is k per variable: fitting the whole solution path of a Lasso problem using the Lars algorithm can be done in $\mathcal{O}(npk)$ complexity [36]. This must be multiplied by p for the whole network, yet we underline that a parallel implementation is straightforward in this case. This makes this approach quite competitive, especially when coupled with additional bootstrap or resampling techniques [37].

On the statistical side, neighborhood selection has been reported to be sometimes empirically more accurate in terms of edge detection than the graphical-Lasso [34, 38] on certain types of data. This is somewhat supported by the statistical analysis

of [32], who show that under the classical irrepresentability conditions for the Lasso [3, 39] and for an appropriate choice of λ , neighborhood selection achieves selection consistency with high probability when the sample size n is of the order of $\mathcal{O}(d \log(p))$ with d the maximal degree of the target graph \mathcal{G} . This is to be compared with the $\mathcal{O}(d^2 \log(p))$ required by the graphical-Lasso (even if the corresponding “irrepresentability conditions” are not strictly comparable). A rough explanation for this difference on the asymptotic is that the graphical-Lasso intends to estimate the concentration matrix on top of selecting the nonzero entries, while neighborhood selection focuses on the selection problem.

Model Selection Issues Up to this point, we have completely avoided the fundamental model selection issue, that is, the choice of the tuning parameter λ , which is at play in all the sparse methods mentioned thus far. The first possibility is to rely on information criteria of the form:

$$\text{IC}_\lambda = -2\text{loglik}(\widehat{\boldsymbol{\Theta}}_\lambda; \mathbf{X}) + \text{pen}(\text{df}(\widehat{\boldsymbol{\Theta}}_\lambda)),$$

where “pen” is a function penalizing the model complexity, described by df , the degrees of freedom of the current estimator. We meet the AIC by choosing $\text{pen}(x) = 2x$ and the BIC by choosing $\text{pen}(x) = \log(n)x$. However, AIC and BIC are based upon assumptions which are not suited to high-dimensional settings (see [40]). Moreover, the notion of degrees of freedom for sparse methods has to be specified, not to mention that one has to adapt these criteria to the case of GGMs. An example of a criterion meeting these prerequisites is the extended BIC for sparse GGMs [41]:

$$\text{EBIC}_\gamma(\widehat{\boldsymbol{\Theta}}_\lambda) = -2\text{loglik}(\widehat{\boldsymbol{\Theta}}_\lambda; \mathbf{X}) + |\mathcal{E}_\lambda|(\log(n) + 4\gamma \log(p)), \quad (8)$$

where the function df is equal to $|\mathcal{E}|$, the total number of edges in the inferred graph. The parameter $\gamma \in [0, 1]$ is used to adjust the tendency of the usual BIC—recovered for $\gamma = 0$ —to choose overly dense graphs in the high-dimensional setting. Further justification can be found in [41]. A competing approach, designed to compare a family of GGMs—possibly inferred with different methods, is GGMSelect [42, 43].

Another possibility is to rely on resampling/subsampling procedures to select a set of edges which are robust to small variations of the sample. The most popular approach is the *Stability Selection* procedure proposed in [37], also related to the bootstrapped procedure of [44]. A similar approach, called StaRS (Stability approach to Regularization Selection) is developed specifically in the context of GGM in [45]. The basic idea is as follows: for a given range of the tuning parameter $\Lambda = [\lambda_{\min}, \lambda_{\max}]$, the same method

is fitted on many subsamples (with or without replacement) with size, say $n/2$. The idea is then to construct a score indexed on Λ that measures stability—or instability—of the selected variables. The selected edges are those matching a given score, for which the probability of false discovery is controlled. This requires an additional threshold in place of a choice of λ , but the authors in [37, 45] claim that such a threshold is typically much less sensitive than is the tuning parameter λ . An application of such resampling techniques to the inference of biological networks has been pursued with success in [46], advocating for the use of stability methods on real problems.

3 Accounting for Multiscale Data: Multiattribute GGM

We now place ourselves in the situation where, for our collection of features \mathcal{P} , we observe not one but several attributes. The question at hand remains the same, that is to say, unraveling strong interactions between these features according to the observation of their attributes. Such networks are known as “association networks,” which are systems of interacting elements, where a link between two different elements indicates a sufficient level of similarity between element attributes. In this section, we are interested in reconstructing such networks based upon n observations of a set of K attributes of the p elements composing the vertices of the network. To this end, we propose a natural generalization of sparse GGMs to sparse *multiattribute* GGMs.

Why Multiattribute Networks? The need for multiattribute networks is relevant in many application fields, but seems particularly applicable in genomics. Indeed, with the plurality of emerging technologies and sequencing techniques, it is possible to record many signals related to the same set of biological features at various scales or locations of the cell. Consider, for instance, the simplifying—still hopefully didactic—central dogma of molecular biology, sketched in Fig. 1: basically, expression of a gene encoding for a protein can be measured either at the transcriptome level, in

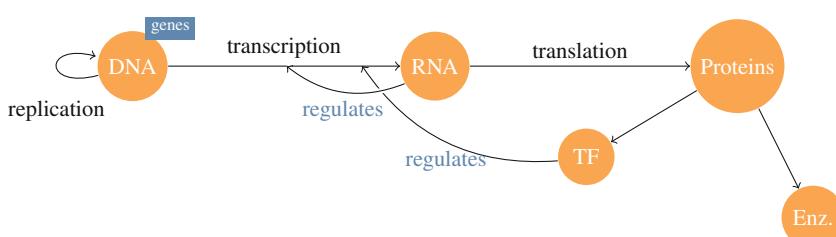


Fig. 1 Basic example of a multiattribute network in genomics: activity of a gene can be measured at the transcriptomic and proteomic levels, and gene regulation affected accordingly (TF Transcription Factor, Enz. Enzyme)

terms of its quantity of RNA, or at the protein level, in terms of the concentration of the associated protein. Still, different technologies are used to measure either the transcriptome or the proteome, typically, microarray or sequencing technology for gene expression levels and cytometric or immunofluorescence experiments for protein concentrations. Although these signals are very heterogeneous (different levels of noise, count vs. continuous data, etc.), they do share commonality as they undergo common biological processes. We then put an edge in the network if it is supported in both spaces (gene and protein spaces). Our hope is that molecular profiles combined on the same set of biological samples can be *synergistic*, in order to identify a “consensus” and hopefully more robust network.

Multiattribute GGM Let $\mathcal{P} = \{1, \dots, p\}$ be a set of variables of interest, each of them having some K attributes. Consider the random vector $X = (X_1, \dots, X_p)^\top$ such as $X_i = (X_{i1}, \dots, X_{iK})^\top \in \mathbb{R}^K$ for $i \in \mathcal{P}$. The vector $X \in \mathbb{R}^{pK}$ describes the K recorded signals for the p features. We assume that X is a multivariate centered Gaussian vector, that is, $X \sim \mathcal{N}(\mathbf{0}, \Sigma)$, with covariance and concentration matrices defined block-wise

$$\Sigma = \begin{bmatrix} \Sigma_{11} & \Sigma_{1p} \\ \ddots & \ddots \\ \Sigma_{p1} & \Sigma_{pp} \end{bmatrix}, \quad \Theta = \begin{bmatrix} \Theta_{11} & \Theta_{1p} \\ \ddots & \ddots \\ \Theta_{p1} & \Theta_{pp} \end{bmatrix},$$

$$\Sigma_{ij}, \Theta_{ij} \in \mathcal{M}_{K,K}, \quad \forall (i, j) \in \mathcal{P}^2,$$

where $\mathcal{M}_{a,b}$ is the set of real-valued matrices with a rows and b columns. Such a multiattribute framework has been studied in Katenka and Kolaczyk [47] with a reconstruction method based upon canonical correlations in order to test dependencies between pairs (i, j) at the attribute level using covariance. Here, we propose to rely on partial correlations in a multivariate framework rather than (canonical) correlations to describe relationships between the features, and thus extend GGM to a multiattribute framework. The objective is to define a “canonical” version of partial correlations. In our setting, the target network $\mathcal{G} = (\mathcal{P}, \mathcal{E})$ is defined as the multivariate analog of the conditional graph for univariate GGM, that is:

$$(i, j) \in \mathcal{E} \Leftrightarrow \Theta_{ij} \neq \mathbf{0}_{KK}, \quad \forall i \neq j. \quad (9)$$

In words, there is no edge between two variables i and j when their attributes are all conditionally independent.

A Multivariate Version of Neighborhood Selection Our idea for performing sparse multiattribute GGM inference is to define a multivariate analog of the neighborhood selection approach [3]

(see Subheading 2.2, Eqs. (5) and (7)). Indeed, it seems to be the most natural and convenient setup towards multivariate generalization. Nevertheless, other sparse GGM inference methods like the graphical-Lasso (4) should have an equivalent multiattribute version. A possibility is explored in [48], for instance.

To extend the neighborhood selection approach to a multiattribute version, we look at the multivariate analog of Eq. (6): in a multivariate linear regression setup, it is a matter of straightforward algebra to see that the conditional distribution of $X_j \in \mathbb{R}^K$ on the other variables is

$$X_j | X_{\setminus j} = x \sim \mathcal{N}(-\boldsymbol{\Theta}_{jj}^{-1} \boldsymbol{\Theta}_{j \setminus j} x, \boldsymbol{\Theta}_{jj}^{-1}) .$$

Equivalently, letting $\mathbf{B}_j^T = -\boldsymbol{\Theta}_{jj}^{-1} \boldsymbol{\Theta}_{j \setminus j}$, one has

$$X_j | X_{\setminus j} = \mathbf{B}_j^T X_{\setminus j} + \boldsymbol{\epsilon}_j \quad \boldsymbol{\epsilon}_j \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Theta}_{jj}^{-1}), \quad \boldsymbol{\epsilon}_j \perp X,$$

where $\mathbf{B}_j \in \mathcal{M}_{(p-1)K, K}$ is defined block-wise

$$\mathbf{B}_j = \begin{bmatrix} \mathbf{B}_j^{(1)} \\ \vdots \\ \mathbf{B}_j^{(j-1)} \\ \mathbf{B}_j^{(j+1)} \\ \vdots \\ \mathbf{B}_j^{(p)} \end{bmatrix} = - \begin{bmatrix} \boldsymbol{\Theta}_{j1} \\ \vdots \\ \boldsymbol{\Theta}_{j(j-1)} \\ \boldsymbol{\Theta}_{j(j+1)} \\ \vdots \\ \boldsymbol{\Theta}_{j(p)} \end{bmatrix}^\top \times \boldsymbol{\Theta}_{jj}^{-1},$$

and where each $\mathbf{B}_j^{(i)}$ is a $K \times K$ matrix which links attributes of variables (i, j) . We see that recovering the support of \mathbf{B}_j block-wise is equivalent to reconstructing the network defined in (9). Estimation of \mathbf{B}_j is thus typically achieved through sparse methods. To this end, we consider an i.i.d. sample $\{X^\ell\}_{\ell=1}^n$ of X such that each attribute is observed n times for the p variables, each \mathbf{x}^ℓ being a pK -size row vector staked in a $\mathcal{M}_{n,pK}$ data matrix \mathbf{X} , so that $\mathbf{X}_j \in \mathcal{M}_{n,K}$ is a real-value, $n \times K$ block matrix containing the data related to the j th variable:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}^1 \\ \vdots \\ \mathbf{x}^n \end{bmatrix} = [\mathbf{X}_1 \dots \mathbf{X}_p] = \left[\begin{array}{c|c|c} X_{11}^1 \dots X_{1K}^1 & \dots & X_{p1}^1 \dots X_{pK}^1 \\ \hline \vdots & \vdots & \dots \\ \hline X_{11}^n \dots X_{1K}^n & \dots & X_{p1}^n \dots X_{pK}^n \end{array} \right].$$

Using these notations, a direct generalization of the neighborhood selection is to predict for each $j = 1, \dots, p$ the data block

\mathbf{X}_j by regressing on $\mathbf{X}_{\setminus j}$. In matrix form, this can be written as the optimization problem:

$$\arg \min_{\mathbf{B}_j \in \mathbb{R}} J(\mathbf{B}_j), \quad J(\mathbf{B}_j) = \frac{1}{2n} \|\mathbf{X}_j - \mathbf{X}_{\setminus j} \mathbf{B}_j\|_F^2 + \lambda \Omega(\mathbf{B}_j), \quad (10)$$

where $\|\mathbf{A}\|_F = \sqrt{\sum_{i,j} \mathbf{A}_{ij}^2}$ is the Frobenius norm of matrix \mathbf{A} and Ω is a penalty which constrains \mathbf{B}_j block-wise.

Choosing a Penalizer Various choices for Ω in Eq. (10) seem relevant: by simply setting $\Omega_0(A) = \sum_{i,j} |A_{i,j}|$, we just encourage sparsity among the \mathbf{B}_i and thus do not couple the attributes. A clever choice would be to activate a set of attributes all together: hence, the group is defined by all the K attributes between variables i and j , therefore the penalizer turns to a group-Lasso-like penalty

$$\Omega_1(\mathbf{B}_j) = \sum_{i \in \mathcal{P} \setminus j} \|\mathbf{B}_j^{(i)}\|_F, \quad (11)$$

in which case convex analysis and subdifferential calculus (see [49]) can be used to show that a \mathbf{B}_i is optimal for Problem (10) iff

$$\begin{cases} \forall i : \mathbf{B}_j^{(i)} \neq 0, & \left(\mathbf{S}_{ij} + \frac{\lambda}{\|\mathbf{B}_j^{(i)}\|_F} I \right)^{-1} \mathbf{S}_{ij} = \mathbf{B}_j^{(i)}, \\ \forall i : \mathbf{B}_j^{(i)} = \mathbf{0}_{KK}, & \|\mathbf{S}_{ij}\|_F \leq \lambda \end{cases}, \quad (12)$$

where $\mathbf{S}_{ij} \in \mathcal{M}_{KK}$ is a $K \times K$ block in the empirical covariance matrix $\mathbf{S}_n = n^{-1} \mathbf{X}^\top \mathbf{X}$, which shows the same block-wise decomposition as $\boldsymbol{\Sigma}$ or $\boldsymbol{\Theta}$. This paves the way for an optimization algorithm like block-coordinate descent which we implemented, although we omit details here.

4 Numerical Experiments

Simulation Study We propose a simple simulation to illustrate the interest of using multiattribute networks. The simulations are set up as follows:

1. Draw a random undirected network with p nodes from the Erdős–Renyi model with adjacency matrix \mathbf{A} ;
2. Expand the associated adjacency matrix to multivariate space with

$$\mathbf{M} = \mathbf{A} \otimes \mathbb{S} + \mathbf{I}_{p \times K}$$

where \otimes is the Kronecker product. The $K \times K$ matrix \mathbb{S} is used to consider different scenarios of agreement across the attributes of two genes. We consider three cases:

- (a) $\mathbb{S} = \mathbf{I}_{K,K}$ the $K \times K$ identity matrix: same intra-attribute network and no inter-attribute interactions;
 - (b) $\mathbb{S} = \mathbf{I}_{K,K} - \mathbf{1}_{K,K}$: same inter-attribute interactions and no intra-attribute interactions;
 - (c) $\mathbb{S} = \mathbf{1}_{K,K}$ a matrix full of ones: full agreement between attributes.
3. Compute $\boldsymbol{\Theta}$ a positive-definite approximation of \mathbf{M} by replacing null and negative eigenvalues by a small constant;
 4. Control the difficulty of the problem with $\gamma > 0$ such that $\boldsymbol{\Theta} = \boldsymbol{\Theta} + \gamma I$;
 5. Draw an i.i.d. n -size sample $\mathbf{X} \in \mathbb{R}^{n \times p^K}$ of $X \sim \mathcal{N}(0, \boldsymbol{\Theta}^{-1})$.

We choose small networks with $p = 40$, with 40 edges on average and vary n from $p/2$ to $2p$. We fix γ to 0.1 and consider cases where the number of attributes is $K = 2, 3$, and 4. We compare our multiattribute approach of neighborhood selection to two baselines:

1. the standard neighborhood selection procedure applied on the data related to each attribute separately: to do so, we separate \mathbf{X} in K data sets $\mathbf{X}^{(1)}, \dots, \mathbf{X}^{(K)}$ all with size $\mathbb{R}^{n \times p}$ and reconstruct one network per attribute. We refer to this method as the *separate* variant.
2. the standard neighborhood selection approach applied on a merge data set, obtained by stacking the data sets $\mathbf{X}^{(1)}, \dots, \mathbf{X}^{(K)}$ of each attribute into a single $\tilde{\mathbf{X}}$ data set in $\mathbb{R}^{nK \times p}$. We refer to this method as the *merge* variant. This method is the exact opposite of the *separate* variant.

We assess the performance of each method in reconstructing the original adjacency matrix \mathbf{A} with the area under ROC curve (AUC). For the *separate* variant, the retained AUC is the AUC averaged over all attributes. We replicate the experiment 100 times.

In Fig. 2, it is clear that aggregation (either by merging data sets or with multiattribute network inference) improves upon a single-attribute approach. Even when there are no inter-attribute interactions (which is barely meaningful towards application to regulatory networks), in which case it is a very good idea to merge the problems together to increase the sample size, our multiattribute approach remains quite competitive and robust. In all other cases, it outperforms the competing approaches.

Illustration: Gene/Protein Regulatory Network Inference As an illustration, we applied our sparse multiattribute GGM approach to reconstruct networks on two large breast cancer data sets from the National Cancer Institute (<https://www.cancer.gov/>)

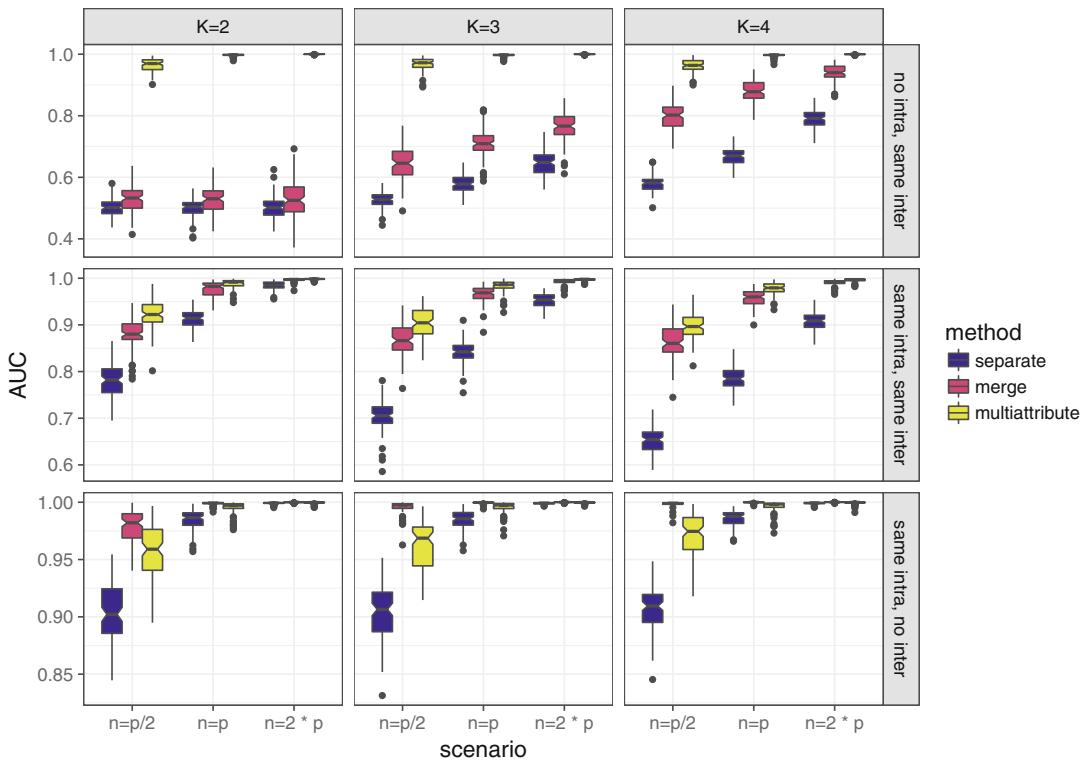


Fig. 2 Simple simulation study for the multiattribute network inference problem: the multiattribute procedure improves over the univariate procedures in every situation when networks are close for each attribute

and the Rational Therapy for Breast Cancer consortium (<http://www.ratherproject.com/>), respectively, referred to as NCI-60 and RATHER hereafter. These data sets contain both proteomic and transcriptomic profiles, respectively, measured with reverse-phase protein arrays (RPPA) and RNA Affymetrix array. We infer the multiattribute network between the subset of molecular entities which is common to the proteins measured by RPPA and the genes measured by RNA array, that we call the *consensus set*: in the NCI-60 cancer line data set [50], a consensus set composed of $p = 91$ proteins and corresponding gene profiles is retained, for the $n = 60$ samples. The RATHER data set [51] contains proteomic and transcriptomic data from $n = 100$ patients for a consensus set of $p = 117$ entities (the data can be downloaded from <https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE66647>).

We infer a sparse GGM for each attribute (gene expression and protein profile), separately to start with, and then get its multiattribute version. We do this on a large grid of the tuning parameter and thus have three families of networks indexed by their number of edges.

Figure 3 demonstrates that our sparse multiattribute method captures the characteristics of both univariate networks, as the

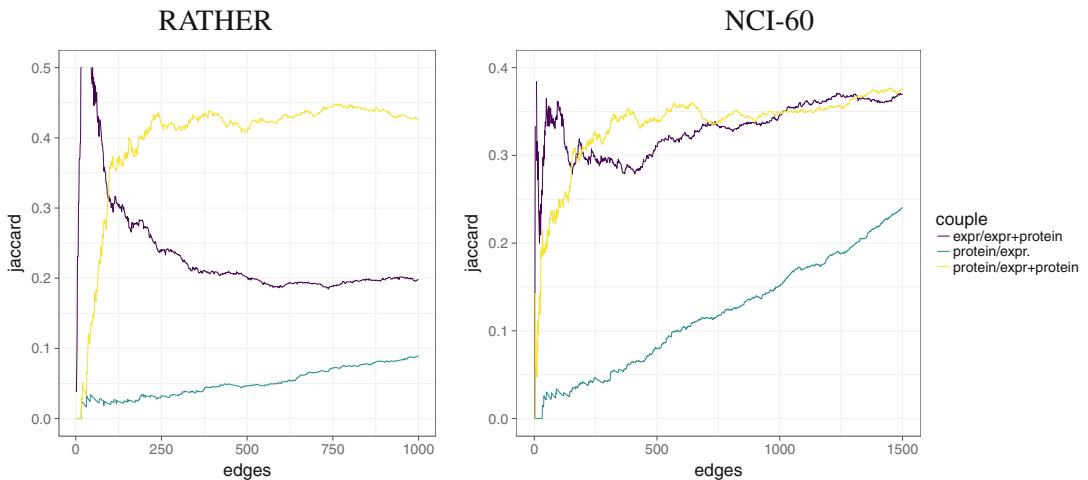


Fig. 3 Jaccard's similarity index $J(A, B) = \frac{|A \cap B|}{|A \cup B|}$ between uni-attribute and multiattribute networks, for RATHER and NCI60 data set: multiattribute networks share a high Jaccard index with both uni-attribute networks

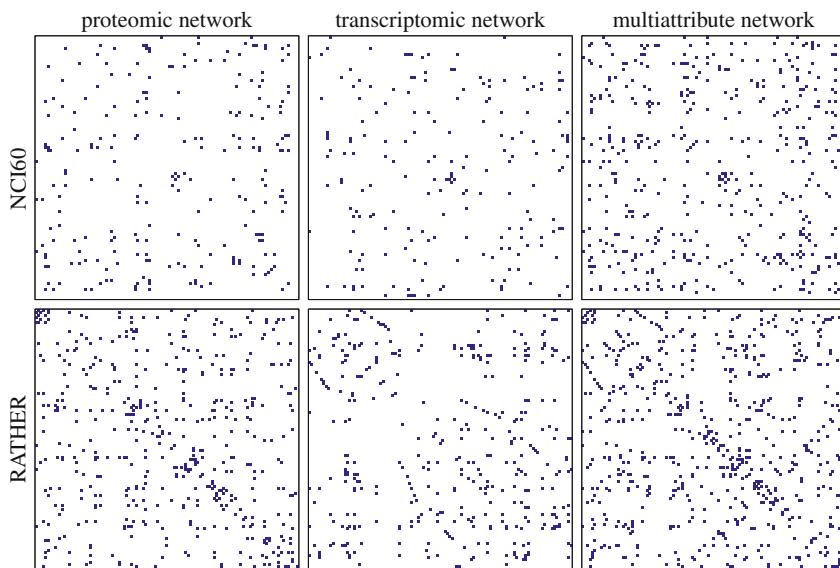


Fig. 4 Uni-attribute and multiattribute networks inferred on both NCI60 and RATHER data sets. The number of neighbors of each entity is chosen by cross-validation. Multiattribute networks catch motifs found in the uni-attribute counterparts

Jaccard similarity index is high between each uni-attribute network and the multiattribute network, while it remains low when comparing uni-attribute networks together.

Figure 4 shows the finally retained networks, where the number of edges is controlled by the tuning parameter λ chosen by 10-fold cross-validation. It is clear that some motifs only present in each uni-attribute networks are caught in their multiattribute counterparts. This tends to prove that the multiattribute version proposes a consensus version of the interactions at hand in the cell, and one which is hopefully more robust to noise.

References

1. Lauritzen S (1996) Graphical models, Oxford statistical science series, vol 17. Clarendon Press, New York. Oxford Science Publications
2. Whittaker J (1990) Graphical models in applied multivariate statistics. Wiley series in probability and mathematical statistics: probability and mathematical statistics. Wiley, Hoboken
3. Meinshausen N, Bühlmann P (2006) High-dimensional graphs and variable selection with the lasso. *Ann Stat* 34(3):1436–1462
4. Friedman J, Hastie T, Tibshirani R (2008) Sparse inverse covariance estimation with the graphical lasso. *Biostatistics* 9(3):432–441
5. Banerjee O, El Ghaoui L, d'Aspremont A (2008) Model selection through sparse maximum likelihood estimation for multivariate Gaussian or binary data. *J Mach Learn Res* 9:485–516
6. Yuan M (2010) Sparse inverse covariance matrix estimation via linear programming. *J Mach Learn Res* 11:2261–2286
7. Cai T, Liu W, Luo X (2011) A constrained l1 minimization approach to sparse precision matrix estimation. *J Am Stat Assoc* 106:594–607
8. Ambroise C, Chiquet J, Matias C (2009) Inferring sparse Gaussian graphical models with latent structure. *Electron J Stat* 3:205–238
9. Chiquet J, Smith A, Grasseau G, Matias C, Ambroise C (2009) SIMoNe: statistical inference for MODular NEtworks. *Bioinformatics* 25(3):417–418
10. Chiquet J, Grandvalet Y, Ambroise C (2011) Inferring multiple graphical models. *Stat Comput* 21(4):537–553
11. Jeanmougin M, Charbonnier C, Guedj M, Chiquet J (2014) Network inference in breast cancer with Gaussian graphical models and extensions. In: Probabilistic graphical models dedicated to applications in genetics, genomics and postgenomics. Oxford University Press, Oxford
12. Giancotti FG (2014) Deregulation of cell signaling in cancer. *FEBS Lett* 588(16):2558–2570
13. Akbani R, Becker KF, Carragher N, Goldstein T, de Koning L, Korf U, Liotta L, Mills GB, Nishizuka SS, Pawlak M et al (2014) Realizing the promise of reverse phase protein arrays for clinical, translational, and basic research: A workshop report the RPPA (reverse phase protein array) society. *Mol Cell Proteomics* 13(7):1625–1643
14. Dempster A (1972) Covariance selection. *Biometrics Spec Multivar Issue* 28:157–175
15. Castelo R, Roverato A (2006) A robust procedure for Gaussian graphical model search from microarray data with p larger than n . *J Mach Learn Res* 7:2621–2650
16. Drton M, Perlman M (2007) Multiple testing and error control in Gaussian graphical model selection. *Stat Sci* 22:430
17. Drton M, Perlman M (2008) A SINful approach to Gaussian graphical model selection. *J Stat Plann Inference* 138(4):1179–1200
18. Kiiveri H (2011) Multivariate analysis of microarray data: differential expression and differential connection. *BMC Bioinf* 12(1):42
19. Wille A, Bühlmann P (2006) Low-order conditional independence graphs for inferring genetic networks. *Stat Appl Genet Mol Biol* 5(1). <https://doi.org/10.2202/1544-6115.1170>
20. Dobra A, Hans C, Jones B, Nevins JR, Yao G, West M (2004) Sparse graphical models for exploring gene expression data. *J Multivar Anal* 90(1):196–212
21. Jones B, Carvalho C, Dobra A, Hans C, Carter C, West M (2005) Experiments in stochastic computation for high-dimensional graphical models. *Stat Sci* 20(4):388–400
22. Rau A, Jaffrézic F, Foulley JL, Doerge R (2012) Reverse engineering gene regulatory networks using approximate Bayesian computation. *Stat Comput* 22(6):1257–1271
23. Schwaller L, Robin S, Stumpf M (2015) Bayesian inference of graphical model structures using trees. arXiv preprint arXiv:150402723
24. Tibshirani R (1996) Regression shrinkage and selection via the lasso. *J R Stat Soc B* 58(1):267–288
25. Yuan M, Lin Y (2007) Model selection and estimation in the Gaussian graphical model. *Biometrika* 94(1):19–35
26. Nesterov Y (2005) Smooth minimization of non-smooth functions. *Math Program* 103(1):127–152
27. Mazumder R, Hastie T (2012) The graphical lasso: new insights and alternatives. *Electron J Stat* 6:2125–2149
28. Witten D, Friedman J, Simon N (2011) New insights and faster computations for the graphical lasso. *J Comput Graph Stat* 20(4):892–900

29. Zhao T, Liu H, Roeder K, Lafferty J, Wasserman L (2014) huge: high-dimensional undirected graph estimation. R package version 1.2.6
30. Hsieh CJ, Sustik M, Dhillon I, Ravikumar P (2014) Quic: quadratic approximation for sparse inverse covariance estimation. *J Mach Learn Res* 15(1):2911–2947
31. Hsieh CJ, Sustik M, Dhillon I, Ravikumar PK, Poldrack R (2013) Big & quic: sparse inverse covariance estimation for a million variables. In: Advances in neural information processing systems (NIPS), pp 3165–3173
32. Ravikumar P, Wainwright M, Raskutti G, Yu B (2011) High-dimensional covariance estimation by minimizing ℓ_1 -penalized log-determinant divergence. *Electron J Stat* 5:935–980
33. Verzelen N (2012) Minimax risks for sparse regressions: ultra-high-dimensional phenomenons. *Electron J Stat* 6:38–90
34. Rocha GV, Zhao P, Yu B (2008) A path following algorithm for sparse pseudo-likelihood inverse covariance estimation (splice). arXiv preprint arXiv:0807.3734
35. Ravikumar P, Wainwright MJ, Lafferty J (2010) High-dimensional Ising model selection using ℓ_1 -regularized logistic regression. *Ann Stat* 38:1287–1319
36. Bach F, Jenatton R, Mairal J, Obozinski G (2012) Optimization with sparsity-inducing penalties. *Found Trends Mach Learn* 4(1):1–106
37. Meinshausen N, Bühlmann P (2010) Stability selection. *J R Stat Soc Ser B* 72: 417–473
38. Villers F, Schaeffer B, Bertin C, Huet S (2008) Assessing the validity domains of graphical Gaussian models in order to infer relationships among components of complex biological systems. *Stat Appl Genet Mol Biol* 7(2). <https://doi.org/10.2202/1544-6115.1371>
39. Zhao P, Yu B (2006) On model selection consistency of Lasso. *J Mach Learn Res* 7:2541–2563
40. Giraud C, Huet S, Verzelen N (2012) High-dimensional regression with unknown variance. *Stat Sci* 27(4):500–518
41. Foygel R, Drton M (2010) Extended Bayesian information criteria for Gaussian graphical models. In: Advances in neural information processing systems (NIPS), pp 2020–2028
42. Giraud C, Huet S, Verzelen N (2012) Graph selection with GGMselect. *Stat Appl Genet Mol Biol* 11(3):1–50
43. Giraud C (2008) Estimation of Gaussian graphs by model selection. *Electron J Stat* 2: 542–563
44. Bach F (2008) Bolasso: model consistent lasso estimation through the bootstrap. In: Proceedings of the 25th international conference on machine learning. ACM, New York, pp 33–40
45. Liu H, Roeder K, Wasserman L (2010) Stability approach to regularization selection (stars) for high dimensional graphical models. In: Advances in neural information processing systems (NIPS), pp 1432–1440
46. Haury AC, Mordelet F, Vera-Licona P, Vert JP (2012) Tigress: trustful inference of gene regulation using stability selection. *BMC Syst Biol* 6(1):145
47. Katenka N, Kolaczyk E (2012) Inference and characterization of multi-attribute networks with application to computational biology. *Ann Appl Stat* 6(3):1068–1094
48. Kolar M, Liu H, Xing E (2014) Graph estimation from multi-attribute data. *J Mach Learn Res* 15(1):1713–1750
49. Boyd S, Vandenberghe L (2006) Convex optimization, 3rd edn. Cambridge University Press, Cambridge
50. Pfister TD, Reinhold WC, Agama K, Gupta S, Khin SA, Kinders RJ, Parchment RE, Tomaszewski JE, Doroshow JH, Pommier Y (2009) Topoisomerase I levels in the NCI-60 cancer cell line panel determined by validated ELISA and microarray analysis and correlation with indenoisoquinoline sensitivity. *Mol Cancer Ther* 8(7):1878–1884
51. Michaut M, Chin SF, Majewski I, Severson TM, Bismeijer T, de Koning L, Peeters JK, Schouten PC, Rueda OM, Bosma AJ et al (2016) Integration of genomic, transcriptomic and proteomic data identifies two biologically distinct subtypes of invasive lobular breast cancer. *Sci Rep* 6:18517



Chapter 7

Integrative Approaches for Inference of Genome-Scale Gene Regulatory Networks

Alireza Fotuhi Siahpirani, Deborah Chasman, and Sushmita Roy

Abstract

Transcriptional regulatory networks specify the regulatory proteins of target genes that control the context-specific expression levels of genes. With our ability to profile the different types of molecular components of cells under different conditions, we are now uniquely positioned to infer regulatory networks in diverse biological contexts such as different cell types, tissues, and time points. In this chapter, we cover two main classes of computational methods to integrate different types of information to infer genome-scale transcriptional regulatory networks. The first class of methods focuses on integrative methods for specifically inferring connections between transcription factors and target genes by combining gene expression data with regulatory edge-specific knowledge. The second class of methods integrates upstream signaling networks with transcriptional regulatory networks by combining gene expression data with protein–protein interaction networks and proteomic datasets. We conclude with a section on practical applications of a network inference algorithm to infer a genome-scale regulatory network.

Key words Gene regulation, Regulatory networks, Integrative network inference, Probabilistic graphical models

1 Introduction

Regulatory networks specify the regulatory proteins (e.g., transcription factors and signaling proteins) [1–3] that control context-specific expression levels of genes. Identification of these regulatory networks is important to understand the underlying mechanisms of complex biological processes such as response to stress, cell differentiation, and occurrences of diseases [4–6]. Inferring regulatory networks is a central problem in the field of systems biology

The authors Alireza Fotuhi Siahpirani and Deborah Chasman contributed equally.

and remains an open challenge using both experimental and computational approaches. Although the majority of computational methods are based on expression, recent work in this area aims to integrate gene expression with other types of data. In this chapter, we cover recent computational approaches for performing integrative regulatory network inference that combine different types of non-gene expression (mRNA) level data, e.g., protein–DNA or protein–protein interactions, proteomic measurements with gene expression data. See also Chapter 6 for an alternative perspective on the problem of network inference from multiple data types. We focus on probabilistic graphical model-based representation of regulatory networks. We discuss two main strategies: the first class of methods is focused on predicting edges between transcription factor and target genes by leveraging additional information about regulatory relationships between regulators and target genes, the second class of methods integrates additional players, e.g., signaling proteins and their interactions to better model gene regulation. We first provide a brief background on transcriptional control of gene regulation, probabilistic models to represent gene regulatory networks and briefly review expression-based network inference. We next describe different computational strategies to infer regulatory networks while integrating different types of data. In the end, we provide a “user’s guide” to infer genome-scale integrative regulatory networks.

2 Background on Gene Regulation and Gene Regulatory Networks

2.1 Gene Regulation

Gene expression is controlled at many levels [7], including transcriptional and post-translational levels. At the transcriptional level gene expression is controlled by transcription factors, chromatin remodelers, and chromatin organization that affect the accessibility of transcription factors. At the post-translational level gene expression is controlled via signal transduction affecting protein stability and activity. Recent advances in omic technologies are enabling us to measure the components of different levels of regulation. Regulatory networks describe the connections between “regulatory proteins” and target genes, whose expression is determined by the activity levels of the regulatory proteins, such as transcription factors and signaling proteins (Fig. 1a). Transcription factors bind to regulatory regions of these genes and attract RNA polymerase that in turn initiates the transcription of the gene (Fig. 1a). These regulatory regions could be immediately upstream of the transcription start site (promoters) or further (more than 10Kb) upstream or downstream (enhancers). Signaling proteins operate via physical protein–protein interactions with other signaling proteins transmitting upstream environmental signals to downstream changes in transcription factors via a series of post-translational modifications. This chapter will focus on transcriptional regulatory

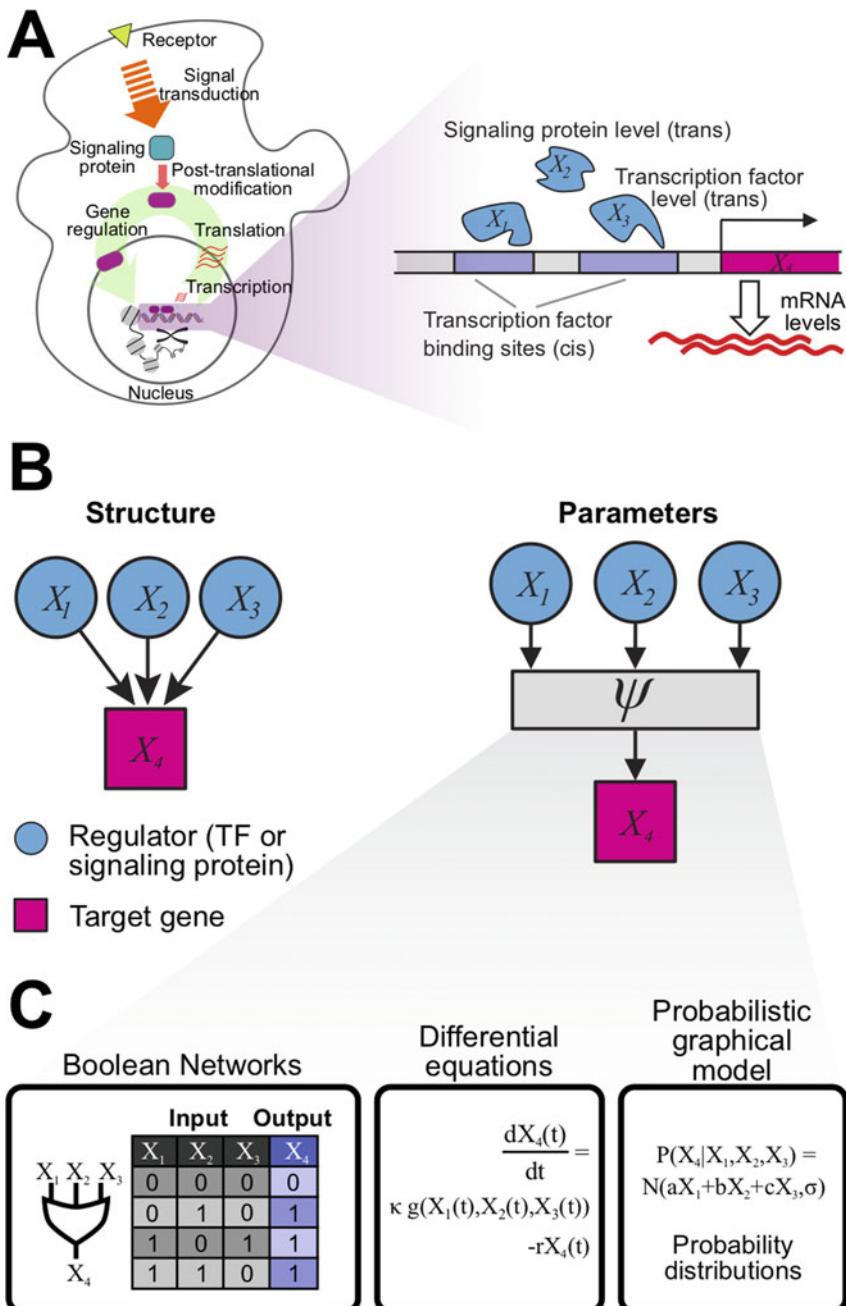


Fig. 1 (a) Overview of gene regulation within a cell. On the left is shown a cartoon of a cell with a signaling cascade ending in a transcription factor (TF, pink rounded rectangles). This transcription factor along with another TF are bound to the promoter of a gene, thereby inducing transcription. On the right is shown a simplified model with the key players, the transcription factors (X_1 and X_3) and signaling protein (X_2), that are involved in the regulation of the expression of the gene X_4 . (b) Regulatory networks describe the relationships between regulators and their targets. These models have two components, the structure (who regulates whom) and the function (how changes in regulators affect the expression of target). (c) Different types of mathematical models can be used for the function of a regulatory network. Shown are three examples of such functions, a boolean model, an ordinary differential equation, and a probabilistic model

networks and signaling networks. In both these networks, we have proteins that play a regulatory role, namely transcription factors and signaling proteins (Fig. 1a).

2.2 Mathematical Models for Representing Gene Regulatory Networks

A regulatory network is defined by two components, a *structure* and a *function* (Fig. 1b). The structure specifies which regulators regulate which genes, while the function specifies how the regulators regulate the expression of a gene. Regulatory networks can be context-specific, meaning that the structure and function could change depending upon the context (e.g., different cell type, tissue, disease, or environmental condition). A comprehensive characterization of regulatory networks requires that we (1) infer the network structure specifying which regulators regulate which genes, (2) infer the network parameters that define quantitative relationships between the regulator state and the expression of a target gene. Both structure and parameters of the network are needed to be inferred to gain a mechanistic understanding of how regulatory networks drive context-specific patterns. Additionally, parameters of the network are essential to predict global cellular phenotypes including mRNA levels, which serve as “easy-to-measure” readouts to validate predicted genome-wide expression levels under different *in silico* perturbations. Gene regulatory networks have been represented in the literature using different types of mathematical models [2, 8]. A few major categories are: (a) boolean networks, (b) differential equations, (c) probabilistic graphical models (Fig. 1c). Each of these models differ in their encoding of a regulatory protein and target gene and the mathematical function used to represent the regulatory function. In particular, in Boolean networks, the network nodes are binary variables taking on “0” or “1” values and the functions are logic gates such as AND and OR gates [9]. In differential equation-based models, the mathematical functions are ordinary differential equations [8]. In probabilistic graphical models, the nodes represent random variables and the functions are represented by local probabilistic functions, such as conditional probability distributions (Fig. 1c).

2.3 Experimental Techniques to Construct Regulatory Networks

There are different experimental techniques that have been used to infer regulatory connections between regulators and target genes. Commonly used experimental methods to infer the gene targets of a transcription factor include genetic perturbation (knock down, knock out, and over-expression [10, 11]) assays, ChIP-chip/-seq experiments, and accessibility assays. Genetic perturbation assays measure the changes in behavior of genes as a result of disrupting a regulatory gene. These methods measure the genome-wide expression profile of genes and use statistical methods to identify which genes have significantly changed their expression patterns (are differentially expressed) as a result of disrupting the expression

of the tested regulator. Chromatin immunoprecipitation (ChIP)-based methods (ChIP-chip or ChIP-seq [12, 13]) are another type of experimental technique that measure binding of a specific protein (for example, a transcription factor) on DNA (for example, promoter of genes). Unlike genetic perturbation experiments, this technique measures direct binding of transcription factors, although some of the binding events can be non-functional. These methods are all regulator-centric, that is they examine one transcription factor (TF) at a time. Another type of experimental method focuses on identifying open chromatin regions in a genome (e.g., using DNase I-seq [14] or ATAC-seq [15] assays) followed by searching for binding sites of known transcription factors. Searching for potential binding sites of transcription factors with known binding affinity in these accessible regions can be a less expensive alternative to TF ChIP experiments. These methods can be used for TFs with known binding affinity and as the number of TFs with known affinity increases, these methods will become more useful for creating the context-specific structure of transcription factor regulatory networks.

Genetic perturbation experiments are also applicable to infer signaling regulatory networks. For example, by knocking out a signaling protein and measuring the change in the phosphoproteome [16], we can infer potential protein targets of the kinase. More often, for signaling networks kinase inhibitors can be used to change the level of a kinase and infer targets based on significant changes in the phosphoproteome. Commonly used techniques specifically detecting protein–protein interactions, of which signaling interactions are a subset, are of two main classes: affinity mass-spectrometry (MS) and two-hybrid assays [17, 18]. Affinity MS assays are best suited for detecting protein complexes while two-hybrid assays are useful for detecting direct protein–protein interactions.

Although experimental methods are more precise than computational methods, they also have limitations and are expensive. Computational methods offer economical alternatives which can be used in concert with experimental methods in an iterative framework to construct a more accurate network than what is possible from one type method alone.

3 Expression-Based Network Inference of Regulatory Networks

Expression-based network inference is perhaps the most straightforward and commonly used approach to infer a genome-scale context-specific regulatory network. The overall intuition of this approach is that gene expression levels are the outputs of the network and measuring these levels across multiple states of the network can inform us of the regulatory dependencies. In this

approach, we collect large amounts of genome-wide gene expression profiles in the context (e.g., cell type) of interest and apply a network inference algorithm to it. There are a large number of methods for inferring networks from expression data alone (*see* Table 1 for example methods and Marbach et al. [30] for an overview of different purely expression-based methods). These algorithms can be grouped into two major classes based on the learning paradigm: (a) per-gene methods [25, 26, 30, 31], also referred to as “direct” network inference methods [32], and (b) per-module methods [20, 24, 32]). Per-gene methods are those that infer the regulatory program of one gene at a time and most algorithms fall into this category. Per-module methods explicitly account for the “modular” nature of gene regulatory networks, where groups of genes are co-expressed in a module and tend to share their regulators. Per-gene methods are able to learn precise regulatory programs for individual genes, but do not directly inform us of the modular organization of regulatory networks. Per-module methods assume that all genes in the module have the same regulators. This is a simplifying assumption and is unlikely to be true, but the module aspect of these methods is beneficial to learn more interpretable networks with relatively small number of samples. Network inference algorithms can also be grouped based on the form of the mathematical function used to describe the relationships among random variables and there are three main categories (Table 1): (a) Information theoretic methods, (b) probabilistic graphical model-based methods, (c) differential equations-based methods. Information theoretic methods make use of mutual information to define the dependency between a regulator and target gene. Differential equations-based methods aim to model the temporal kinetics of a gene’s transcription rate as a function of the expression level of its regulators. Probabilistic models estimate a full joint distribution of the graph and are described in more detail in this chapter.

3.1 Probabilistic Graphical Model-Based Representation and Learning of Gene Regulatory Networks

Probabilistic graphical models (PGMs) are models that can be used to tractably represent joint probability distributions over a large number of random variables. PGMs have two components, a graph structure that specifies the statistical dependencies among the random variables and parameters that describe the nature of the dependency structure among the random variables. PGMs are powerful models for representing regulatory networks because these models naturally capture both the structure and the parameters of a regulatory network [1, 31, 33, 34]. In a PGM representation of a regulatory network, each node (gene) of the network is modeled as a random variable, and the expression measurements of each gene in different time points/conditions are considered as observations of the corresponding random variable. The graph structure encodes different statistical independencies among the random variables, which is useful to tractably compute

Table 1
Examples of different classes of purely expression-based network inference methods

Method	Description	Per-gene or -module	Model type
Sparse candidate[19]	The “sparse candidate” is a fast Bayesian network learning algorithm that works by limiting the maximum number of candidate parents of each node and only considering the best potential parents (defined based on the strength of their statistical dependency to the target node).	Per-gene	Bayesian network
Module network [20]	Instead of creating a regulatory program for each target gene, “Module network” groups the genes based on their expression profile and infers a regulatory program for each module.	Per-module	Bayesian network
ARACNE [21]	This method calculates the mutual information between all pairs of genes in the network, and then filters these edges, by considering each triplet of nodes and removing the edge with lowest MI (based on the assumption that that edge corresponds to an indirect link).	Per-gene	Information theoretic
CLR[22]	Context Likelihood of Relatedness (CLR) calculates the mutual information between a regulator-target gene pair, and scores this value based on a background distribution produced from mutual information of the regulator and target to all other genes in the networks.	Per-gene	Information theoretic
WGNA[23]	This method creates an undirected network where an edge between two genes is weighted by the absolute value of correlation of expression profile of those two genes, to the power of $\beta \geq 1$ (raising to the power of β is used as a soft thresholding).	Per-gene	Correlation

Table 1
(continued)

Method	Description	Per-gene or -module	Model type
LeMoNE[24]	This method uses a similar approach as Module network, however, instead of a direct optimization, they infer an ensemble of statistical models and produce a consensus (by ensemble averaging) for both modules and regulatory edges.	Per-module	Bayesian network
GENIE3[25]	This method uses Random Forest and Extra-Trees to create the regulatory program of each target gene. In each regression tree, regulators are used as features, and the expression of the target gene is modeled as a non-linear function of the expression of regulators.	Per-gene	Dependency network
TIGRESS[26]	This method uses LASSO regression [27] in a stability selection framework [28] to infer the regulatory program of each target gene.	Per-gene	Dependency network
MERLIN[29]	MERLIN is based on probabilistic graphical model that uses a modularity prior to combine the strength of both per-gene and per-module models. MERLIN iteratively updates the regulatory program of each gene to have a similar (but not necessarily identical) program to other genes in that module.	Both	Dependency network

The selected methods represent examples of per-gene or per-module methods, as well as methods that are information theoretic or based on probabilistic graphical models.

the full joint distribution from more tractable smaller functions. Two well-known classes of PGMs are directed acyclic graphs (Bayesian networks) and undirected graphical models (also known as Markov networks or random fields). In the context of regulatory networks, there is a third class of PGMs, called dependency networks, introduced by Heckerman et al. [35], which are learned by estimating the best set of predictor variables for every random variable. “Consistent” dependency networks are a subset of the generic dependency networks and are equivalent to Markov networks. The problem of inferring the regulatory network can be formulated as finding the dependency structure and estimating the conditional distributions (parameters) between the observed variables. Both Bayesian networks and dependency networks have been used in the literature to model regulatory networks.

Bayesian networks were used to model a transcriptional regulatory network by Friedman and colleagues [33]. In a Bayesian network the graph is a directed acyclic graph (DAG), which is a graph without any cycles and the parameters are associated with conditional probability distributions for each variable given its parents. The assumption is that variable X_i is independent of its non-descendants given its parents, \mathbf{Pa}_i . In a Bayesian network, the joint distribution can be written as a product of the per-variable conditional distributions,

$$P(X_1, \dots, X_n) = \prod_{i=1}^n P(X_i|\mathbf{Pa}_i)$$

which enables one to search the space of graphs efficiently by searching for the best parent set of each random variable independently (subject to the DAG constraint). A few common implementations of Bayesian networks for modeling regulatory networks employ the sparse candidate algorithm [19], which restricts the search to parent sets up to a maximum size, and dynamic Bayesian networks [36, 37].

Dependency networks are the second class of PGMs commonly used to represent regulatory networks. As in Bayesian networks, these models require one to learn a predictive distribution of each variable given its neighbor set. In these models, a regulatory network is constructed by solving a set of per-gene regression problems, where the expression of a gene is predicted as a function of its upstream regulators. Unlike Bayesian networks, the resulting network does not need to be acyclic, enabling these models to represent feedback loops. However, without the DAG constraint, the product of the target conditional probability distributions may not form a valid joint probability distribution, and hence learning in these models entails optimizing the “pseudo likelihood” [35]. A few example methods that are based on dependency networks are TIGRESS [26], Inferelator [38, 39], MERLIN [29], and GENIE3 [25].

3.2 Learning PGMs from Expression Data

A gene expression dataset is denoted as $\mathcal{D} = (\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^{|\mathcal{D}|})$, where $\mathbf{x}^d = (x_1^d, x_2^d, \dots, x_n^d)$ is the joint assignment of expression values of all the n genes in the d th sample in the dataset. The problem of network inference can be viewed as searching for the model (including the structure of the network, \mathcal{G} , and model parameters Θ) that maximizes the likelihood of the model given the data, $P(\mathcal{G}, \Theta | \mathcal{D})$. By Bayes rule we can write this as:

$$P(\mathcal{G}, \Theta | \mathcal{D}) \propto P(\mathcal{D} | \mathcal{G}, \Theta) P(\Theta | \mathcal{G}) P(\mathcal{G})$$

The term $P(\mathcal{G})$ indicates the prior distribution over the network structure, and in the next section, we will discuss how we can incorporate our prior knowledge in this term. $P(\Theta | \mathcal{G})$ is a prior over the parameters. In the uniform prior case, Θ is set to its maximum likelihood setting. If an informative prior is available, Θ can be set to the maximum a posteriori value.

In a Bayesian network, the search over models is done by directly optimizing the data likelihood. By i.i.d assumption (that the samples are independent measurements from identical distributions) we can split the data likelihood term on the samples:

$$P(\mathcal{D} | \mathcal{G}, \Theta) = \prod_{d=1}^{|\mathcal{D}|} P(\mathbf{x}^d | \mathcal{G}, \Theta).$$

Each $P(\mathbf{x}^d | \mathcal{G}, \Theta)$ term can be computed using

$$P(x_1^d, \dots, x_n^d | \mathcal{G}, \Theta) = \prod_{i=1}^n P(X_i = x_i^d | \mathbf{Pa}_i = x_{\mathbf{Pa}_i}^d),$$

where $x_{\mathbf{Pa}_i}^d$ corresponds to the joint assignment to X_i 's parents, \mathbf{Pa}_i , in the d th sample.

In a dependency network, the network structure is learned by optimizing the pseudo likelihood, which also decomposes as a product over individual conditional distributions for each random variable X_i and a set of regulators, \mathbf{R}_i , that best predict the expression of a gene:

$$P(X_i | \mathbf{R}_i, \theta_i) = \prod_{d=1}^{|\mathcal{D}|} P(X_i = x_i^d | \mathbf{R}_i = x_{\mathbf{R}_i}^d, \theta_i).$$

This can be accomplished by learning a regression model (linear or non-linear) per target gene. However, unlike Bayesian networks, where the DAG constraint enables us to compute the joint probability distribution from the product of the conditional distributions, multiplying these local probability distributions does not result in a valid joint distribution and instead refers to the pseudo likelihood [35].

Given a particular structure, there can be different probability distributions that could be used to represent the conditional distribution of each random variable. A common assumption in these models is that the distribution of the data is normal. In this case, the term $P(X_i = x_i^d | \mathbf{R}_i = x_{\mathbf{R}_i}^d, \theta_i)$ can be estimated using a linear Gaussian, $P(X_i | X_1, \dots, X_k) \sim N(\beta_0 + \sum_{j=1}^k \beta_j X_j, \sigma^2)$, where β_j is the regression coefficient. Thus the expression of X_i is assumed to be a linear combination of expression of its regulators, X_1, \dots, X_k or as a multivariate conditional Gaussian distribution.

Searching over the space of possible structures is a computationally intractable problem and several greedy and heuristic algorithms have been developed [40, 41]. In all network inference search algorithms, one selects a move from a set of moves, each move representing a change to the current graph structure to a neighboring structure in the space of all possible structures. In a greedy search, one selects the graph structure that has the best score, where the score of the move is derived from the data likelihood or pseudo likelihood. Because this greedy approach can converge to a local optimum, search algorithms that do not always choose the move with highest score (for example, simulated annealing or Markov Chain Monte Carlo methods) can be used.

4 Incorporating Priors for Integrative Inference of Gene Regulatory Networks

Although a large number of methods have been developed for inferring gene regulatory networks from gene expression, expression alone is not sufficient to infer a regulatory network accurately. There are two main reasons why expression alone may not be sufficient. First, the network inference problem is computationally difficult, that is there are a large number of possible network structures that could be explained by a set of transcriptome profiles. Second, mRNA levels only capture one component of the gene regulation machinery and may not accurately reflect the true activity level of proteins, which are subject to post-translational regulation. To address this problem, one direction of work has focused on using prior knowledge to constrain the structure of the network, which we discuss in detail in this section. A second direction of work has been to integrate other types of regulatory connections by either incorporating the signaling network or other types of omic datasets, which we describe in the next section.

In a probabilistic graphical model, constraints on the structure can be incorporated by defining probabilistic priors on the inferred network [29, 36, 37, 39, 42, 43]. The prior-based framework is a powerful approach for network inference and offers a principled framework to integrate different types of evidences supporting the presence of a regulatory edge. Priors have been introduced for Bayesian networks [36, 37] as well as for dependency networks [39, 43]. Broadly speaking, there are two classes of methods that

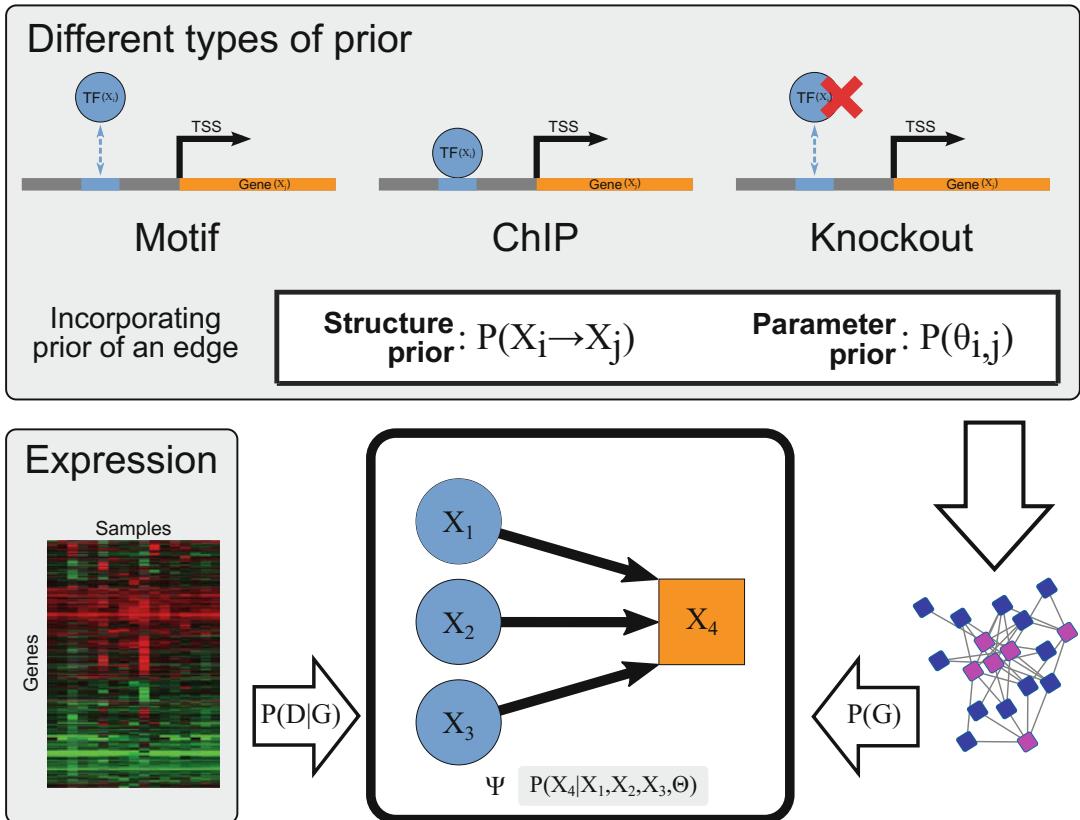


Fig. 2 Incorporating prior knowledge in expression-based network inference. X_i and X_j denote a regulator (such as a transcription factor) and a target gene, respectively. For each candidate edge $X_i \rightarrow X_j$, different sources of prior networks can be used. The figure shows three different types of prior networks: ChIP-chip/seq, Motif, and Knockout. D is the expression dataset and G is the graph that represents the regulatory network to be inferred. Prior knowledge can be encoded as probability distribution over the graph $P(G)$. Different methods for incorporating priors can be grouped into those that define a prior on graph structure (structure prior) and those that define a prior on the parameters (parameter prior)

have used priors, the *structure prior* approach and the *parameter prior* approach (Fig. 2). The *structure prior* approach extends the prior on graphs $P(\mathcal{G})$, while the *parameter prior* approach defines the prior on parameters $P(\Theta)$. The structure prior approach is more general, but the learning problem is harder than the parameter prior approach. Examples of the structure prior approach are [36, 37, 43] and parameter prior approach are [39, 42], some of which we describe in detail in the next section.

4.1 Prior-Based Framework for Bayesian Networks

Several strategies have been proposed to impose constraints on the graph structure by using a prior on the graph, $P(\mathcal{G})$. The overall goal in the structure prior-based approach is to maximize $P(\mathcal{G}, \Theta | \mathcal{D})$, the likelihood of structure and parameters given the data, which by Bayes rule can be written as:

$$P(\mathcal{G}, \Theta | \mathcal{D}) \propto P(\mathcal{D} | \mathcal{G}, \Theta) P(\Theta | \mathcal{G}) P(\mathcal{G}).$$

Here $P(\mathcal{G})$ is defined as the prior on the graph structure. Werhli and Husmeier [36] use an energy function to incorporate prior knowledge of regulatory relationships. Assume the inferred network is represented as a binary adjacency matrix \mathcal{G} , where $\mathcal{G}_{i,j}$ is 0 if there is no edge between X_i and X_j and is 1 if there is an edge. Prior knowledge of the network structure is represented as an adjacency matrix B , where $0 \leq B_{i,j} < 0.5$ encodes the belief that there is no edge between X_i and X_j (closer to 0 means higher confidence in the absence of the edge), $0.5 < B_{i,j} \leq 1$ encodes the belief that there is an edge between X_i and X_j (closer to 1 means higher confidence in the presence of the edge), and $B_{i,j} = 0.5$ indicates the absence of any prior knowledge of presence or absence of that edge. The energy of the graph \mathcal{G} is defined as $E(\mathcal{G}) = \sum_{i,j=1}^n |B_{i,j} - \mathcal{G}_{i,j}|$, and the prior probability of the graph is specified as:

$$P(\mathcal{G} | \beta) = \frac{\exp(-\beta E(\mathcal{G}))}{Z(\beta)}$$

where $Z(\beta) = \sum_{\mathcal{G}} \exp(-\beta E(\mathcal{G}))$ is the partition function. The energy measures the agreement between the prior knowledge and the inferred graph: if an edge is present or absent in both \mathcal{G} and B we will not add anything to the energy of the graph, but if an edge is present in one and absent in the other, the energy of the graph increases, and higher energy results in lower probability. β is a hyper-parameter to tune the influence of prior knowledge ($\beta = 0$ means no effect, and higher β means more effect). Werhli and Husmeier estimate the β as part of their MCMC-based inference process, where the acceptance probability, $A(\beta_{new} | \beta_{old})$, of the new value of β is defined based on the Metropolis–Hastings update rule (assuming uniform prior on β and symmetric proposal distribution):

$$A(\beta_{new} | \beta_{old}) = \min \left\{ \frac{P(\mathcal{G} | \beta_{new})}{P(\mathcal{G} | \beta_{old})}, 1 \right\}$$

Hill et al. [37] use a similar formulation to incorporate prior knowledge. Let E^* denote the set of true edges based on prior knowledge and $E(\mathcal{G})$ be the set of edges in the inferred network. By defining $f(\mathcal{G}) = -|E(\mathcal{G}) \setminus E^*|$ we can define the prior probability of the graph as:

$$P(\mathcal{G}) \propto \exp(\lambda f(\mathcal{G}))$$

where λ is the hyper-parameter that tunes the effect of prior. This formulation penalizes the edges in \mathcal{G} that are not in the prior set,

but does not award the edges that are. Hill et al. set the hyper-parameter using an empirical Bayes approach.

4.2 Prior-Based Framework for Dependency Networks

A prior-based framework has also been used for dependency networks. One strategy is to use “structure priors” similar to the approaches described in the previous section. This approach is used in the MERLIN-P method [29, 43], which is a dependency network-based method and incorporates prior knowledge about the regulatory network (in the form of motif, ChIP-chip/seq, gene knockout experiments) to create a more accurate regulatory network. In addition to using priors for integrating different data types, MERLIN-P [29] also uses a graph prior to learn a “modular” regulatory network, which combines the strength of per-gene methods and per-module methods (Table 1). By adding the modularity prior, it is more likely for genes with similar expression patterns to have similar (and not necessarily identical) regulatory programs. In MERLIN-P, the prior distribution over a graph is described as

$$P(\mathcal{G}) = \prod_{X_j \rightarrow X_i \in \mathcal{G}} P(X_j \rightarrow X_i) \prod_{X_j \rightarrow X_i \notin \mathcal{G}} 1 - P(X_j \rightarrow X_i)$$

where $P(X_j \rightarrow X_i)$ is our prior belief in the edge $X_j \rightarrow X_i$. The edge prior is further defined as follows:

$$P(X_j \rightarrow X_i) = \frac{1}{1 + \exp(-(p + \beta^R f_{j,i} + \sum_k \beta^k \times w_{j,i}^k))},$$

where p denotes the sparsity parameter, β^R indicates the modularity parameter, and $f_{j,i}$ specifies the tendency of a regulator X_j to regulate other genes in X_i 's module. We define $f_{j,i} = \frac{d_{j,M_i}}{d_j}$ where d_{j,M_i} is the number of targets of X_j in X_i 's module, and d_j is total number of targets of X_j . β^k controls the importance of the k^{th} prior network, and $w_{j,i}^k$ quantifies our confidence in the edge $X_j \rightarrow X_i$ in the k^{th} prior network. Increasing β^k and $w_{j,i}^k$ increases the prior probability of adding the edge $X_j \rightarrow X_i$ to the network. Examples of the different types of regulatory evidences that can be integrated in this approach include sequence-specific motif instances within the promoter of a target gene, the presence of a transcription factor binding on the target gene, and the measured effect on expression of genes after TF knockout or knockdown (Fig. 2).

The MERLIN-P network inference algorithm iterates between two steps: (1) update the graph structure given the current module assignments, (2) update the module assignments given the current graph structure [29]. In the first step, the algorithm performs a greedy score-based search to add the edge that improve the overall score of the network. In the second step, the algorithm uses the

regulatory program inferred in the previous step and similarity in expression profiles to update the module assignments. The algorithm starts with a given initial module assignment as input and iterates between these two steps until convergence (the delta pseudo likelihood of the model becomes lower than a threshold) or for a fixed number of iterations. Further details of how the modules are defined based on co-expression and co-regulation and how it is iteratively updated are described in [29].

A second approach is based on the Inferelator algorithm by Greenfield et al. [39], which uses a regularized linear regression model to infer the regulatory network. Inferelator imposes a prior on the network parameters, namely the regression weights capturing the relationship between the regulator and the target gene. The overall idea of this approach is to learn a “sparse” network by imposing a penalty on the regression weights. The prior knowledge is incorporated in Inferelator by reducing the amount of penalty. Inferelator proposed two strategies: Modified Elastic Net (MEN) and Bayesian Best Subset Regression (BBSR). In the MEN regression approach [44], Greenfield et al. combine l_1 (sparsity) and l_2 (smoothness) penalty. Given the response variable X_i (either from the time-series formulation or steady-state formulation), the elastic net regression minimizes the following objective:

$$\sum_{d=1}^{|D|} \left(x_i^d - \sum_{j \in R_i} \beta_{i,j} x_j^d \right)^2 + \lambda_1 \sum_{j \in R_i} |\beta_{i,j}| + \lambda_2 \sum_{j \in R_i} \beta_{i,j}^2$$

where λ_1 and λ_2 are the l_1 and l_2 penalty hyper-parameters, respectively. l_1 enables us to learn a sparse model and l_2 is useful for correlated regulators. Prior knowledge of the presence of an edge can be imposed by changing this penalty term of that edge. For an edge between X_j and X_i , let $\theta_{i,j}$ be a small value < 1 if the edge is part of our prior set, and 1 otherwise. In the MEN formulation, the new regularized regression with this prior knowledge is written as:

$$\sum_{d=1}^{|D|} \left(x_i^d - \sum_{j \in R_i} \beta_{i,j} x_j^d \right)^2 + \lambda_1 \sum_{j \in R_i} |\theta_{i,j} \beta_{i,j}| + \lambda_2 \sum_{j \in R_i} \beta_{i,j}^2$$

Edges in the prior set have a small $\theta_{i,j}$ and, therefore, will be penalized less than edges without any prior knowledge. Greenfield et al. used a 10-fold cross-validation to select the λ 's that minimize the prediction error, and used a grid search to find a value of θ which increases the AUPR of inferred network.

In the BBSR approach, the prior knowledge is incorporated in the prior distribution of the regression weights. Let y denote the

expression levels of the i th gene, $y = (x_i^1, x_i^2, \dots, x_i^{|\mathcal{D}|})^\top$. BBSR uses a probabilistic model of expression as follows:

$$(y|\beta, \sigma^2, \mathcal{D}_{\mathbf{R}_i}^T) \propto N_{|\mathcal{D}|}(\mathcal{D}_{\mathbf{R}_i}^T \beta, \sigma^2 I)$$

where $N_{|\mathcal{D}|}$ is multivariate normal, y is the response variable, $\mathcal{D}_{\mathbf{R}_i}^T$ is a $|\mathcal{D}| \times |\mathbf{R}_i|$ matrix where rows correspond to samples and columns correspond to regulators, and β is the vector of the regression coefficients. In other words, the predicted response $\mathcal{D}_{\mathbf{R}_i}^T \beta$ is the mean of the multivariate normal, and σ^2 is the variance of the normal. The prior over the regression weights is specified using the Zellner's \mathcal{J} prior:

$$p(\beta|\sigma^2) \propto N(\beta^0, \mathcal{J}(\mathcal{D}_{\mathbf{R}_i} \mathcal{D}_{\mathbf{R}_i}^T)^{-1} \sigma^2)$$

where β^0 represents our prior belief on β , and \mathcal{J} is the hyper-parameter to tune the effect of the prior. When \mathcal{J} is close to 0 the distribution of β will be centered around β^0 (biasing the method to choose β based on our prior knowledge), and when \mathcal{J} is large, β will be centered around the ordinary least squares solution (biasing the method to choose β solely based on the data). To incorporate prior knowledge of the network, Greenfield et al. change \mathcal{J} into a vector $\bar{\mathcal{J}}$ with one dimension for each regulator. For those regulators that have prior knowledge, $\bar{\mathcal{J}}$'s entries are set to \mathcal{J} , while those regulators without prior are set to $\frac{1}{\mathcal{J}}$. A value of $\mathcal{J} > 1$ would enable regulators with priors to shift away from β_0 . The BBSR algorithm searches over all possible subsets of regulators and uses the Bayesian information criterion (BIC) to pick the best set of regulators for a gene. However, since this can become prohibitively expensive as the number of regulators grow, a filtering step is used to choose the k most promising regulators and therefore limit the search space (all 2^k possible models). In Inferelator, k was set to 10.

Both the MERLIN-P and Inferelator algorithms were compared against methods that do not use prior knowledge and were found to have significantly improved agreement for network structure recovery [43].

5 Integrating Signaling Networks with Gene Expression Data

Gene expression levels are the integrated outputs of transcriptional and signaling networks. Explicit modeling of the signaling network may therefore capture a more realistic picture of gene regulation. To that end, PGM-based methods have been developed to explicitly integrate other data types upstream of transcriptional regulation, including protein levels, protein–protein interactions, and gene sets (or hits) identified by functional screens. In this

section, we describe approaches that combine transcriptional regulatory network learning with an explicit physical signaling network component (Fig. 3). Physical interactions from protein–protein and protein–DNA interaction screens are used as a scaffold to posit the mechanistic connections between the upstream signaling proteins and transcription factors. The signaling networks can also predict additional regulatory genes that cannot be inferred from variation in expression data alone. We highlight three methods that use different strategies for inferring transcriptional and signaling networks in tandem: (1) Physical Module Networks (PMNs, Fig. 3b) [45], (2) Signaling and Dynamic Regulatory Events Miner (SDREM, Fig. 3c) [46, 47], and (3) stepwise integration of signaling and regulatory networks (Fig. 3d) [49].

5.1 Physical Module Networks (PMNs)

PMNs extend Module Networks [20] by integrating protein–protein and protein–DNA network data to identify the physical pathways underlying the statistical regulatory network model (Fig. 3b). An inferred PMN \mathcal{P} consists of two components $\mathcal{P} = \langle \mathcal{M}, \mathcal{I} \rangle$. The Module Network \mathcal{M} is a set of gene expression modules with regulators for each module. The Interaction Graph \mathcal{I} is a connected set of physical interactions, including protein–protein, protein–DNA, and DNA to encoded protein product that describe the physical interactions from signaling proteins to transcription factors to target gene modules. \mathcal{M} and \mathcal{I} must be *consistent* with each other and are considered consistent if each regulator in \mathcal{M} can be connected to its module genes by an acyclic regulatory path in \mathcal{I} .

The score for a consistent PMN decomposes over contributions from \mathcal{M} and \mathcal{I} , which are assumed to be independent. Let D be the input data, composed of expression data D_X and interaction data D_I .

$$\begin{aligned}\text{Score}(\mathcal{P}|D) &= \log P(\mathcal{M})P(D_X|\mathcal{M}) + \log P(\mathcal{I})P(D_I|\mathcal{I}) \\ &= \text{Score}(\mathcal{M}: D_X) + \text{Score}(\mathcal{I}: D_I)\end{aligned}$$

The first term is the log likelihood of \mathcal{M} [20]. The second, $\text{Score}(\mathcal{I}: D_I)$, after some simplification, decomposes over the individual edges that have been added to \mathcal{I} :

$$\text{Score}(\mathcal{I}: D_I) = \sum_{e \in \mathcal{I}} W_e$$

where for each possible edge e , W_e is defined as

$$\begin{aligned}W_e &= \log \frac{P(d_e|I_e = 1)P(I_e = 1)}{P(d_e|I_e = 0)P(I_e = 0)} \\ P(d_e = 1|I_e = 1) &= \omega e^{-\omega p_e}\end{aligned}$$

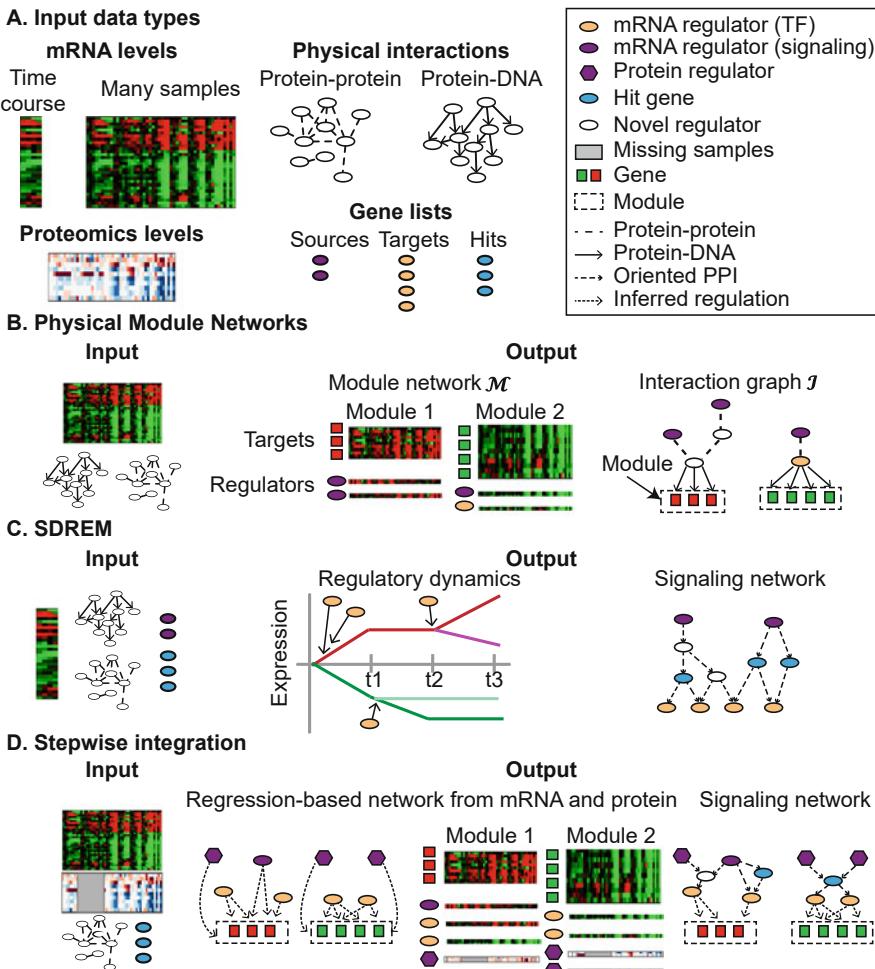


Fig. 3 Integrating signaling networks with transcriptional networks. **(a)** Shown are different input data types different methods incorporate: mRNA levels (time course or unrelated samples), proteomics levels, physical interaction data from public databases (protein–protein and protein–DNA), and gene lists from auxiliary data sources. **(b)** Physical Module Networks (PMN [45]) takes as input mRNA levels and physical interaction data. A PMN consists of a module network \mathcal{M} , which defines gene modules and their regulators, and an interaction graph \mathcal{I} , which provides physical interactions connecting regulators to modules. The two components \mathcal{M} and \mathcal{I} are learned simultaneously. **(c)** SDREM [46, 47] takes as input time course mRNA levels, a protein–DNA network, a protein–protein interaction network, and a gene list of candidate upstream regulators. SDREM is learned by iteratively looping between learning regulatory dynamics (DREM), including TF regulators associated with expression changes at a time point, and identifying an upstream signaling network that links upstream regulators to the TFs. Part of this panel is modeled after images produced by DREM software [48]. **(d)** The stepwise approach [49] takes as input mRNA levels and proteomics levels measured at matched time points, a protein–protein interaction network and an optional list of genes. The proteomic and transcriptomic data need not have complete overlap and in fact the proteomic data time points might be a subsample of the transcriptomic dataset. The approach begins by inferring a regulatory module network from the mRNA data, which defines gene modules as well as a regulatory network linking regulators to genes. Next, protein regulators are inferred for each module. Finally, a minimal upstream signaling network is identified to link protein regulators, mRNA regulators, and auxiliary gene hits to each module

$$P(d_e = 1 | I_e = 0) = p_e$$

I_e indicates whether edge e is in \mathcal{I} , and d_e indicates whether or not e was observed in the input interaction data. The edge scores W_e depend on the p -values p_e for edges in the interaction data, a hyper-parameter ω , and a predefined prior probability $P(I_e = 1)$, for adding any edge. This prior is set to a small value, e.g., $P(I_e = 1) = 0.001$, so that adding an edge to \mathcal{I} always incurs a cost.

To learn a PMN, both \mathcal{M} and \mathcal{I} are inferred simultaneously in a greedy-hill climbing approach. At each step, the algorithm chooses amongst three moves: (1) adding a regulator to a module in \mathcal{M} , (2) removing a regulator from a module in \mathcal{M} , or (3) reassigning genes to a different module in \mathcal{M} . Each move also entails updating \mathcal{I} such that the PMN remains consistent. In the update phase, edges are added or removed from \mathcal{I} and the score is accordingly computed. If a proposed regulator does not have a direct edge to the target gene module, the algorithm will aim to a TF by first using the heaviest (shortest) path from the regulator to the TF and add TF-DNA edges from the TF to the target gene modules. Each of the corresponding edge modifications will be incorporated into the computed score.

PMNs were shown to outperform module networks (MNs) in reconstructing regulatory interactions from synthetic data, while assigning the same likelihood to held-aside expression samples. In application to the yeast cell cycle, the approach was able to reconstruct known pathways and offer biologically plausible additional hypotheses. PMNs were also applied to mammalian gene expression data to include interactions between influenza virus and host proteins that revealed novel biological insights into the pathways that are used to respond to viral infections. Although PMNs capture a more realistic picture of gene regulation compared to MNs, they are less scalable than MNs and require additional data pre-processing to reduce the number of genes and modules that can be handled.

5.2 Signaling and Dynamic Regulatory Events Miner (SDREM)

SDREM was designed to combine static protein–protein interaction networks with dynamic gene expression patterns to integrate the signaling and transcriptional components of the gene regulation machinery [46, 47]. SDREM is specifically geared towards modeling short time-series data which are commonly used to measure global transcriptional dynamics of many biological processes such as stress response or response to infections. To link signaling and transcriptional networks SDREM assumes that there are a small number of upstream proteins, e.g., receptors, sensory or signaling proteins, that via a subnetwork of physical interactions regulate transcription factors and the downstream gene expression patterns (Fig. 3c). The model output by SDREM has two components: a downstream dynamic transcriptional regulatory

network that identifies transcription factors for target gene sets at specific time points, and an upstream signaling network that specifies directed, physical paths by which a small number of upstream proteins link to the transcription factors. The transcriptional regulatory network component is learned via the DREM (Dynamic Regulatory Events Miner [50]) approach. The signaling interaction network is inferred by orienting edges from upstream regulators to downstream gene expression levels [51]. By orienting the signaling network, one can predict upstream master signaling regulators and predict the effects of perturbation experiments.

The DREM component is based on an Input-Output Hidden Markov Model [52]. An instance of DREM consists of a set of hidden states, H , each of which is associated with a time point t in the input time course. Each hidden state h emits gene expression values according to a univariate Gaussian distribution. The possible transitions between the states are defined by a set of directed edges, E , which links a state at one time point to another state at a future time. In the simplest version of DREM, E is constrained to be a binary tree rooted at the beginning of the time course. Transitions from one state to two or more subsequent states are parameterized by regularized logistic regression functions that use the prior transcription factor network as predictor features. The model structure and parameters are estimated by an iterative algorithm that scores each DREM instance based on held-aside test genes. The final DREM model is used to infer the most likely state sequence for each gene, and to assign transcription factors to each outgoing edge of a transition branch point on the basis of overrepresentation of targets compared to the other branch.

The input to SDREM's signal orientation step is a weighted graph where the edges represent protein–protein and protein–DNA interactions and edge weights are proportional to confidence on the edge. The signaling network orientation step identifies a directed physical interaction subnetwork that links sources to the transcription factors that were identified by DREM. The orientation step begins by enumerating depth-bounded (e.g., five), acyclic paths through the weighted protein–protein interaction network from all sources to all targets. We refer to the set of paths as P . Each path $p \in P$ is composed of a set of edges, each denoted by e , and is assigned an indicator variable $I_s(p)$ and a weight, w_p . $I_s(p)$ is used to check the orientation of the path p . A path is considered correctly oriented if all arrows are pointing to the target TF. The path weight $w(p)$ is obtained from the weight of each edge, $w(e)$, the weight of the target node $w(t)$, and, if needed, node weights $w(v)$. $w(t)$ is associated with the activity of the TF t output by DREM. $w(v)$ by default is set to 1, although SDREM has ability to incorporate additional prior knowledge for nodes.

The path weight is computed as follows:

$$w(p) = w(t) \prod_{v \in p} w(v) \prod_{e \in p} w(e).$$

SDREM aims to maximize the score of paths that are correctly oriented from source to target:

$$\max \sum_{p \in P} I_s(p) w(p)$$

The orientation problem is solved using a search algorithm with restarts described in [51].

The SDREM learning algorithm iterates between these two components. It starts by applying DREM on the time-series expression data, where TFs are assigned to diverging expression paths. Those TFs are then passed to the orientation procedure, which identifies an oriented, weighted subnetwork linking the source genes to the TFs. Each TF is then assigned a score based on its connectivity in the subnetwork, which is then used by a subsequent iteration of DREM as a prior probability for including the TF. The two components are alternated for a set number of iterations or until the nodes included in the signaling network converge.

SDREM was first applied to study signaling and transcriptional dynamics of the yeast stress response [46]. Here, a small set of cell surface receptor proteins were used as upstream sources in the signaling network. Several novel intermediate regulators were validated by genome-wide expression profiling in knockout yeast strains. SDREM was also successfully scaled to analyze human cellular response to influenza viruses [47]. The upstream signaling network was used to integrate human proteins observed to interact with viral components. The resulting networks successfully recovered gene sets from independent RNAi screens for influenza host factors. These results demonstrate SDREM's potential for guiding experiments to gain insight into signaling networks in multiple systems, including yeast and human.

5.3 Stepwise Integration of Multiple Data Types

While the above two approaches used static interaction networks with gene expression profiles, several recent studies have generated proteomic datasets in addition to transcriptomic measurements. To integrate these different types of omic measurements, a stepwise strategy can be used. In a stepwise strategy, multiple data types are integrated sequentially into a signaling and regulatory network. This strategy is motivated by situations in which multiple omics data types are available, but there is a mismatch of samples between

data types, or the data types exhibit properties that make them unsuitable for a one-step integration by the same algorithm. Furthermore, such an algorithm could be computationally expensive.

Chasman et al. [49] developed a stepwise approach to integrate transcriptomic and proteomic datasets measured across multiple experimental conditions to infer signaling and regulatory networks. They applied this approach to examine regulatory networks governing host response to influenza infection, although the approach is broadly applicable to scenarios with both transcriptomic and proteomic measurements. The approach has three algorithmic components:

Step 1: Transcriptional Network Inference MERLIN [29] or another module network-based algorithm is applied to the transcriptomic data to infer a regulatory network wherein signaling proteins and transcription factors are linked to target genes and modules. An advantage of MERLIN over module-based methods is that it employs a soft module constraint, where different genes within a module are encouraged, but not required, to have the same regulators. Network inference is carried out in a stability selection framework to identify consensus edges and modules.

Step 2: Integration of Proteomics Data to Each Module To integrate the proteomics data, this approach uses a structured sparsity regression, Multi-task Group LASSO (MTG-LASSO) [53, 54], to predict regulators for modules. MTG-LASSO can be used to select a small number of regulators at the module level. Given an $m \times p$ matrix X measuring protein levels for p proteins in m samples and an expression matrix Υ for n genes in m samples, the MTG-LASSO objective is to infer a $p \times n$ regression weight matrix, W for p proteins to n genes, that minimizes the following:

$$\min_W \frac{1}{2} \|XW - \Upsilon\|_2^2 + \lambda \|W\|_{l_1/l_2} \quad (1)$$

The first term measures the model fit to the measured expression Υ and the prediction using the protein data X and regression weights W . The second term is the Group LASSO penalty on the weight matrix W . λ is a tunable parameter that defines the trade-off between the terms. The Group LASSO is a special class of structured sparsity regression method, which enables one to select a group of regression weights. In our case, each group is defined by the regression weights of one regulator for all genes in the module. The Group LASSO norm for W is defined as

$$\|W\|_{l_1/l_2} = \sum_{j=1}^p \|W_j\|_2.$$

W_j is the j th row in W and corresponds to the regression weights for the j th protein regulator for all n genes. Using the l_1 norm enables many groups to be driven to 0.

Step 3: Defining Signaling Networks Upstream of Modules The previous two steps identify statistical interactions between regulators and target genes, but these interactions could be the result of one or more physical interactions. In the third step, the regulators (both mRNA and protein) are connected to each module using physical protein–protein interactions to capture the upstream signaling network between signaling proteins and transcription factors. This step can be used to include additional important genes, e.g., genes known to be involved in the influenza host response.

Similar to the analogous component of SDREM, this approach begins by enumerating depth-limited paths between the upstream signaling proteins and downstream TFs. However, here the task was approached using mixed integer linear programming (ILP) [55–58]. A basic ILP approach to identifying signaling networks entails representing the inclusion/exclusion of network elements (nodes, edges, and paths) as binary indicator variables. The desiderata for the best connected, oriented subnetwork are defined as linear inequalities (constraints) and an objective function over those binary variables. The specific constraints and objective function can be varied by application [55–58]. The objective function used by Chasman et al. [49] aims to connect as many as possible source-target pairs while minimizing the number of intermediate nodes.

To illustrate how ILP approaches can be used to identify subnetworks, we consider the simplified version of the ILP that was defined in Chasman et al. [49]. The constraints defined for this ILP aim to capture a connected graph. For example, a constraint for a node in the subnetwork would be *an included node requires an included edge*. This can be specified as:

$$\forall n \in N, y_n \leq \sum_{e \in E_n} x_e$$

where N is the set of nodes, y_n is an indicator variable for node $n \in N$ that specifies if the node is included in the subnetwork or not. E is the set of edges, $E_n \subset E$ is the set of edges that has n as its end point, and x_e indicates the presence or absence of edge e in the subnetwork. Similarly, a constraint for the edge inclusion can be defined as:

$$\forall e \in E, n \in N_e \quad x_e \leq y_n$$

where N_e are the two nodes at the end points of e . Similarly, additional constraints are defined at the path level as follows:

An included edge requires an included path:

$$\forall e \in E \quad x_e \leq \sum_{p \in P_e} \sigma_p,$$

where P_e are all paths that touch the edge e , and σ_p is the indicator variable for p 's inclusion in the subnetwork.

An included path requires included edges:

$$\forall p \in P, e \in E_p \quad \sigma_p \leq x_e,$$

where E_p is the set of edges in path p , P is the set of all paths.

An included path requires properly oriented edges:

$$\forall p \in P, e \in E_p \quad \sigma_p \leq I(d_e = \text{dir}(p, e)),$$

where $\text{dir}(p, e)$ is a function that returns the value for d_e that is required to orient path p .

A connected source-target pair, $(s, t) \in S$, requires an included path:

$$\forall (s, t \in S) \quad c_{s,t} \leq \sum_{p \in P(s,t)} \sigma_p,$$

where S is the set of input source-target pairs and $c_{s,t}$ is an indicator variable to denote that a source and target are connected by an oriented path.

The optimal subnetwork is identified by optimizing two objective functions. The first one finds the maximum number of valid connected source-target pairs, max_pairs , subject to the set of constraints C and is defined as:

$$\text{max_pairs} = \max \sum_{(s,t) \in S} c_{s,t}, \text{ subject to } C$$

The second objective tries to minimize the number of intermediate nodes, while making sure the number of connected pairs remains the same by defining a new constraint C' , $\text{max_pairs} = \sum_{(s,t) \in S} c_{s,t}$.

The second objective is defined as:

$$(\hat{y}_1, \dots, \hat{y}_{|N|}) = \arg \min_{y_1, \dots, y_{|N|}} \sum_{n \in N} y_n, \text{ subject to } C \cup C'$$

Solving this objective returns an assignment to all the y_n determining the minimal connected paths. Each optimal solution to the ILP identifies a subnetwork. To assess confidence in each node, edge, and path, we obtain an ensemble of subnetworks by finding multiple solutions to the ILP [59].

Application of this approach to the influenza host response transcriptomic and proteomic dataset identified several mRNA and protein regulators, which were validated by siRNA screens to suppress viral replication in cell culture. Several predicted regulators had a significant impact on viral replication. The integrated host response network revealed gene modules with pathogenicity-specific expression profiles.

In summary, all three approaches identify signaling subnetworks from physical interactions. PMNs and the stepwise approach identify the relevant physical interactions underlying the statistical dependencies inferred from many samples of molecular measurements [45, 49]. SDREM [47] and the stepwise approach [49] integrate other relevant gene sets upstream of the transcriptional regulatory network. For all approaches, the signaling network can be used to identify additional important regulators and connections, including those that cannot be identified from expression data alone. The applicability of each method also depends on the data that are available in the system of interest. All three require physical interaction data, which are available for many organisms from databases such as BioGRID [60] and STRING [61]. PMNs may have the most basic requirements: many samples of transcriptomic data from the biological system of interest. SDREM can take advantage of time-series expression data. The stepwise approach integrates transcriptomic and proteomic data from the largely same conditions, but where different properties of the datasets make it difficult to integrate the data within the same algorithm.

6 Network Inference in Practice

In this last section, we provide step-by-step instructions for doing network inference in practice. We focus on the MERLIN-P algorithm, however, many of our steps are applicable to other algorithms as well. Broadly speaking there are six steps for inferring regulatory networks.

Step 1. Collect Expression and Other Large-Scale Omic Profiling Data

The first step is to collect as many as possible expression samples. Data can be obtained from one specific project and/or by searching for publicly available data aggregated in large databases. The Gene Expression Omnibus (GEO) [62] includes most high-

throughput data types, while the Sequence Read Archive (SRA) [63] stores sequencing samples. Some meta-analytic projects have been launched to uniformly process many datasets together. These resources include human and mouse microarrays performed on specific platforms [64], as well as human [65, 66] and mouse [66] RNA-seq samples. GEO can be queried using an advanced search query to obtain a list of GSE (experiment) datasets. Because the sample metadata fields are not standardized, collecting a good dataset for a specific application will require manual searching and curation. Towards ameliorating this common frustration, the MetaSRA project [67] annotates RNA-seq samples from human samples (e.g., tissues, cell lines, primary cell types) from sample metadata by incorporating biomedical ontologies and provides a web search tool to access these samples. Once the datasets have been identified, a number of pre-processing steps are needed to create the input data matrix for network inference.

A standard pre-processing pipeline would include reprocessing the samples (e.g., for RNA-seq experiments align reads to the genome), normalizing within each dataset, and performing batch correction [68, 69]. Batch effects are sources of variation between samples that are attributable to the physical context in which the experiment was conducted, rather than the intentional experimental perturbation. Uniform data processing, *e.g.*, by running the same alignment and quantification algorithms for RNA-seq, will not mitigate batch effects. A simple but effective diagnostic is to inspect the samples in PCA space or by using another dimensionality reduction-based visualization tool. If batch effects are present, the samples from the same laboratory or preparation will tend to cluster together, often more strongly than samples from different batches that were subject to similar perturbations. While optimal batch correction is an active area of research [68, 69], there are a few popular methods that have been widely applied. The R package SVA [70] implements two popular methods for batch correction: ComBat [71] and surrogate variable analysis (SVA) [72]. The simplest approach, which we have adopted, is to mean-center the values for each gene within each batch (after log transform for one-channel arrays and RNA-seq counts). At the end of these steps, the data matrix is ready for network inference.

Step 2: Define Candidate Regulators

Most network inference algorithms accept a list of input candidate regulators to simplify the inference problem and also to make the inferred networks easier to interpret. The candidate regulators are the only nodes in the network allowed to have outgoing arrows. These are typically transcription factors and signaling proteins, lists of which can be obtained for many organisms by searching Uniprot

[73] or Gene Ontology [74]. Other publications and databases provide organism-specific lists of transcription factors [20, 75, 76].

Step 3: Collect Prior Knowledge of Regulatory Edges

Prior knowledge for regulatory edges can be obtained from different pre-existing experimental datasets or databases. The type of prior edge knowledge that is most straightforward and easy to obtain is from sequence-specific motifs that are catalogued for many species in a number of databases such as JASPAR [77] and CIS-BP [78]. A motif network is obtained by scanning the gene promoters with known sequence specificities for transcription factors, e.g., by applying FIMO [79]. If DNase1-seq or ATAC-seq data for the condition of interest are available, a motif network can be filtered using peaks called by programs such as MACS2 [80]. Condition-specific motif instances can also be found using footprinting algorithms [81]. For a few transcription factors and contexts, ChIP-chip or ChIP-seq datasets can also be used as prior knowledge. Finally, knockout expression profiling assays can also be used as prior knowledge of edges between perturbed regulators and differentially expressed genes.

Step 4: Learning Consensus Networks and Modules

Due to the difficult nature of the network inference problem, both due to small sample size and high computational complexity, network inference algorithms can get stuck in local optima. Hence it is important to assess confidence in the inferred networks and modules. This can be done using “bootstrap” or “stability selection” [28], where we create different subsamples of the original dataset and infer a network on each subsample. The difference in bootstrap and stability selection is whether we sample with replacement as many samples as in the original dataset (bootstrap), or we sample a smaller fraction of the original dataset (stability selection). Once subsamples have been generated, network inference is performed on each subsample to obtain an ensemble of networks, followed by the creation of a consensus network and, if applicable (e.g., in MERLIN-P), consensus modules. In our applications of MERLIN-P, we have performed stability selection drawing 100 subsamples without replacement, where each subsample has 50% of the total number of samples. However, when there were fewer than 50 total samples available, a higher proportion of samples can be used.

A consensus MERLIN-P network is defined by first computing the confidence in each regulator-target edge as the fraction of networks that contain the edge and then selecting edges with confidence greater than a minimum confidence threshold. To determine a threshold, we recommend estimating the false discovery rate (FDR) based on comparison to a background ensemble of networks. The background ensemble is estimated by applying

the same procedure to a randomized expression dataset, where the values for each gene are permuted in order to break the true associations between genes. If a prior network is available, the same prior network is used to infer the null networks as the real networks. For a threshold t , the FDR is computed by comparing the relative numbers of edges identified in the background and real network. For the background (real) network, let N (R) be the total number of edges with any confidence ≥ 0 , and let n (r) be the number of edges with confidence $\geq t$. FDR is computed as $= \frac{n/N}{r/R}$. We define the consensus network by choosing a confidence threshold t above which $\text{FDR} \leq \alpha$.

The consensus MERLIN-P modules are defined based on the co-clustering patterns between genes. We define a weighted co-clustering graph with an edge where the weight of the edge is the fraction of MERLIN-P runs in which the two genes are assigned to the same module. The consensus modules are defined by applying hierarchical clustering to this graph and cutting the clustering dendrogram at a maximum tolerated difference, estimated from the co-clustering frequencies in the background ensemble.

Step 5: Integrate Upstream Regulators and Network (Optional)

The inputs for this step are physical interaction networks (e.g., protein–protein and protein–DNA interactions) and gene lists derived from other experiments probing the same system as the inferred network (Fig. 3). Gene lists can be split into two or three categories: sources, targets, and additional hits. The sources and targets are used to orient the signaling network. For example in MERLIN-P, the sources are the inferred signaling regulators and the targets are transcription factors. The additional hit genes can be included as path intermediates. The hit genes can be drawn from application-specific functional screening assays. Scalable, freely available tools for defining minimal or high-confidence signaling networks include PathLinker [82] and OmicsIntegrator [83]. To obtain uniquely oriented edges, options include Maximum Edge Orientation [51] or ANAT [84]. For more complex requirements, including orientation and edge sign, a custom ILP can be developed and optimized with the aid of open source libraries or commercial solvers, e.g., CPLEX (<https://www.ibm.com/products/ilog-cplex-optimization-studio>) and Gurobi [85].

Step 6: Evaluation and Interpretation of Modules and Networks

The final step in network inference is the validation and evaluation of the inferred networks and modules. Several different metrics can be used to assess the goodness of the results.

- *Area under the precision-recall curve (AUPR):* The precision-recall curve is commonly used to compare the ranking of edges

(for example, by defining confidence for each edge using a sub-sampling method like stability selection), to a “gold-standard network” derived from the literature or experimental methods, e.g., genetic perturbation or ChIP-chip/seq experiments. For any given confidence threshold π , let E_π denote all edges with confidence higher than π . E_π is used to estimate precision (p^π) and recall (r^π) for the given threshold. Plotting p^π and r^π for all values of π provides a measure of how well the inference method was able to recover the gold standard. This is quantified using the area under precision-recall curve (AUPR) [86]. The maximum value of AUPR is 1 which corresponds to a precision-recall curve with perfect precision ($p = 1$) for all values of recall.

- *F-score:* If the network inference method returns a network, rather than a ranking of edges, the F-score metric can be used. The F-score corresponds to a point on the precision-recall curve and corresponds to the harmonic mean of precision and recall: $f = \frac{2 \times p \times r}{p + r}$.
- *Predictable targets and regulators:* While AUPR and F-score assess the entire network as a whole, it is also possible to assess the quality of the network using more local measures to identify specific nodes that are predicted well over others [43]. Predictable TFs are defined as a set of TFs whose inferred targets have a significant overlap with the true targets (defined in a gold standard that could be derived from ChIP-chip/seq or knockout experiments). Predictable targets are defined as those genes for which their expression can be predicted much better using the inferred network than a random network. A good method is one that has a high value for these metrics.
- *Enrichment of curated gene sets:* Modules can be evaluated and interpreted by testing a module for overrepresentation of genes annotated in curated processes and pathways using databases such as Gene Ontology [74], KEGG [87], and MSigDB [88]. The enrichment is carried out using a hypergeometric test which asks whether the overlap of genes annotated with a term/pathway of interest and a module is significantly greater than expected by chance. Due to the large number of terms, the P-values obtained from these tests are corrected using some form of multiple hypothesis correction [89]. Modules can additionally be interpreted by testing them for enrichment of binding sites of transcription factors derived using sequence or ChIP-chip/seq data. The enrichment test uses a similar hypergeometric test as for curated gene sets.
- *Indirect support for inferred edges:* Due to the lack of large-scale gold standards, several approaches [49, 90, 91] have used protein–protein interaction networks and genetic interaction networks to support the inferred statistical dependencies in

regulatory networks. The rationale of these approaches is that genes that are regulated by the same regulator are likely to share protein–protein or genetic interactions. Another strategy is to assess the inferred network using topological properties such as degree distributions as well as the overrepresentation of specific types of network motifs that have been shown to be associated with complex networks [90]. Typically one would compute these metrics on the inferred network and compare it to a background network. While these are not perfect metrics to assess the quality of the inferred network they have been shown to be enriched in smaller ground truth networks and provide some support to the quality of the inferred networks [90].

7 Conclusion

In this chapter, we have described current computational methods, based on probabilistic graphical models, for inferring genome-scale regulatory networks by integrating different types of data. The methods aim to overcome some of the drawbacks of purely expression-based network inference methods. The first class of methods incorporate non-expression prior knowledge to infer a more accurate network. The second class of methods incorporate additional regulators using protein–protein interaction networks and/or other omic datasets. When multiple types of omic datasets are available a stepwise approach can be used especially when there are mismatches in the samples at the different omic levels. Ideally, one would infer a network combining different omics levels in a more integrated manner. An exciting area of work is the development of mixed graphical models that integrate data types with different distributions [92, 93], which may pave the way for more tightly integrated inference of multi-omic data. Currently network inference remains a difficult problem due to the lack of large-scale benchmarks. Another important area of future work is to use iterative experimental design to systematically generate prioritized predictions followed by experimental validations to improve the learned networks. Experimentally validated interactions can serve as gold standards for future method development.

Acknowledgements

This work is supported in part by US Environmental Protection Agency grant 83573701, NIH NIGMS grant 1R01GM117339, and NSF CAREER award to Sushmita Roy.

References

1. Markowetz F, Spang R (2007) Inferring cellular networks—a review. *BMC Bioinf* 8(Suppl 6):S5
2. Kim HD, Shay T, O’Shea EK, Regev A (2009) Transcriptional regulatory circuits: predicting numbers from alphabets. *Science* 325(5939):429–432
3. Thompson D, Regev A, Roy S (2015) Comparative analysis of gene regulatory networks: from network reconstruction to evolution. *Annu Rev Cell Dev Biol* 31:399–428
4. Gasch AP, Spellman PT, Kao CM, Carmel-Harel O, Eisen MB, Storz G, Botstein D, Brown PO (2000) Genomic expression programs in the response of yeast cells to environmental changes. *Mol Biol Cell* 11(12):4241–4257
5. Ideker T, Krogan NJ (2012) Differential network biology. *Mol Syst Biol* 8:565
6. Lee TI, Young RA (2013) Transcriptional regulation and its misregulation in disease. *Cell* 152(6):1237–1251
7. Voss TC, Hager GL (2014) Dynamic regulation of transcriptional states by chromatin and transcription factors. *Nat Rev Genet* 15(2):69–81
8. de Jong H (2002) Modeling and simulation of genetic regulatory systems: a literature review. *J Comput Biol J Comput Mol Cell Biol* 9:67–103
9. Huang S, Kauffman SA (2009) Complex gene regulatory networks – from structure to biological observables: cell fate determination. In: Encyclopedia of complexity and systems science. Springer New York, pp 1180–1213
10. Carpenter AE, Sabatini DM (2004) Systematic genome-wide screens of gene function. *Nat Rev Genet* 5(1):11–22
11. Giaever G, Nislow C (2014) The yeast deletion collection: a decade of functional genomics. *Genetics* 197(2):451–465
12. Ren B, Robert F, Wyrick J, Aparicio O, Jennings E, Simon I, Zeitlinger J, Schreiber J, Hannett N, Kanin E, Volkert T, Wilson C, Bell S, Young R (2000) Genome-wide location and function of DNA binding proteins. *Science* 290(5500):2306–2309
13. Furey TS (2012) ChIP-seq and beyond: new and improved methodologies to detect and characterize protein-DNA interactions. *Nat Rev Genet* 13(12):840–852
14. Song L, Crawford GE (2010) DNase-seq: a high-resolution technique for mapping active gene regulatory elements across the genome from mammalian cells. *Cold Spring Harb Protoc* 2010(2):pdb.prot5384–pdb.prot5384
15. Buenrostro JD, Giresi PG, Zaba LC, Chang HY, Greenleaf WJ (2013) Transposition of native chromatin for fast and sensitive epigenomic profiling of open chromatin, DNA-binding proteins and nucleosome position. *Nat Methods* 10:1213–1218
16. MacGilvray ME, Shishkova E, Chasman D, Place M, Gitter A, Coon JJ, Gasch AP (2018) Network inference reveals novel connections in pathways regulating growth and defense in the yeast salt response. *PLoS Comput Biol* 13(5):1–28
17. Figeys D (2008) Mapping the human protein interactome. *Cell Res* 18:716–724
18. Braun P (2012) Interactome mapping for analysis of complex phenotypes: insights from benchmarking binary interaction assays. *Proteomics* 12:1499–1518
19. Friedman N, Nachman I, Peér D (1999) Learning bayesian network structure from massive datasets: The “sparse candidate” algorithm. In: Proceedings of the fifteenth conference on uncertainty in artificial intelligence, UAI’99. Morgan Kaufmann Publishers Inc., San Francisco, CA, pp 206–215
20. Segal E, Shapira M, Regev A, Pe’er D, Botstein D, Koller D, Friedman N (2003) Module networks: identifying regulatory modules and their condition-specific regulators from gene expression data. *Nat Genet* 34(2):166–176
21. Margolin AA, Nemenman I, Basso K, Wiggins C, Stolovitzky G, Favera RD, Califano A (2006) ARACNE: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. *BMC Bioinf* 7(Suppl 1):S7+
22. Faith JJ, Hayete B, Thaden JT, Mogno I, Wierzbowski J, Cottarel G, Kasif S, Collins JJ, Gardner TS (2007) Large-scale mapping and validation of *Escherichia coli* transcriptional regulation from a compendium of expression profiles. *PLoS Biol* 5(1):e8+
23. Langfelder P, Horvath S (2008) WGCNA: an R package for weighted correlation network analysis. *BMC Bioinf* 9:559
24. Joshi A, De Smet R, Marchal K, Van de Peer Y, Michoel T (2009) Module networks revisited: computational assessment and prioritization of model predictions. *Bioinformatics* 25(4):490–496

25. Huynh-Thu VA, Irrthum A, Wehenkel L, Geurts P (2010) Inferring regulatory networks from expression data using Tree-Based methods. *PLoS One* 5(9):e12776+
26. Haury ACC, Mordelet F, Vera-Licona P, Vert JPP (2012) TIGRESS: trustful inference of gene REgulation using stability selection. *BMC Syst Biol* 6(1):145+
27. Tibshirani R (1996) Regression shrinkage and selection via the lasso. *J R Stat Soc Ser B (Methodol)* 58(1):267–288
28. Meinshausen N, Bühlmann P (2010) Stability selection. *J R Stat Soc Ser B (Stat Methodol)* 72(4):417–473
29. Roy S, Lagree S, Hou Z, Thomson JA, Stewart R, Gasch AP (2013) Integrated module and Gene-Specific regulatory inference implicates upstream signaling networks. *PLoS Comput Biol* 9(10):e1003252+
30. Marbach D, Costello JC, Küffner R, Vega NM, Prill RJ, Camacho DM, Allison KR, Aderhold A, Allison KR, Bonneau R, et al (2012) Wisdom of crowds for robust gene network inference. *Nat Methods* 9(8):796–804
31. Friedman N (2004) Inferring cellular networks using probabilistic graphical models. *Science* 303(5659):799–805
32. De Smet R, Marchal K (2010) Advantages and limitations of current network inference methods. *Nat Rev Microbiol* 8(10):717–729
33. Friedman N, Linial M, Nachman I, Pe'er D (2000) Using Bayesian networks to analyze expression data. *J Comput Biol* 7(3–4):601–620
34. Pe'er D, Regev A, Tanay A (2002) Minreg: inferring an active regulator set. *Bioinformatics (Oxford, England)* 18(Suppl 1):S258–S267
35. Heckerman D, Chickering DM, Meek C, Rounthwaite R, Kadie C (2001) Dependency networks for inference, collaborative filtering, and data visualization. *J Mach Learn Res* 1:49–75
36. Werhli AV, Husmeier D (2007) Reconstructing gene regulatory networks with Bayesian networks by combining expression data with multiple sources of prior knowledge. *Stat Appl Genet Mol Biol* 6(1): Article15
37. Hill SM, Lu Y, Molina J, Heiser LM, Spellman PT, Speed TP, Gray JW, Mills GB, Mukherjee S (2012) Bayesian inference of signaling network topology in a cancer cell line. *Bioinformatics* 28(21):2804–2810
38. Bonneau R, Reiss D, Shannon P, Facciotti M, Hood L, Baliga N, Thorsson V (2006) The inferelator: an algorithm for learning parsimonious regulatory networks from systems-biology data sets de novo. *Genome Biol* 7(5):R36+
39. Greenfield A, Hafemeister C, Bonneau R (2013) Robust data-driven incorporation of prior knowledge into the inference of dynamic regulatory networks. *Bioinformatics* 29(8):1060–1067
40. Koller D, Friedman N (2009) Probabilistic graphical models: principles and techniques. MIT Press, Cambridge, MA
41. Grzegorczyk M, Husmeier D, Werhli AV (2008) Reverse engineering gene regulatory networks with various machine learning methods. Wiley-VCH Verlag GmbH & Co. KGaA, Weinheim, pp 101–142
42. Lee SI, Dudley AM, Drubin D, Silver PA, Krogan NJ, Pe'er D, Koller D (2009) Learning a prior on regulatory potential from eQTL data. *PLoS Genet* 5(1):e1000358
43. Siahpirani AF, Roy S (2017) A prior-based integrative framework for functional transcriptional regulatory network inference. *Nucleic Acids Res* 45:e21
44. Zou H, Hastie T (2005) Regularization and variable selection via the elastic net. *J R Stat Soc Ser B Stat Methodol* 67(2):301–320
45. Novershtern N, Regev A, Friedman N (2011) Physical module networks: an integrative approach for reconstructing transcription regulation. *Bioinformatics* 27(13):i177–i185
46. Gitter A, Carmi M, Barkai N, Bar-Joseph Z (2013) Linking the signaling cascades and dynamic regulatory networks controlling stress responses. *Genome Res* 23(2):365–376
47. Gitter A, Bar-Joseph Z (2013) Identifying proteins controlling key disease signaling pathways. *Bioinformatics* 29(13):i227–i236
48. Schulz MH, Devanny WE, Gitter A, Zhong S, Ernst J, Bar-Joseph Z (2012) Drem 2.0: improved reconstruction of dynamic regulatory networks from time-series expression data. *BMC Syst Biol* 6:104
49. Chasman D, Walters KB, Lopes TJS, Eisfeld AJ, Kawaoka Y, Roy S (2016) Integrating transcriptomic and proteomic data using predictive regulatory network models of host response to pathogens. *PLoS Comput Biol* 12:e1005013
50. Ernst J, Vainas O, Harbison CT, Simon I, Bar-Joseph Z (2007) Reconstructing dynamic regulatory maps. *Mol Syst Biol* 3:74

51. Gitter A, Klein-Seetharaman J, Gupta A, Bar-Joseph Z (2011) Discovering pathways by orienting edges in protein interaction networks. *Nucleic acids Res* 39:e22
52. Bengio Y, Frasconi P (1996) Input-output HMMs for sequence processing. *IEEE Trans Neural Netw* 7:1231–1249
53. Yuan M, Lin Y (2006) Model selection and estimation in regression with grouped variables. *J R Stat Soc Ser B (Stat Methodol)* 68(1):49–67
54. Obozinski G, Taskar B, Jordan M (2006) Multi-task feature selection, Technical report 2. Statistics Department, UC Berkeley
55. Ourfali O, Shlomi T, Ideker T, Ruppin E, Sharan R (2007) SPINE: a framework for signaling-regulatory pathway inference from cause-effect experiments. *Bioinformatics* 23(13):i359–i366
56. Silverbush D, Elberfeld M, Sharan R (2011) Optimally orienting physical networks. *J Comput Biol J Comput Mol Cell Biol* 18:1437–1448
57. Chasman D, Gancarz B, Hao L, Ferris M, Ahlquist P, Craven M (2014a) Inferring host gene subnetworks involved in viral replication. *PLoS Comput Biol* 10(5):e1003626
58. Chasman D, Ho Y, Berry DB, Nemec CM, MacGilvray ME, Hose J, Merrill AE, Lee MV, Will JL, Coon JJ, Ansari AZ, Craven M, Gasch AP (2014b) Pathway connectivity and signaling coordination in the yeast stress-activated signaling network. *Mol Syst Biol* 10(11):759+
59. Danna E, Fenelon M, Gu Z, Wunderling R (2007) Generating multiple solutions for mixed integer programming problems. In: Integer programming and combinatorial optimization. Springer, Berlin/Heidelberg, pp 280–294
60. Stark C, Breitkreutz BJ, Reguly T, Boucher L, Breitkreutz A, Tyers M (2006) BioGRID: a general repository for interaction datasets. *Nucleic Acids Res* 34(Suppl 1):D535–D539
61. Franceschini A, Szklarczyk D, Frankild S, Kuhn M, Simonovic M, Roth A, Lin J, Minguez P, Bork P, von Mering C, Jensen LJ (2013) String v9.1: protein-protein interaction networks, with increased coverage and integration. *Nucleic Acids Res* 41(Database issue):D808–D815
62. Edgar R, Domrachev M, Lash AE (2002) Gene Expression Omnibus: NCBI gene expression and hybridization array data repository. *Nucleic Acids Res* 30(1):207–210
63. Leinonen R, Sugawara H, Shumway M, Collaboration INSD (2010) The sequence read archive. *Nucleic acids Res* 39(Suppl 1):D19–D21
64. Cahan P, Li H, Morris SA, Lummertz da Rocha E, Daley GQ, Collins JJ (2014) Cellnet: network biology applied to stem cell engineering. *Cell* 158(4):903–915
65. Collado-Torres L, Nellore A, Kammer K, Ellis SE, Taub MA, Hansen KD, Jaffe AE, Langmead B, Leek JT (2017) Reproducible RNA-seq analysis using recount2. *Nat Biotechnol* 35:319–321
66. Lachmann A, Torre D, Keenan AB, Jagodnik KM, Lee HJ, Silverstein MC, Wang L, Ma'ayan A (2017) Massive mining of publicly available RNA-seq data from human and mouse. *bioRxiv* preprint
67. Bernstein MN, Doan A, Dewey CN (2017) MetaSRA: normalized human sample-specific metadata for the sequence read archive. *Bioinformatics* (Oxford, England) 33:2914–2923
68. Leek JT, Scharpf RB, Bravo HC, Simcha D, Langmead B, Johnson WE, Geman D, Baggerly K, Irizarry RA (2010) Tackling the widespread and critical impact of batch effects in high-throughput data. *Nat Rev Genet* 11:733–739
69. Goh WWB, Wang W, Wong L (2017) Why batch effects matter in omics data, and how to avoid them. *Trends Biotechnol* 35:498–507
70. Leek JT, Johnson WE, Parker HS, Fertig EJ, Jaffe AE, Storey JD (2015) SVA: Surrogate Variable Analysis. R package version 3.18.0
71. Johnson WE, Li C, Rabinovic A (2007) Adjusting batch effects in microarray expression data using empirical bayes methods. *Biostatistics* (Oxford, England) 8:118–127
72. Leek JT, Storey JD (2007) Capturing heterogeneity in gene expression studies by surrogate variable analysis. *PLoS Genet* 3:1724–1735
73. Apweiler R, Bairoch A, Wu CH, Barker WC, Boeckmann B, Ferro S, Gasteiger E, Huang H, Lopez R, Magrane M, et al (2004) Uniprot: the universal protein knowledgebase. *Nucleic acids Res* 32(Suppl 1):D115–D119
74. Ashburner M, Ball CA, Blake JA, Botstein D, Butler H, Cherry JM, Davis AP, Dolinski K, Dwight SS, Eppig JT, et al (2000) Gene ontology: tool for the unification of biology. *Nat Genet* 25(1):25–29
75. Ravasi T, Suzuki H, Cannistraci CV, Katayama S, Bajic VB, Tan K, Akalin A, Schmeier S, Kanamori-Katayama M, Bertin N, et al (2010)

- An atlas of combinatorial transcriptional regulation in mouse and man. *Cell* 140(5):744–752
76. Jin J, Tian F, Yang DC, Meng YQ, Kong L, Luo J, Gao G (2017) Planttfdb 4.0: toward a central hub for transcription factors and regulatory interactions in plants. *Nucleic acids Res* 45(D1):D1040–D1045
77. Mathelier A, Fornes O, Arenillas DJ, Chen CY, Denay G, Lee J, Shi W, Shyr C, Tan G, Worsley-Hunt R, Zhang AW, Parcy F, Lenhard B, Sandelin A, Wasserman WW (2016) Jaspar 2016: a major expansion and update of the open-access database of transcription factor binding profiles. *Nucleic Acids Res* 44:D110–D115
78. Weirauch MT, Yang A, Albu M, Cote AG, Montenegro-Montero A, Drewe P, Najafabadi HS, Lambert SA, Mann I, Cook K, Zheng H, Goity A, van Bakel H, Lozano JC, Galli M, Lewsey MG, Huang E, Mukherjee T, Chen X, Reece-Hoyes JS, Govindarajan S, Shaulsky G, Walhout AJM, Bouget FY, Ratsch G, Larrondo LF, Ecker JR, Hughes TR (2014) Determination and inference of eukaryotic transcription factor sequence specificity. *Cell* 158(6):1431–1443
79. Grant CE, Bailey TL, Noble WS (2011) Fimo: scanning for occurrences of a given motif. *Bioinformatics* 27(7):1017–1018
80. Zhang Y, Liu T, Meyer CA, Eeckhoute J, Johnson DS, Bernstein BE, Nusbaum C, Myers RM, Brown M, Li W, Liu XS (2008) Model-based analysis of ChIP-Seq (MACS). *Genome Biol* 9:R137
81. Gusmao EG, Allhoff M, Zenke M, Costa IG (2016) Analysis of computational footprinting methods for DNase sequencing experiments. *Nat Methods* 13(4):303–309
82. Ritz A, Poirel CL, Tegge AN, Sharp N, Simmons K, Powell A, Kale SD, Murali TM (2016) Pathways on demand: automated reconstruction of human signaling networks. *npj Syst Biol Appl* 2:16002+
83. Tuncbag N, Gosline SJC, Kedaigle A, Soltis AR, Gitter A, Fraenkel E (2016) Network-based interpretation of diverse high-throughput datasets through the omics integrator software package. *PLOS Comput Biol* 12(4):e1004879+
84. Almozino Y, Atias N, Silverbush D, Sharan R (2017) Anat 2.0: reconstructing functional protein subnetworks. *BMC Bioinf* 18:495
85. Gurobi Optimization, Inc (2016) Gurobi optimizer reference manual
86. Davis J, Goadrich M (2006) The relationship between Precision-Recall and ROC curves. In: Proceedings of the 23rd international conference on machine learning (ICML 2006), ICML '06. ACM, New York, NY, pp 233–240
87. Kanehisa M, Furumichi M, Tanabe M, Sato Y, Morishima K (2017) KEGG: new perspectives on genomes, pathways, diseases and drugs. *Nucleic Acids Res* 45:D353–D361
88. Liberzon A, Subramanian A, Pinchback R, Thorvaldsdóttir H, Tamayo P, Mesirov JP (2011) Molecular signatures database (MSigDB) 3.0. *Bioinformatics* (Oxford, England) 27:1739–1740
89. Noble WS (2009) How does multiple testing correction work? *Nat Biotechnol* 27:1135–1137
90. Marbach D, Roy S, Ay F, Meyer PE, Candeias R, Kahveci T, Bristow CA, Kellis M (2012) Predictive regulatory models in drosophila melanogaster by integrative inference of transcriptional networks. *Genome Res* 22(7):1334–1349
91. Bonnet E, Calzone L, Michoel T (2015) Integrative multi-omics module network inference with Lemon-Tree. *PLoS Comput Biol* 11:e1003983
92. Chen S, Witten DM, Shojaie A (2014) Selection and estimation for mixed graphical models. *Biometrika* <https://doi.org/10.1093/biomet/asu051>
93. Žitnik M, Zupan B (2015) Gene network inference by fusing data from diverse distributions. *Bioinformatics* (Oxford, England) 31:i230–i239



Chapter 8

Unsupervised Gene Network Inference with Decision Trees and Random Forests

Vân Anh Huynh-Thu and Pierre Geurts

Abstract

In this chapter, we introduce the reader to a popular family of machine learning algorithms, called decision trees. We then review several approaches based on decision trees that have been developed for the inference of gene regulatory networks (GRNs). Decision trees have indeed several nice properties that make them well-suited for tackling this problem: they are able to detect multivariate interacting effects between variables, are non-parametric, have good scalability, and have very few parameters. In particular, we describe in detail the GENIE3 algorithm, a state-of-the-art method for GRN inference.

Key words Machine learning, Decision trees, Regression trees, Tree ensembles, Random forest

1 Introduction

This chapter focuses on a popular family of machine learning algorithms, called decision trees. The goal of tree-based algorithms is to learn a model, in the form of a decision tree or an ensemble of decision trees, that is able to predict the value of an output variable given the values of some input variables. Tree-based methods have been widely used to solve diverse problems in computational biology, such as DNA sequence annotation or biomarker discovery (*see [1–3]* for reviews). In particular, several approaches based on decision trees have been developed for the inference of gene regulatory networks (GRNs) from expression data. Decision trees have indeed several advantages that make them attractive for tackling this problem. First, they are potentially able to detect multivariate interacting effects between variables, which make them well-suited for modelling gene regulation, as the regulation of the expression of one gene is expected to be combinatorial, i.e., to involve several regulators. Tree-based methods have also the advantage

to be non-parametric. They thus do not make any assumption about the nature of the interactions between the variables (such as linearity or Gaussianity). As their computational complexity is typically at most linear in the number of features, they can deal with high-dimensionality, a characteristic usually encountered in gene expression datasets. They are also flexible as they can handle both continuous and discrete variables. With respect to other supervised learning methods such as support vector machines or artificial neural networks, tree-based methods have very few parameters, which make them easy to use, even for non-specialists. See also Chapter 9 for another usage of tree-based methods in GRN inference.

One of the most widely used tree-based methods for GRN inference is GENIE3 [4]. This method exploits variable importance scores derived from ensembles of regression trees to identify the regulators of each target gene. GENIE3 was the best performer of the DREAM4 *Multifactorial Network* challenge and the DREAM5 *Network Inference* challenge [5], and is currently one of the state-of-the-art approaches for GRN inference. This method has also been evaluated and compared to other methods in numerous independent studies (e.g., [6–13]), usually achieving competitive results, and has often been used (either alone or in combination with other inference methods) to reconstruct real networks in various organisms such as bacteria [14–16], plants [17–19], drosophila [20], mouse [21, 22], and human [23, 24].

The chapter is structured as follows. Subheading 2 introduces general notions of supervised learning, while Subheading 3 specifically focuses on regression tree-based approaches. Subheading 4 presents several tree-based methods for GRN inference. In particular, the GENIE3 algorithm is described in detail. Finally, Subheading 5 discusses potential improvements of GENIE3.

2 Supervised Learning

Machine learning is a branch of artificial intelligence whose goal is to extract knowledge from observed data. In particular, supervised learning is the machine learning task of inferring a model f that predicts the value of an output variable Υ , given the values of m inputs X_1, X_2, \dots, X_m . The model f is learned from N instances (also called samples or observations) of input–output pairs, drawn from the (usually unknown) joint distribution $p(X_1, X_2, \dots, X_m, \Upsilon)$ of the variables:

$$\text{LS} = \{(\mathbf{x}_k, y_k)\}_{k=1}^N. \quad (1)$$

The set of instances is called *learning sample*. Depending on whether the output is discrete or continuous, the learning problem is a *classification* or a *regression* problem, respectively. In this chapter, we will focus on regression problems.

Let L be a loss function that, given an instance (\mathbf{x}, y) , measures the difference between the value $f(\mathbf{x})$ predicted by the model f from the input \mathbf{x} and the observed value y of the target variable. For a regression problem, a widely used loss function is the squared error:

$$L(y, f(\mathbf{x})) = (y - f(\mathbf{x}))^2. \quad (2)$$

The goal of supervised learning is to find, from a learning sample LS, a model f that minimizes the *generalization error*, i.e., the expected value of the loss function, taken over different instances randomly drawn from the joint distribution $p(X_1, X_2, \dots, X_m, Y)$:

$$E_{\mathbf{x}, y} [L(y, f(\mathbf{x}))]. \quad (3)$$

Since the joint distribution $p(X_1, X_2, \dots, X_m, Y)$ is usually unknown, supervised learning algorithms typically work by minimizing the *training error*, which is the average prediction error of the model over the instances of the learning sample:

$$\frac{1}{N} \sum_{k=1}^N L(y_k, f(\mathbf{x}_k)). \quad (4)$$

As the training error is calculated on the same samples that were used to learn the predictive model, it typically underestimates the generalization error, as shown in Fig. 1. The training error typically decreases when the complexity of the model is increased, i.e., when the model is allowed to fit more closely the training data. If the complexity is too high, the model may also fit the noise contained in the data and thus will have a poor generalization performance. In this case, we say that the model *overfits* the training data. On the other hand, if the model has a too low complexity, it *underfits* the data and will also have a high generalization error. Hence there is an optimal model complexity that leads to the minimal generalization error.

For more details, the reader is invited to refer to general books about machine learning [25, 26].

3 Regression Trees

A popular approach to the regression problem is the regression tree [27]. Figure 2 shows the structure of a tree. In this example, there are two input variables X_1 and X_2 , which are both continuous.

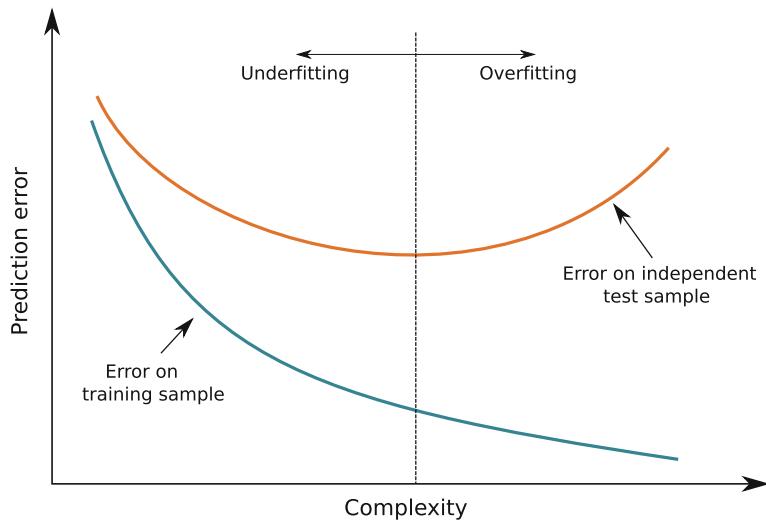


Fig. 1 Overfitting and underfitting. The blue (resp. orange) curve plots, for varying levels of complexity of the predictive model, the average value of the loss function over the instances of the learning sample (resp. of an independent test sample). Overfitting occurs when the model is too complex and underfitting occurs when the model is not complex enough

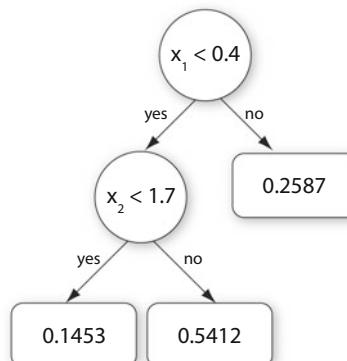


Fig. 2 Example of a regression tree. Each interior node of a tree is a test on one input variable and each terminal node contains a predicted value for the output variable

Each interior node of the tree contains a test of the type “ $X_i < c$,” where X_i is one of the input variables and c a threshold value, and each terminal node (or *leaf*) contains a predicted value for the output variable. Given a new sample, for which we have observed values of the input variables, a prediction for the output is obtained by propagating the sample down the tree, until it reaches a leaf. The predicted output value is then the value at that leaf.

3.1 Learning a Regression Tree

Using a learning sample LS, the goal of a tree-based method is to identify the tree that minimizes the training error in Eq. (4). A brute-force approach would consist in enumerating all the possible trees. This approach is however intractable and for this reason tree-based methods are rather based on greedy algorithms. A regression tree is typically constructed top-down, starting from a root node corresponding to the whole learning sample. The idea is then to recursively split the learning sample with binary tests on the values of the input variables, trying to reduce as much as possible the variance of the output variable in the resulting subsets of samples. At each interior node \mathcal{N} , the best test “ $X_i < c$ ” is chosen, i.e., the variable X_i and the threshold value c that maximize:

$$I(\mathcal{N}) = \#S.\text{Var}_Y(S) - \#S_t.\text{Var}_Y(S_t) - \#S_f.\text{Var}_Y(S_f), \quad (5)$$

where S denotes the set of samples of LS that reach node \mathcal{N} , S_t (resp. S_f) denotes its subset for which the test is true (resp. false), $\#$ denotes the cardinality of a set of samples, and $\text{Var}_Y(\cdot)$ is the variance of the output in a subsample. The samples of S are then split into two subsamples following the optimal test and the same procedure is applied on each of these subsamples. A node becomes a terminal node if the variance of the output variable, computed over the samples reaching that node, is equal to zero. Each terminal node contains a predicted value for the output, corresponding to the mean value of the output taken over the samples that reach that node.

However, a fully grown tree typically overfits the training data. Overfitting can be avoided by *pruning* the tree, i.e., by removing some of its subtrees. Two types of pruning exist: pre-pruning and post-pruning. In a pre-pruning procedure, a node becomes a terminal node instead of a test node if it meets a given criterion, such as:

- The number of samples reaching the node is below a threshold N_{\min} ;
- The variance of the output variable, over the samples reaching the node, is below a threshold Var_{\min} ;
- The optimal test is not statistically significant, according to some statistical test.

On the other side, the post-pruning procedure consists in fully developing a first tree \mathcal{T}_1 from the learning sample and then computing a sequence of trees $\{\mathcal{T}_2, \mathcal{T}_3, \dots\}$ such that \mathcal{T}_i is a pruned version of \mathcal{T}_{i-1} . The prediction error of each tree is then calculated on an independent set of samples and the tree that leads to the lowest prediction error is selected. The main drawback of the post-pruning procedure is that an independent set of samples is needed, while the main drawback of pre-pruning is that the

optimal value of the parameter related to the chosen stop-splitting criterion (N_{\min} , Var_{\min} , the significance level) is dependent on the considered problem.

Besides pruning, ensemble methods constitute another way of avoiding overfitting. These methods are described in the following section.

3.2 Ensemble Methods

Single regression trees are usually very much improved by ensemble methods, which average the predictions of several trees. The goal of ensemble methods is to use diversified models to reduce the overfitting of a learning algorithm. In the case of a tree, the overfitting comes mostly from the choices, made at each split node, of the input variable and the threshold value used for the test. Among the most widely used tree-based ensemble methods are methods that rely on randomization to generate diversity among the different models. These methods are Bagging [28], Random forest [29], and Extra-Trees [30].

3.2.1 Bagging

In the Bagging (for “Bootstrap AGGREGatING”) algorithm, each tree of the ensemble is built from a bootstrap replica, i.e., a set of samples obtained by N random samplings with replacement in the original learning sample. The choices of the variable and of the threshold at each test node are thus implicitly randomized via the bootstrap sampling.

3.2.2 Random Forest

This method adds an extra level of randomization compared to the Bagging. In a Random forest ensemble, each tree is built from a bootstrap sample of the original learning sample and at each test node, K variables are selected at random (without replacement) among all the input variables before determining the best split. When K is set to the total number of input variables, the Random forest algorithm is equivalent to Bagging.

3.2.3 Extra-Trees

In the Extra-Trees (for “EXTremely Randomized Trees”) method, each tree is built from the original learning sample but at each test node, the best split is determined among K random splits, each determined by randomly selecting one input variable (without replacement) and a threshold value (chosen uniformly between the minimum and maximum values of the input variable in the local subset of samples).

3.3 Parameters

Tree-based (ensemble) methods have several parameters whose values must be set by the user:

- The parameters related to the chosen stop-splitting criterion, such as N_{\min} , the minimal number of samples that a leaf node must contain. Increasing the value of N_{\min} results in smaller trees and hence models with a higher bias (i.e., more

prone to underfitting) and a lower variance (i.e., less prone to overfitting). Its optimal value depends on the level of noise contained in the learning sample. The noisier the data, the higher the optimal value of N_{\min} . Usually, N_{\min} is fixed to 1 for ensemble methods, so that each tree of the ensemble is fully developed.

- K , the number of input variables that are randomly chosen at each node of a tree. This parameter thus determines the level of randomization of the trees. A smaller value of K results in more randomized trees. The optimal value of K is problem-dependent, but $K = \sqrt{m}$ and $K = m$, where m is the number of input variables, are usually good default values [30].
- T , the number of trees in an ensemble. It can be shown that the higher the number of trees, the lower the generalization error [3, 29]. Therefore, the chosen value of T is a compromise between model accuracy and computing times.

3.4 Variable Importance Measures

One interesting characteristic of tree-based methods is the possibility to compute from a tree an importance score for each input variable. This score measures the relevance of a variable for the prediction of the output. In the case of regression, an importance measure that can be used is based on the reduction of the variance of the output at each test node \mathcal{N} , i.e., $I(\mathcal{N})$ as defined in Eq. (5). For a single tree, the overall importance w_i of one variable X_i is then computed by summing the $I(\mathcal{N})$ values of all the tree nodes where X_i is used to split:

$$w_i = \sum_{k=1}^p I(\mathcal{N}_k) \mathbb{1}_{\mathcal{N}_k}(X_i), \quad (6)$$

where p is the number of test nodes in the tree and \mathcal{N}_k denotes the k -th test node. $\mathbb{1}_{\mathcal{N}_k}(X_i)$ is a function that is equal to one if X_i is the variable selected at node \mathcal{N}_k and zero otherwise. The features that are not selected at all thus obtain an importance value of zero and those that are selected close to the root node of the tree typically obtain high scores. Variable importance measures can be easily extended to ensembles, simply by averaging importance scores over all the trees of the ensemble. The resulting importance measure is then even more reliable because of the variance reduction effect resulting from this averaging.

In the context of the Random forest method, an alternative procedure was proposed to compute the importance of a variable [29]. For each tree that was learned, this procedure consists in computing the prediction accuracy of the tree on the out-of-bag samples (i.e., the training instances that were not present in the bootstrap sample used to build the tree), before and after randomly permuting the values of the corresponding variable in

these samples. The reduction of the tree accuracy that is obtained after the permutation is then computed, and the importance of the variable is given by the average accuracy reduction over all the trees of the ensemble. While this procedure has some advantages with respect to the variance reduction-based measure [31], it gives in most practical applications very similar results while being much more computationally demanding. Furthermore, it does not extend to methods that do not consider bootstrap sampling, like the Extra-Trees.

4 Tree-Based Approaches for Gene Network Inference

This section presents several approaches based on decision tree algorithms that were developed for the unsupervised inference of GRNs. In particular, we start by describing in detail the state-of-the-art GENIE3 approach.

4.1 GENIE3

GENIE3, for “GEne Network Inference with Ensemble of trees,” uses ensembles of regression trees to infer GRNs from steady-state expression data. (Many people think that the digit 3 in the acronym GENIE3 indicates a third version of the algorithm. This is however not the case. The presence of the digit 3 is actually due to the fact that the word “three” sounds exactly like the word “tree,” when pronounced with a (strong) French accent.)

In what follows, we define an expression dataset from which to infer the network as a collection of N measurements:

$$D = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}, \quad (7)$$

where $\mathbf{x}_k \in \mathbb{R}^G$, $k = 1, \dots, N$ is the vector of expression values of G genes in the k -th experiment:

$$\mathbf{x}_k = (x_k^1, x_k^2, \dots, x_k^G)^\top.$$

The goal of GENIE3 is to exploit the expression dataset D to assign weights $w_{i,j} > 0$, $(i, j = 1, \dots, G)$ to putative regulatory links from any gene g_i to any gene g_j , with the aim of yielding larger values for weights that correspond to actual regulatory interactions. GENIE3 returns directed and unsigned edges, which means that $w_{i,j}$ can take a different value than $w_{j,i}$, and when g_i is connected to g_j , the former can be either an activator or a repressor of the latter.

To solve the network inference problem, GENIE3 decomposes the problem of recovering a network of G genes into G different subproblems, where each of these subproblems consists in identifying the regulators of one of the genes of the network. The method makes the assumption that the expression of each gene in a given

condition is a function of the expression of the other genes in the same condition (plus some random noise). Denoting by \mathbf{x}_k^{-j} the vector containing the expression values in the k -th experiment of all the genes except gene g_j :

$$\mathbf{x}_k^{-j} = (x_k^1, \dots, x_k^{j-1}, x_k^{j+1}, \dots, x_k^G)^\top,$$

we can write:

$$x_k^j = f_j(\mathbf{x}_k^{-j}) + \varepsilon_k, \quad \forall k, \quad (8)$$

where ε_k is a random noise with zero mean (conditionally to \mathbf{x}_k^{-j}). GENIE3 further makes the assumption that the function f_j only exploits the expression in \mathbf{x}_k^{-j} of the genes that are direct regulators of g_j , i.e., genes that are directly connected to g_j in the targeted network. Recovering the regulatory links pointing to the target gene g_j thus amounts to finding those genes whose expression is predictive of the expression of g_j . In machine learning terminology, this can be considered a *feature selection* problem (in regression) for which many solutions can be found in the literature. The solution that is used by GENIE3 exploits the variable importance scores derived from tree ensemble models.

The GENIE3 procedure is illustrated in Fig. 3 and works as follows:

- For $j = 1$ to G :
 - Generate the learning sample of input–output pairs for gene g_j :

$$LS^j = \{(\mathbf{x}_k^{-j}, x_k^j), k = 1, \dots, N\}. \quad (9)$$

- Learn an ensemble of trees from LS^j using the Random forest or Extra-Trees algorithm.
- From the learned tree model, compute variable importance scores $w_{i,j}$ for all the genes g_i (except g_j itself). These importance scores are computed as sums of variance reductions (Eq. (6)).
- Use $w_{i,j}$ as weight for the regulatory link directed from g_i to g_j .

Note that in GENIE3, it is possible—and even advisable—to restrict the set of candidate regulators to a subset of the genes only (rather than using all the G genes as candidate regulators). This can be useful when we know which genes are transcription factors, for example. In that case, the learning sample LS^j is constructed with only those transcription factors as input genes.

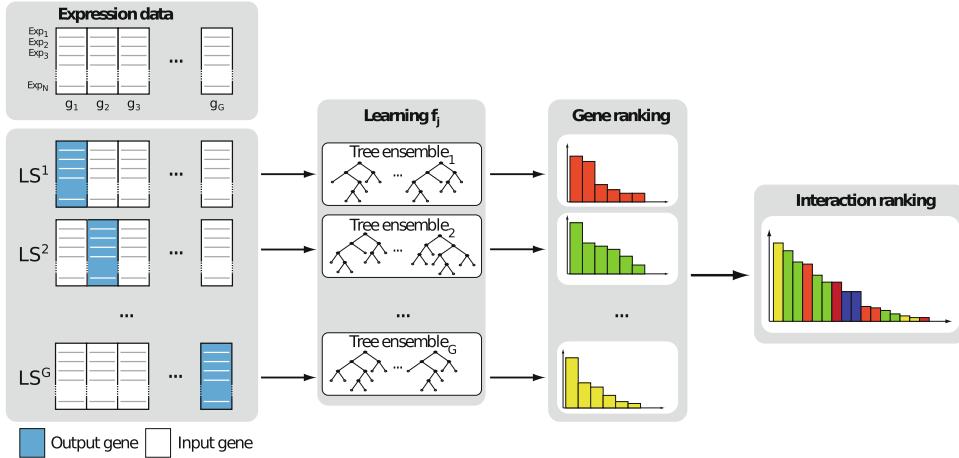


Fig. 3 GENIE3 procedure. For each gene $g_j, j = 1, \dots, G$, a learning sample \mathbf{LS}^j is generated with the expression levels of g_j as output values and the expression levels of all the other genes as input values. An ensemble of trees is learned from \mathbf{LS}^j and a variable importance score $w_{i,j}$ is computed for each input gene g_i . The score $w_{i,j}$ is then used as weight for the regulatory link directed from g_i to g_j . Figure reproduced from [4]

4.1.1 Output Normalization

In the GENIE3 procedure, each tree-based model yields a separate ranking of the genes as potential regulators of a target gene g_j , derived from importance scores $w_{i,j}$. It can be shown that the sum of the importance scores of all the input variables for a tree is equal to the total variance of the output variable explained by the tree, which in the case of unpruned trees (as they are in the case of tree ensembles) is usually very close to the initial total variance of the output:

$$\sum_{i \neq j} w_{i,j} \approx N\text{Var}_j(\mathbf{LS}^j), \quad (10)$$

where \mathbf{LS}^j is the learning sample from which the tree was built (i.e., \mathbf{LS}^j in the case of the Extra-Trees method and a bootstrap sample in the case of the Bagging and Random forest methods) and $\text{Var}_j(\mathbf{LS}^j)$ is the variance of the target gene g_j estimated in the corresponding learning sample. As a consequence, if the regulatory links are simply ranked according to the weights $w_{i,j}$, this is likely to introduce a bias where some putative regulatory links will have a high weight simply because they are directed towards highly variable genes. To avoid this bias, the expression of the target gene g_j is normalized to have a unit variance in the learning sample \mathbf{LS}^j , before applying the tree-based ensemble method:

$$\mathbf{x}^j \leftarrow \frac{\mathbf{x}^j}{\sigma^j}, \quad \forall j, \quad (11)$$

Table 1
Running times of the different GENIE3 implementations

Network	Python	MATLAB	R
DREAM4 ($N = 100$, $G = 100$, $n_{TF} = 100$)	130 s	65 s	50 s
<i>E. coli</i> ($N = 805$, $G = 4511$, $n_{TF} = 334$)	20 h	20 h	14 h

N number of samples, G number of genes, n_{TF} number of transcription factors

where $\mathbf{x}^j \in \mathbb{R}^N$ is the vector of expression levels of g_j in the N experiments and σ^j denotes its standard deviation. This normalization indeed implies that the different weights inferred from different models predicting the different gene expressions are comparable.

4.1.2 Software Availability

Python, MATLAB, and R implementations of GENIE3, as well as tutorials explaining how to run them, are available from:

<https://github.com/vahuynh/GNIE3>

4.1.3 Computational Complexity

The computational complexity of the Random forest and Extra-Trees algorithms is on the order of $O(TKN \log N)$, where T is the number of trees, N is the learning sample size, and K is the number of randomly selected genes at each node of a tree. GENIE3's complexity is thus on the order of $O(GTKN \log N)$ since it requires to build an ensemble of trees for each of the G genes. The complexity of the whole procedure is thus log linear with respect to the number of measurements and, at worst, quadratic with respect to the number of genes (when $K = G - 1$).

To give an idea of computing times, Table 1 shows the time needed for GENIE3 to infer one network of the DREAM4 *Multifactorial* challenge (100 experiments and 100 genes) and the *E. coli* network of the DREAM5 challenge (805 experiments and 4511 genes, among which 334 known transcription factors). In each case, GENIE3 was run with Random forest, $T = 1000$ trees per ensemble and $K = \sqrt{n_{TF}}$, where n_{TF} is the number of candidate regulators (i.e., $n_{TF} = 100$ for DREAM4 and $n_{TF} = 334$ for *E. coli*). These computing times were measured on a 16GB RAM, Intel Xeon E5520 2.27 GHz computer.

Note that if needed, the GENIE3 algorithm can be easily parallelized as the G feature selection problems, as well as the different trees in an ensemble, are independent of each other.

4.1.4 Parameter Analysis

Figure 4 shows the performances and running times of GENIE3, for two networks of the DREAM5 challenge (an artificial in silico network and a real *E. coli* network), when varying the values of the different parameters of GENIE3. The performances were measured using the area under the precision-recall curve (AUPR) metric, which assesses the quality of the ranking of interactions

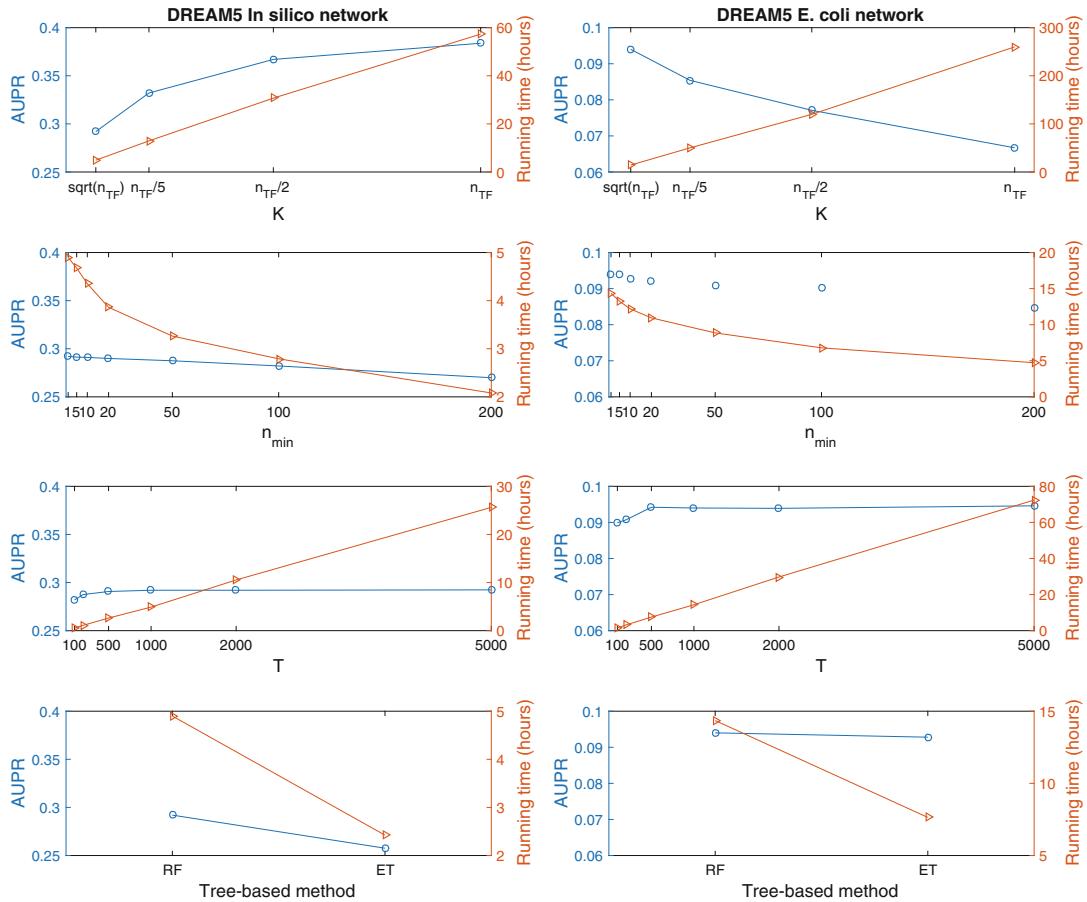


Fig. 4 AUPRs (blue circles) and running times (orange triangles) of GENIE3, when varying the values of the parameters K (number of randomly chosen candidate regulators at each split node of a tree), n_{\min} (minimum number of samples at a leaf) and T (number of trees per ensemble), and when using either Random forest (RF) or Extra-Trees (ET) as tree-based algorithm. When varying the values of one parameter, the values of the remaining parameters were set to their default values. The default values are: $K = \sqrt{n_{TF}}$, where n_{TF} is the number of transcription factors, $n_{\min} = 1$, $T = 1000$ and the tree-based method is the Random forest algorithm. The results shown in this figure were obtained by using the R implementation of GENIE3. In silico dataset: 805 samples, 1643 genes, 195 transcription factors. *E. coli* dataset: 805 samples, 4511 genes, 334 transcription factors

returned by a method. A perfect ranking, where all the true interactions are ranked at the top, yields an AUPR equal to 1, while a random ranking returns an AUPR equal to the proportion of true interactions among all the possible interactions (which is typically very small, since regulatory networks are very sparse).

Clearly, the parameter with the highest impact is K , i.e., the number of randomly selected candidate regulators at each tree node. Its optimal value is very dataset-dependent: increasing the value of K improves the predictions (i.e., the AUPR is increased) for the in silico network, while the opposite is observed on the *E. coli* network. This difference between the two networks can

Table 2
AUPRs of GENIE3 for the DREAM5 in silico network, when noise is added to the data

	Noise $\mathcal{N}(0,0.25)$	Noise $\mathcal{N}(0,0.5)$
$K = \sqrt{n_{TF}}$	0.1741	0.0435
$K = n_{TF}/5$	0.1893	0.0436
$K = n_{TF}/2$	0.2004	0.0435
$K = n_{TF}$	0.2059	0.0426
Random	0.0125	0.0125

probably be explained by the fact that the *E. coli* data contains more noise than the artificial data. We thus checked how the performance of GENIE3 varies when adding further noise to the artificial data, in the form of a Gaussian noise $\sim \mathcal{N}(0, 0.25)$ or $\mathcal{N}(0, 0.5)$ (Table 2). As expected, the predictions are worse when noise is added, for all the values of K . In the presence of a high amount of noise, increasing K from $\frac{n_{TF}}{2}$ to n_{TF} results in a (slightly) lower AUPR, a result closer to what is observed for the *E. coli* network. This could be explained by the fact that decreasing the value of K results in predictive tree-based models that overfit less the data and that are therefore more robust to the noise.

The other parameters of GENIE3 have only a minor impact on the performances (Fig. 4). In terms of AUPR, the best value of n_{\min} , i.e., the minimum number of samples at a leaf, is 1. Increasing n_{\min} allows to save some computational time, at only a small cost in terms of performances. Regarding the number T of trees per ensemble, we observe that 500 trees already allow to obtain good performances. Further increasing T only results in more computational time, without improving the AUPR. Finally, the Extra-Trees algorithm has slightly less good performances than Random forest, but is more computationally efficient.

4.2 Extensions of GENIE3

Several methods for GRN inference building on GENIE3 have been developed. The purpose of this section is to give a brief overview of these methods. For more details, the reader can refer to the original articles (referenced in the following subsections).

4.2.1 Analysis of Time Series Data

dynGENIE3 (for “dynamical GENIE3”) is a variant of GENIE3 that was developed for the analysis of time series of expression data [32]. dynGENIE3 assumes that the expression level x_j of gene g_j is modelled through the following ordinary differential equation (ODE):

$$\frac{dx_j(t)}{dt} = -\alpha_j x_j(t) + f_j(\mathbf{x}(t)), \quad (12)$$

where $\mathbf{x}(t)$ is the vector containing the expressions of all the G genes at time t and α_j is a parameter specifying the decay rate of x_j . In this ODE, it is assumed that the transcription rate of x_j is a (potentially non-linear) function f_j of the expression levels of the G genes. Like in GENIE3, this function f_j is learned in the form of an ensemble of regression trees and the regulators of g_j are then identified by computing the variable importance scores derived from the tree model. dynGENIE3 is therefore a semi-parametric approach, as the temporal evolution of each gene expression is modelled with a formal ODE while the transcription function in each ODE is learned in the form of a non-parametric (tree-based) model.

Given the observation time points t_1, t_2, \dots, t_T , the ODE (12) has the following finite approximation:

$$\frac{x_j(t_{k+1}) - x_j(t_k)}{t_{k+1} - t_k} + \alpha_j x_j(t_k) = f_j(\mathbf{x}(t_k)), \quad (13)$$

$$k = 1, \dots, T - 1,$$

and the function f_j can thus be learned using the following learning sample:

$$\text{LS}^j = \left\{ \left(\mathbf{x}(t_k), \frac{x_j(t_{k+1}) - x_j(t_k)}{t_{k+1} - t_k} + \alpha_j x_j(t_k) \right), k = 1, \dots, T - 1 \right\}. \quad (14)$$

The gene decay rates α_j in LS^j are parameters that are fixed by the user. Their values may be retrieved from the literature, since there exist many studies that experimentally measure the mRNA decay rates in different organisms. However, when such information is not available, a data-driven approach can be used to estimate the α_j value directly from the observed expressions \mathbf{x}_j of g_j . For example, a rough estimate of α_j can be obtained by assuming an exponential decay $e^{-\alpha_j t}$ between the highest and lowest values of \mathbf{x}_j .

4.2.2 Analysis of Genotype Data

Two extensions of GENIE3 were proposed for the joint analysis of expression and genotype data [33]. It is assumed that we have at our disposal a dataset containing the expression levels of G genes measured in N individuals, as well as the genotype value of one genetic marker for each of these genes in the same N individuals:

$$D = \{(\mathbf{x}_1, \mathbf{m}_1), (\mathbf{x}_2, \mathbf{m}_2), \dots, (\mathbf{x}_N, \mathbf{m}_N)\}, \quad (15)$$

where $\mathbf{x}_k \in \mathbb{R}^G$ and $\mathbf{m}_k \in \{0, 1\}^G$, $k = 1, \dots, N$ are respectively the vectors of expression levels and genotype values of the G genes in the k -th individual:

$$\begin{cases} \mathbf{x}_k = (x_k^1, x_k^2, \dots, x_k^G)^\top, \\ \mathbf{m}_k = (m_k^1, m_k^2, \dots, m_k^G)^\top. \end{cases} \quad (16)$$

Note that it is supposed that each genetic marker can have two possible genotype values only (0 or 1), as it would be the case for homozygous individuals.

To exploit such data, the first procedure, called GENIE3-SG-joint, assumes that a unique model f_j explains the expression of a gene \mathcal{G}_j in a given individual, knowing the expression levels and the genotype values of the different genes:

$$x_k^j = f_j(\mathbf{x}_k^{-j}, \mathbf{m}_k) + \varepsilon_k, \forall k, \quad (17)$$

where ε_k is a random noise. In the second procedure, called GENIE3-SG-sep, it is assumed that two different models f_j^x and f_j^m can both explain the expression of \mathcal{G}_j , either from the expression levels of the other genes, or from the genotype values:

$$\begin{cases} x_k^j = f_j^x(\mathbf{x}_k^{-j}) + \varepsilon_k, \forall k, \\ x_k^j = f_j^m(\mathbf{m}_k) + \varepsilon'_k, \forall k. \end{cases} \quad (18)$$

The functions f_j^x and f_j^m are therefore respectively learned from two different learning samples. Both GENIE3-SG-joint and GENIE3-SG-sep learn the different functions f_j as ensembles of trees and compute for each candidate regulator \mathcal{G}_i two scores $w_{i,j}^x$ and $w_{i,j}^m$, measuring respectively the importances of the expression and of the marker of \mathcal{G}_i when predicting the expression of \mathcal{G}_j . These two scores are then aggregated, either by a sum or a product, to obtain a single weight $w_{i,j}$ for the regulatory link directed from \mathcal{G}_i to \mathcal{G}_j .

4.2.3 Analysis of Single-Cell Data

GENIE3 is used as such in two frameworks that were developed for the inference of GRNs from single-cell transcriptomic data.

The framework developed by Ocone et al. uses GENIE3 to obtain a prior GRN, which is then refined using an ODE-based approach [34]. The whole procedure allows to identify the GRN as well as the parameters of the ODEs that are used to model the gene expression dynamics.

In the SCENIC framework [35], GENIE3 is used in a first step to identify co-expression modules, i.e., groups of genes that are regulated by the same transcription factor. In a second step, a motif enrichment analysis is performed for each module, and only the modules such that the target genes show an enrichment of a motif of the corresponding transcription factor are retained. The activity of each module in each cell is then evaluated using the single-cell expression data and the activity levels of the different modules are used to perform cell clustering.

4.2.4 Exploitation of Prior Knowledge

The iRafNet method [36] allows to take into account prior information that we have about the network, in the form of prior weights associated with the different network edges. In the original article introducing iRafNet, the prior weights are obtained from diverse types of data, such as protein–protein interaction data or knockout data. To exploit the prior weights, iRafNet uses the same framework as GENIE3, but with a modified version of the Random forest algorithm. At each tree node, instead of randomly sampling K input variables according to a uniform distribution, the K variables are sampled with a bias that favors the variables with a higher prior weight.

4.2.5 Inference of Context-Specific Networks

Let us assume that we have different expression datasets respectively related to different contexts (e.g., different pathological conditions). One could then be interested in identifying a GRN that is specific to each of these contexts. To achieve such goal, one could apply a network inference algorithm like GENIE3 to each dataset. However, when the contexts are related (e.g., different subclasses of a cancer), one can expect the different networks to have a lot of commonalities. In this case, approaches that jointly analyze the different datasets will potentially yield better performances, as they will assign higher weights to regulatory links that are active in a higher number of contexts. Many of such joint approaches are based on Gaussian graphical models (*see*, e.g., [37–39]). An approach based on trees, called JRF (for “Joint Random Forest”) [40], has also been proposed. Given D datasets, JRF consists, for each target gene, in simultaneously learning D tree models. When learning D regression trees in parallel, the idea is to select the same input variable at the same node position in the D different trees. More specifically, for a given input variable g_i , let c_i^d be the threshold value that yields the best split at test node \mathcal{N} in the d -th tree ($d = 1, \dots, D$). c_i^d is thus the threshold value that maximizes the variance reduction $I_i^d(\mathcal{N})$, as defined in Eq. (5), among all the possible threshold values for g_i . JRF then selects, at node \mathcal{N} in all the D trees, the input variable g_{i^*} that maximizes:

$$g_{i^*} = \arg \max_i \sum_{d=1}^D \frac{I_i^d(\mathcal{N})}{N_d}, \quad (19)$$

where N_d is the number of samples in the d -th dataset. The importance score of a candidate regulator g_i in the d -th context is then the sum of output variance reductions (as defined in Eq. (6)), computed by propagating the samples of the d -th dataset in the d -th tree model. By selecting, for a given target gene, the same candidate regulators in the D tree models, JRF enforces the similarity between the inferred networks. However, since the importance score of g_i in the d -th context is computed by using the samples that are related to this context, JRF also allows to identify

regulatory links that are active in only one or a few contexts. Note that when there is only one context, JRF is equivalent to GENIE3.

4.3 Other Tree-Based Approaches

Besides GENIE3, other tree-based approaches have been proposed for the unsupervised inference of GRNs, which use different types of trees in different frameworks. In [41], networks are reconstructed by learning a single classification tree for each target gene, predicting the state of the gene (up- or down-regulated) from the expression levels of the other genes. In [42], Segal et al. propose a method that partitions the genes into *modules*, such that genes in the same module have the same regulators and the same regulatory model. The regulatory model of each module is represented by one probabilistic regression tree (where each leaf is associated with a probabilistic distribution of the output variable). Inspired by the work of Segal et al. the LeMoNe algorithm [43] also learns module networks, where the regulatory model for each module is in the form of an ensemble of *fuzzy* decision trees. In [44] M5' model trees are used, i.e., regression trees with linear models at the leaves. In [45], networks are reconstructed by fitting a dynamical model of the gene expressions, where one of the terms of the model is learned in the form of an ensemble of decision trees. Tree-based methods were also used to model expression data jointly with other types of data, such as motif counts in the promoter regions of the gene or transcription factor-binding data [46–49]. Another example is [50], which extends the module network procedure to exploit both expression and genotype data.

5 Discussion

In this chapter, we presented GENIE3 and other tree-based approaches that were developed for the inference of gene regulatory networks from expression data. The main advantages of GENIE3 are its non-parametric nature, its ability to detect multivariate interacting effects between candidate regulators, and the fact that it has very few parameters. However, like any method, GENIE3 also has its own limitations and could thus be improved along several directions, discussed below.

A first limitation of GENIE3 is that it provides a ranking of the putative regulatory links, rather than a network topology. Since tree-based importance scores are not interpretable from a statistical point of view, choosing a threshold value to distinguish present and absent edges is not a trivial task. Several methods were proposed for addressing the problem of selecting, from tree-based importance scores, the input variables that are relevant for output prediction [51, 52]. These methods could in principle be applied in the context of GENIE3 in order to select the regulators of each target gene. However, most of them are based on multiple—

say 1000—reruns of the tree-based algorithm. If one wishes to incorporate such feature selection approaches into GENIE3, one would need to learn $1000 \times G$ ensembles of trees, where G is the number of genes, which would be impractical for a large value of G . Another property of these methods is that they are designed to identify the *maximal* subset of relevant variables, i.e., all the variables that convey at least some information about the output. For that reason, these methods are not appropriate for the network inference problem. Even if there is no direct edge from gene g_i to gene g_j in the true network, the expression of g_i can still be predictive of the expression of g_j , through one or several other genes (e.g., g_i regulates some gene g_k , which in turn regulates g_j). In conclusion, each gene of the network is indirectly regulated by (almost) all the other genes, but most of these indirect edges are considered false positives since they are not part of the true network. To avoid the inclusion of such indirect effects, one would need a feature selection method able to determine a *minimal* subset of variables that convey all the information about an output (and thus make all the other variables conditionally irrelevant). An optimal treatment of this problem would probably require to adopt a more global approach that exploits jointly the G individual rankings related to the different target genes, respectively.

GENIE3 could also be improved on the way the variable importance scores are normalized. Given the current normalization (Eq. (11)), the scores of all the putative edges directed towards a given target gene sum up to one. As a consequence, the importance scores that are derived from different tree models are not entirely comparable. For example, let us assume that gene g_1 has a single regulator g_2 , and that gene g_3 has two regulators g_4 and g_5 . With a sufficient amount of data, GENIE3 will assign a score of 1 to the edge $g_2 \rightarrow g_1$, but a score of only 0.5 to $g_4 \rightarrow g_3$ and $g_5 \rightarrow g_3$. An optimal way of normalizing the importance score is thus still needed at this stage.

So far, GENIE3 has only been evaluated in an empirical way. It would however be interesting to better characterize the method—in particular the tree-based importance scores used within—from a theoretical point of view. This would actually constitute an important contribution in the machine learning field, as there has been very few works focusing on the theoretical analysis of the importance measures derived from tree ensembles [53–55].

Despite the good scalability of GENIE3 with respect to some families of methods (such as methods based on differential equations or Bayesian methods), a substantial amount of time is still needed to reconstruct a large network (e.g., it takes 20 h to reconstruct the *E. coli* network from 805 samples, *see* Table 1). With the recent developments in single-cell RNA-seq technologies, datasets with a size of the order of 100K cells are becoming available. Speeding up the GENIE3 algorithm would be necessary if one wishes to apply it on such large datasets.

Acknowledgements

VAHT is a Post-doctoral Fellow of the F.R.S.-FNRS.

References

1. Geurts P, Irrthum A, Wehenkel L (2009) Supervised learning with decision tree-based methods in computational and systems biology. *Mol Biosyst* 5(12):1593–1605
2. Boulesteix AL, Janitz S, Kruppa J, König IR (2012) Overview of random forest methodology and practical guidance with emphasis on computational biology and bioinformatics. *Wiley Interdiscip Rev Data Min Knowl Disc* 2(6):493–507
3. Biau G, Scornet E (2016) A random forest guided tour. *TEST* 25(2):197–227
4. Huynh-Thu VA, Irrthum A, Wehenkel L, Geurts P (2010) Inferring regulatory networks from expression data using tree-based methods. *PLoS ONE* 5(9):e12776
5. Marbach D, Costello JC, Küffner R, Vega N, Prill RJ, Camacho DM, Allison KR, the DREAM5 Consortium, Kellis M, Collins JJ, Stolovitzky G (2012) Wisdom of crowds for robust gene network inference. *Nat Methods* 9(8):796–804
6. Omranian N, Eloundou-Mbebi JMO, Mueller-Roeber B, Nikoloski Z (2016) Gene regulatory network inference using fused lasso on multiple data sets. *Sci Rep* 6:20533
7. Kiani NA, Zenil H, Olczak J, Tegnér J (2016) Evaluating network inference methods in terms of their ability to preserve the topology and complexity of genetic networks. *Semin Cell Dev Biol* 51:44–52
8. Bellot P, Olsen C, Salembier P, Oliveras-Vergés A, Meyer PE (2015) NetBenchmark: a bioconductor package for reproducible benchmarks of gene regulatory network inference. *BMC Bioinf* 16:312
9. Maetschke SR, Madhamshettiar PB, Davis MJ, Ragan MA (2014) Supervised, semi-supervised and unsupervised inference of gene regulatory networks. *Brief Bioinform* 15(2):195–211
10. Zhang X, Liu K, Liu ZP, Duval B, Richer JM, Zhao XM, Hao JK, Chen L (2013) NARROMI: a noise and redundancy reduction technique improves accuracy of gene regulatory network inference. *Bioinformatics* 29(1):106–113
11. Feizi S, Marbach D, Médard M, Kellis M (2013) Network deconvolution as a general method to distinguish direct dependencies in networks. *Nat Biotechnol* 31:726–733
12. Madhamshettiar PB, Maetschke SR, Davis MJ, Reverter A, Ragan MA (2012) Gene regulatory network inference: evaluation and application to ovarian cancer allows the prioritization of drug targets. *Genome Med* 4(5):41
13. Qi J, Michoel T (2012) Context-specific transcriptional regulatory network inference from global gene expression maps using double two-way t-tests. *Bioinformatics* 28(18):2325–2332
14. Imam S, Noguera DR, Donohue TJ (2015) An integrated approach to reconstructing genome-scale transcriptional regulatory networks. *PLoS Comput Biol* 11(2):e1004103
15. Arrieta-Ortiz ML, Hafemeister C, Bate AR, Chu T, Greenfield A, Shuster B, Barry SN, Gallitto M, Liu B, Kacmarczyk T, Santoriello F, Chen J, Rodrigues CD, Sato T, Rudner DZ, Driks A, Bonneau R, Eichenberger P (2015) An experimentally supported model of the *Bacillus subtilis* global transcriptional regulatory network. *Mol Syst Biol* 11(11):839
16. Carrera J, Estrela R, Luo J, Rai N, Tsoukalas A, Tagkopoulos I (2014) An integrative, multi-scale, genome-wide model reveals the phenotypic landscape of *Escherichia coli*. *Mol Syst Biol* 10(7):735
17. Sabaghian E, Drebert Z, Inzé D, Saeys Y (2015) An integrated network of *Arabidopsis* growth regulators and its use for gene prioritization. *Sci Rep* 5:17617
18. Taylor-Teeple M, Lin L, de Lucas M, Turco G, Toal TW, Gaudinier A, Young NF, Trabucco GM, Veling MT, Lamothe R, Handakumbura PP, Xiong G, Wang C, Corwin J, Tsoukalas A, Zhang L, Ware D, Pauly M, Kliebenstein DJ, Dehesh K, Tagkopoulos I, Breton G, Pruneda-Paz JL, Ahnert SE, Kay SA, Hazen SP, Brady SM (2015) An *Arabidopsis* gene regulatory network for secondary cell wall synthesis. *Nature* 517(7536):571–575
19. Marchand G, Huynh-Thu VA, Kane N, Arribat S, Varès D, Rengel D, Balzergue S, Rieseberg L, Vincourt P, Geurts P, Vignes M, Langlade NB

- (2014) Bridging physiological and evolutionary time-scales in a gene regulatory network. *New Phytol* 203(2):685–696
20. Potier D, Davie K, Hulselmans G, Naval Sanchez M, Haagen L, Huynh-Thu V, Koldere D, Celik A, Geurts P, Christiaens V, Aerts S (2014) Mapping gene regulatory networks in *Drosophila* eye development by large-scale transcriptome perturbations and motif inference. *Cell Rep* 9(6):2290–2303
 21. Jo J, Hwang S, Kim HJ, Hong S, Lee JE, Lee SG, Baek A, Han H, Lee JI, Lee I, Lee DR (2016) An integrated systems biology approach identifies positive cofactor 4 as a factor that increases reprogramming efficiency. *Nucleic Acids Res* 44(3):1203–1215
 22. Acquaah-Mensah GK, Taylor RC (2016) Brain in situ hybridization maps as a source for reverse-engineering transcriptional regulatory networks: Alzheimer's disease insights. *Gene* 586(1):77–86
 23. Verfaillie A, Imrichova H, Atak ZK, Dewaele M, Rambow F, Hulselmans G, Christiaens V, Svetlichnyy D, Luciani F, Van den Mooter L, Claerhout S, Fiers M, Journe F, Ghanem GE, Herrmann C, Halder G, Marine JC, Aerts S (2015) Decoding the regulatory landscape of melanoma reveals TEADS as regulators of the invasive cell state. *Nat Commun* 6:6683
 24. Ko JH, Gu W, Lim I, Zhou T, Bang H (2014) Expression profiling of mitochondrial voltage-dependent anion channel-1 associated genes predicts recurrence-free survival in human carcinomas. *PLoS ONE* 9(10):e110094
 25. Hastie T, Tibshirani R, Friedman J (2009) The elements of statistical learning: data mining, inference and prediction, 2nd edn. Springer, Berlin
 26. Bishop CM (2006) Pattern recognition and machine learning. Springer, Berlin
 27. Breiman L, Friedman JH, Olsen RA, Stone CJ (1984) Classification and regression trees. Wadsworth International (California), Belmont
 28. Breiman L (1996) Bagging predictors. *Mach Learn* 24(2):123–140
 29. Breiman L (2001) Random forests. *Mach Learn* 45(1):5–32
 30. Geurts P, Ernst D, Wehenkel L (2006) Extremely randomized trees. *Mach Learn* 36(1):3–42
 31. Strobl C, Boulesteix AL, Zeileis A, Hothorn T (2007) Bias in random forest variable importance measures: illustrations, sources and a solution. *BMC Bioinf* 8:25
 32. Huynh-Thu VA, Geurts P (2018) dynGENIE3: dynamical GENIE3 for the inference of gene networks from time series expression data. *Sci Rep* 8(1):3384
 33. Huynh-Thu VA, Wehenkel L, Geurts P (2013) Gene regulatory network inference from systems genetics data using tree-based methods. In: de la Fuente A (ed) Gene network inference - verification of methods for systems genetics data. Springer, Berlin, pp 63–85
 34. Ocone A, Haghverdi L, Mueller NS, Theis FJ (2015) Reconstructing gene regulatory dynamics from high-dimensional single-cell snapshot data. *Bioinformatics* 31(12):i89–i96
 35. Aibar S, González-Blas CB, Moerman T, Huynh-Thu VA, Imrichova H, Hulselmans G, Rambow F, Marine JC, Geurts P, Aerts J, van den Oord J, Atak ZK, Wouters J, Aerts S (2017) SCENIC: single-cell regulatory network inference and clustering. *Nat Methods* 14:1083–1086
 36. Petralia F, Wang P, Yang J, Tu Z (2015) Integrative random forest for gene regulatory network inference. *Bioinformatics* 31(12):i197–i205
 37. Chiquet J, Grandvalet Y, Ambroise C (2011) Inferring multiple graphical structures. *Stat Comput* 21(4):537–553
 38. Mohan K, London P, Fazel M, Witten D, Lee SI (2014) Node-based learning of multiple gaussian graphical models. *J Mach Learn Res* 15(1):445–488
 39. Tian D, Gu Q, Ma J (2016) Identifying gene regulatory network rewiring using latent differential graphical models. *Nucleic Acids Res* 44(17):e140
 40. Petralia F, Song WM, Tu Z, Wang P (2016) New method for joint network analysis reveals common and different coexpression patterns among genes and proteins in breast cancer. *J Proteome Res* 15(3):743–754
 41. Soinov LA, Krestyaninova MA, Brazma A (2003) Towards reconstruction of gene networks from expression data by supervised learning. *Genome Biol* 4(1):R6
 42. Segal E, Shapira M, Regev A, Pe'er D, Botstein D, Koller D, Friedman N (2003) Module networks: identifying regulatory modules and their condition-specific regulators from gene expression data. *Nat Genet* 34:166–176
 43. Joshi A, De Smet R, Marchal K, Van de Peer Y, Michoel T (2009) Module networks revisited:

- computational assessment and prioritization of model predictions. *Bioinformatics* 25(4):490–496
44. Nepomuceno-Chamorro IA, Aguilar-Ruiz JS, Riquelme JC (2010) Inferring gene regression networks with model trees. *BMC Bioinf* 11: 517
45. Huynh-Thu VA, Sanguinetti G (2015) Combining tree-based and dynamical systems for the inference of gene regulatory networks. *Bioinformatics* 31(10):1614–1622
46. Middendorf M, Kundaje A, Wiggins C, Freund Y, Leslie C (2004) Predicting genetic regulatory response using classification. *Bioinformatics* 20(Suppl_1):i232–i240
47. Phuong TM, Lee D, Lee KH (2004) Regression trees for regulatory element identification. *Bioinformatics* 20(5):750–757
48. Ruan J, Zhang W (2006) A bi-dimensional regression tree approach to the modeling of gene expression regulation. *Bioinformatics* 22(3):332–340
49. Xiao Y, Segal MR (2009) Identification of yeast transcriptional regulation networks using multivariate random forests. *PLoS Comput Biol* 5(6):e1000414
50. Lee SI, Pe'er D, Dudley AM, Church GM, Koller D (2006) Identifying regulatory mechanisms using individual variation reveals key role for chromatin modification. *Proc Natl Acad Sci* 103(38):14062–14067
51. Huynh-Thu VA, Saeys Y, Wehenkel L, Geurts P (2012) Statistical interpretation of machine learning-based feature importance scores for biomarker discovery. *Bioinformatics* 28(13):1766–1774
52. Degenhardt F, Seifert S, Szymczak S (2017) Evaluation of variable selection methods for random forests and omics data sets. *Brief Bioinf* bbx124. <https://doi.org/10.1093/bib/bbx124>
53. Ishwaran H (2007) Variable importance in binary regression trees and forests. *Electron J Stat* 1:519–537
54. Louppe G, Wehenkel L, Sutera A, Geurts P (2013) Understanding variable importances in forests of randomized trees. In: Burges CJC, Bottou L, Welling M, Ghahramani Z, Weinberger KQ (eds) *Advances in neural information processing systems*, vol 26. Curran Associates, Inc., Red Hook, pp 431–439
55. Sutera A, Louppe G, Huynh-Thu VA, Wehenkel L, Geurts P (2016) Context-dependent feature analysis with random forests. In: *Proceedings of the thirty-second conference on uncertainty in artificial intelligence, UAI'16*. AUAI Press, Corvallis, pp 716–725



Chapter 9

Tree-Based Learning of Regulatory Network Topologies and Dynamics with Jump3

Vân Anh Huynh-Thu and Guido Sanguinetti

Abstract

Inference of gene regulatory networks (GRNs) from time series data is a well-established field in computational systems biology. Most approaches can be broadly divided in two families: model-based and model-free methods. These two families are highly complementary: model-based methods seek to identify a formal mathematical model of the system. They thus have transparent and interpretable semantics but rely on strong assumptions and are rather computationally intensive. On the other hand, model-free methods have typically good scalability. Since they are not based on any parametric model, they are more flexible than model-based methods, but also less interpretable.

In this chapter, we describe Jump3, a hybrid approach that bridges the gap between model-free and model-based methods. Jump3 uses a formal stochastic differential equation to model each gene expression but reconstructs the GRN topology with a nonparametric method based on decision trees. We briefly review the theoretical and algorithmic foundations of Jump3, and then proceed to provide a step-by-step tutorial of the associated software usage.

Key words Hybrid method, Decision trees, Stochastic differential equations, Gaussian processes

1 Introduction

Many methods have been proposed for the inference of gene regulatory networks (GRNs) from gene expression time series data. These methods can be broadly divided in two families, that we will name here *model-based* and *model-free* approaches. Model-based methods first define a formal mathematical model of the system, usually expressed with differential or difference equations. They then reconstruct the GRN topology by learning the parameters of the mathematical model from observed data (including discrete parameters such as network structure). Model-based methods have several advantages: they have a transparent model, which makes them interpretable for human experts, and once learned, this model can be used for predicting the dynamical

behavior of the system in new experimental conditions. However, model-based methods have also several drawbacks: they tend to be computationally intensive, and by essence they make very strong assumptions about the system dynamics (e.g., linear dependencies between genes), which are not always justified biologically. Model-free methods on the other hand infer the GRN without defining any prior mathematical model and directly estimate the gene dependencies from the data, using more or less sophisticated statistical measures or machine learning-based analyses. These model-free methods are typically highly scalable and can thus be used for the inference of very large networks. Since they are not based on any parametric model, they are much more flexible than model-based methods, but also less interpretable. Moreover, they cannot be used for prediction in a straightforward way.

This chapter presents a hybrid approach to the network inference problem, called *Jump3* [1], that bridges the gap between model-based and model-free methods. *Jump3* uses a formal stochastic differential equation to model the dynamics of each gene of the network but reconstructs the GRN topology with a nonparametric method based on decision trees (*see Chapter 8* for more details about tree-based GRN inference methods). *Jump3* combines the computational efficiency of model-free methods with a biophysically plausible and interpretable model of gene expression; its performance on benchmark tests (described in [1]) is competitive with, or better than, the state-of-the-art, and a fully documented MATLAB implementation is freely available at <https://github.com/vahuynh/Jump3>.

In the rest of the chapter, we start by briefly reviewing the mathematical and algorithmic bases of *Jump3*, with the main aim to illustrate how the model-based approach is embedded into a tree learning framework. We then provide a step-by-step tutorial on the usage of the *Jump3* software, using the synthetic data set accompanying the software as an example.

2 The *Jump3* Framework

Let us assume that we observe the expression levels of G genes at T time points following some perturbation of the system:

$$TS = \{\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, \dots, \hat{\mathbf{x}}_G\}, \quad (1)$$

where $\hat{\mathbf{x}}_i$ is the vector containing the observed expressions of gene \mathcal{G}_i at the T time points:

$$\hat{\mathbf{x}}_i = (\hat{x}_{i,1}, \hat{x}_{i,2}, \dots, \hat{x}_{i,T})^\top \quad (2)$$

From this time series data TS , the goal of Jump3 is to learn a weight for each putative regulatory link of the network, with higher weights assigned to links that are true regulatory interactions. Jump3 is a semi-parametric approach: a parametric model is used for modelling the expression of each gene (Subheading 2.1), but a nonparametric model is used for modelling the regulatory interactions between the genes (Subheading 2.2).

2.1 The Parametric Part: Modelling Gene Expression

Jump3 models the dynamics of the expression x_i of each gene g_i using the *on/off model* of gene expression [2], which uses the following stochastic differential equation (SDE):

$$dx_i = (\alpha_i \mu_i(t) + b_i - \lambda_i x_i)dt + \sigma_{sys} dw(t) \quad (3)$$

In this model, the transcription rate of g_i depends on the state μ_i of the promoter of g_i , which can be either active ($\mu_i = 1$) or inactive ($\mu_i = 0$). $\Theta_i = \{\alpha_i, b_i, \lambda_i\}$ is the set of kinetic parameters. α_i represents the efficiency of the promoter in recruiting polymerase when being in the active state, b_i denotes the basal transcription rate, and λ_i is the exponential decay constant of x_i . The term $\sigma_{sys} dw(t)$ represents a white noise with variance σ_{sys}^2 .

Let us assume for the moment that we know the trajectory of the promoter state, i.e., that we know the state $\mu_i(t)$ at any time t . In that case, the SDE 3 is linear and it can be shown that its solution $x_i(t)$ is equivalent to a Gaussian process [3]. Generally speaking, the fact that $x_i(t)$ is a Gaussian process means that for any subset of T time points $\{t_1, t_2, \dots, t_T\}$, the vector $(x_i(t_1), x_i(t_2), \dots, x_i(t_T))^\top$ follows a multivariate normal distribution with some mean $\mathbf{m}_i \in \mathbb{R}^T$ and covariance $\mathbf{C}_i \in \mathbb{R}^{T \times T}$. In the specific case of the on/off model, it can be shown that the mean and covariance functions of $x_i(t)$ are

$$m_i(t) = x_i(0)e^{-\lambda_i t} + \alpha_i \int_0^t e^{-\lambda_i(t-\tau)} \mu_i(\tau) d\tau + \frac{b_i}{\lambda_i} (1 - e^{-\lambda_i t}) \quad (4)$$

$$C_i(t, t') = \frac{\sigma_{sys}^2}{2\lambda_i} (e^{-\lambda_i|t-t'|} - e^{-\lambda_i(t+t')}) \quad (5)$$

Besides the chosen on/off model, another assumption made by Jump3 is that the gene expression x_i is observed at the T time points with i.i.d. Gaussian noise:

$$\hat{x}_{i,k} = x_i(t_k) + \epsilon_{i,k}, \quad (6)$$

$$\epsilon_{i,k} \sim \mathcal{N}(0, \sigma_{obs,i,k}^2), k = 1, \dots, T, \quad (7)$$

where $\sigma_{obs, i, k}^2$ is the variance of the observation noise at time point t_k for gene g_i . Under this assumption, since both \mathbf{x}_i and ϵ_i are normally distributed, the observed expression levels $\hat{\mathbf{x}}_i$ also follow a multivariate normal distribution:

$$\hat{\mathbf{x}}_i \sim \mathcal{N}(\mathbf{m}_i, \mathbf{C}_i + \mathbf{D}_i), \quad (8)$$

where $\mathbf{D}_i \in \mathbb{R}^{T \times T}$ is a diagonal matrix with the values $\sigma_{obs, i, k}^2$ along the diagonal. One can therefore derive the log-likelihood of the observations:

$$\begin{aligned} \mathcal{L}_i = \log p(\hat{\mathbf{x}}_i) &= -\frac{T}{2} \log(2\pi) - \frac{1}{2} \log |\mathbf{C}_i + \mathbf{D}_i| \\ &\quad - \frac{1}{2} (\hat{\mathbf{x}}_i - \mathbf{m}_i)^\top (\mathbf{C}_i + \mathbf{D}_i)^{-1} (\hat{\mathbf{x}}_i - \mathbf{m}_i) \end{aligned} \quad (9)$$

2.2 The Nonparametric Part: Reconstructing the Network Topology

To summarize the previous section, a first key ingredient of Jump3 is to use the on/off model for modelling the expression of each gene g_i . This model allows, under the assumption that the expressions of g_i are observed with i.i.d. Gaussian noise, to compute a likelihood value \mathcal{L}_i for any given promoter state trajectory μ_i .

Now, the second key ingredient of Jump3 is to assume that the promoter state μ_i is a function of the expression levels of the genes that are direct regulators of g_i (see Fig. 1):

$$\mu_i(t) = f_i(\mathbf{x}_{reg, i}(t)) + \xi_t, \forall t, \quad (10)$$

where $\mathbf{x}_{reg, i}(t)$ is the vector containing the expression levels at time t of the regulators of g_i and ξ_t is a random noise with zero mean.

Within this context, the goal of Jump3 is, for each target gene g_i :

1. To identify the promoter state trajectory μ_i over the time interval $[t_1, t_T]$ that maximizes the likelihood \mathcal{L}_i ;

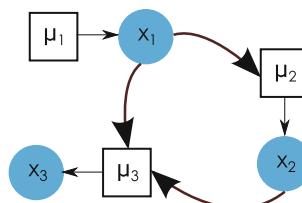


Fig. 1 Example of GRN. Circles represent the gene expressions, and squares represent the promoter states. Thick arrows model the promoter activations and show the network topology. Figure reproduced from [1], by permission of Oxford University Press

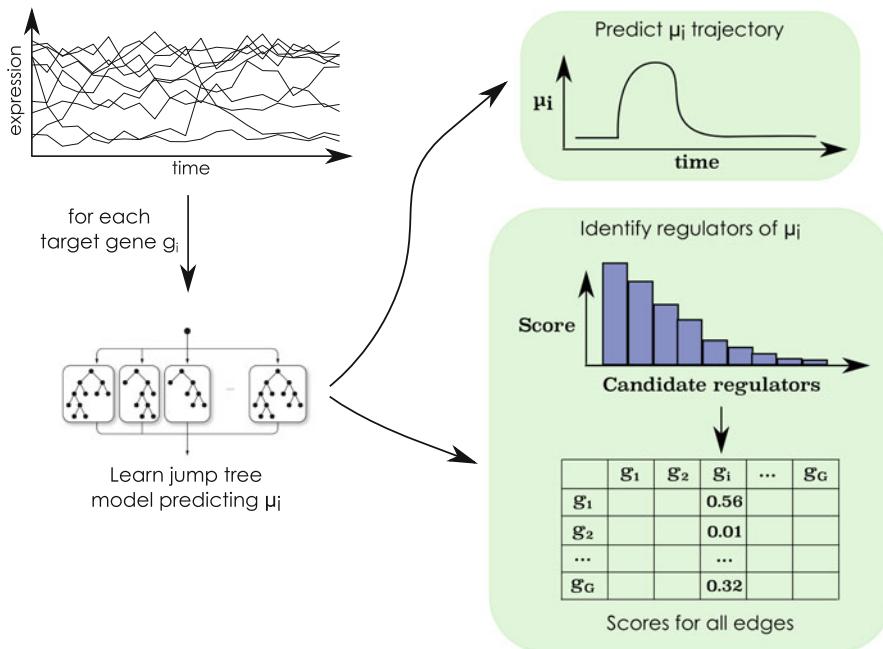


Fig. 2 The Jump3 framework. For each target gene g_i , $i = 1, \dots, G$, a function f_i in the form of an ensemble of jump trees is learned from the time series of expression data. The trajectory of the state of the promoter of g_i (μ_i) is predicted from the jump tree model and an importance score is computed for each candidate regulator. The score of a candidate regulator g_j is used as weight for the regulatory link directed from g_j to g_i . Figure reproduced from [1], by permission of Oxford University Press

2. To identify the regulators of g_i , i.e., the genes whose expression levels are predictive of the promoter state μ_i .

In principle, these goals could be achieved using a parametric model of promoter activation, like in [4]; this however necessarily makes strong assumptions on the form of the activation functions, and incurs significant computational overheads. Jump3 instead addresses both problems nonparametrically, by learning the functions f_i in the form of ensembles of decision trees (Fig. 2).

2.2.1 Decision Trees with a Latent Output Variable

A decision tree is a model that allows to predict the value of an *output* variable given the values of some *input* variables, using binary tests of the type " $x_j < c$," where x_j is one input variable and c is a threshold value (Fig. 3). Tree-based methods are currently one of the state-of-the-art approaches for GRN inference [5, 6]. They have several appealing properties, among which are their nonparametric nature (no assumption is made about the function f_i) and their scalability (they can be applied on high-dimensional datasets).

The idea of Jump3 is thus to learn for each target gene g_i a tree-based model that predicts the promoter state μ_i at any time point from the expression levels of the other genes (or a set of

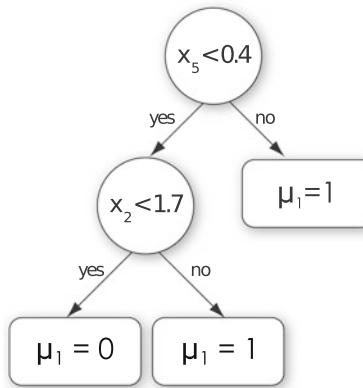


Fig. 3 Example of decision tree. A decision tree is a model that predicts the value of an output variable (here the state μ_1 of the promoter of gene g_1) from the values of some input variables (here the expression levels x_2 and x_5 of candidate regulators g_2 and g_5 , respectively). Each interior node of the tree is a test on the value of one input variable and each terminal node (or leaf) contains a prediction for the output

candidate regulators) at the same time point. Traditional tree-based algorithms can however only be used on datasets containing observed values for the input *and* output variables. In our case, the values of the input variables (i.e., the gene expression levels) are observed at multiple time points, but not the values of the output (i.e., the promoter state μ_i). We say that μ_i is a *latent* variable. Jump3 thus resorts to a new type of decision tree called *jump tree* that allows to predict the value of a latent output variable.

Algorithm 1 shows the pseudo-code for growing a jump tree predicting μ_i . The idea is to split the data samples (each corresponding to a different time point) into different subsets based on tests on the expression levels of the candidate regulators. For each test, the promoter state μ_i is assumed to be equal to 0 (resp., 1) at the time points where the test is true (resp., false). Each new split thus generates a new promoter state trajectory, corresponding to a certain likelihood value. A jump tree is constructed top-down using a greedy algorithm that iteratively replaces one leaf with a test node in a way to maximize the likelihood. More specifically, the jump tree is initialized as a single leaf, containing all the T observation time points. At each iteration, for each leaf N of the current jump tree and its corresponding set of time points P_N , the optimal split of P_N is identified, i.e., the test $s_N = "x_j < \text{value}"$ resulting in the promoter state trajectory that yields the maximum likelihood (lines 5–9 of Algorithm 1). The leaf N_* with the highest maximum likelihood is then replaced with a test node containing the optimal test s_{N_*} (lines 10–15). The child nodes of this new test node are two leaves containing, respectively, the subsets of

Algorithm 1: buildJumpTree

input : The expressions \hat{x}_i of a target gene g_i at T time points t_1, \dots, t_T , and the expressions \hat{X}_R of R candidate regulators at the same time points

output: A jump tree JT predicting the state μ_i of the promoter of g_i , from the expressions of the R candidate regulators

- 1 $JT \leftarrow$ a leaf containing the set of all the observation time points $\{t_1, \dots, t_T\}$
- 2 **foreach** t between t_1 and t_T **do** $\mu_i(t) \leftarrow 0$
- 3 $\mathcal{L}_i \leftarrow \text{computeLikelihood}(\mu_i, \hat{x}_i)$
- 4 **while** \mathcal{L}_i can be increased **do**
- 5 **for each leaf** N of JT **do**
- 6 $P_N \leftarrow \text{getTimePoints}(N)$
- 7 $(s_N, \mu_{i,N}) \leftarrow \text{pickOptimalSplit}(\hat{x}_i, \hat{X}_R, P_N, \mu_i)$
- 8 $\mathcal{L}_{i,N} \leftarrow \text{computeLikelihood}(\mu_{i,N}, \hat{x}_i)$
- 9 **end**
- 10 $N_* \leftarrow \arg \max_N \mathcal{L}_{i,N}$
- 11 **if** $\mathcal{L}_{i,N_*} > \mathcal{L}_i$ **then**
- 12 $\mu_i \leftarrow \mu_{i,N_*}$
- 13 $\mathcal{L}_i \leftarrow \mathcal{L}_{i,N_*}$
- 14 Split P_{N_*} into P_0 and P_1 according to s_{N_*}
- 15 $JT \leftarrow \text{createSplitNode}(N_*, s_{N_*}, P_0, P_1)$
- 16 **else**
- 17 **return** JT
- 18 Exit while loop
- 19 **end**
- 20 **end**

time points P_0 and P_1 , obtained from P_{N_*} using the test s_{N_*} (this procedure is illustrated in Fig. 4). The algorithm stops when the likelihood cannot be increased anymore.

Algorithm 2 shows how to choose a test at a new split node. For each possible test, composed of a candidate regulator g_j and a threshold value c (where c is set to the expression of g_j at one time point t_k), a candidate promoter state trajectory $\mu_{j,k}$ is obtained by setting $\mu_{j,k}$ to 0 (resp., 1) at the time points where the expression of g_j is lower (resp., higher) than c (lines 6–12). The test that is selected is then the one that yields the trajectory $\mu_{j,k}$ with the highest likelihood.

2.2.2 Ensemble of Decision Trees

A single decision tree typically *overfits* the observed data. This means that the tree model tends to also fit the noise that is contained in the data, leading to bad performance when predicting unseen data. One solution to reduce overfitting is to grow an ensemble of several trees and to average the predictions of these different trees. A way to obtain different trees is to introduce some randomization when learning each of the trees. In Jump3, the *Extra-Trees* procedure is used [7]: at each test node, the best

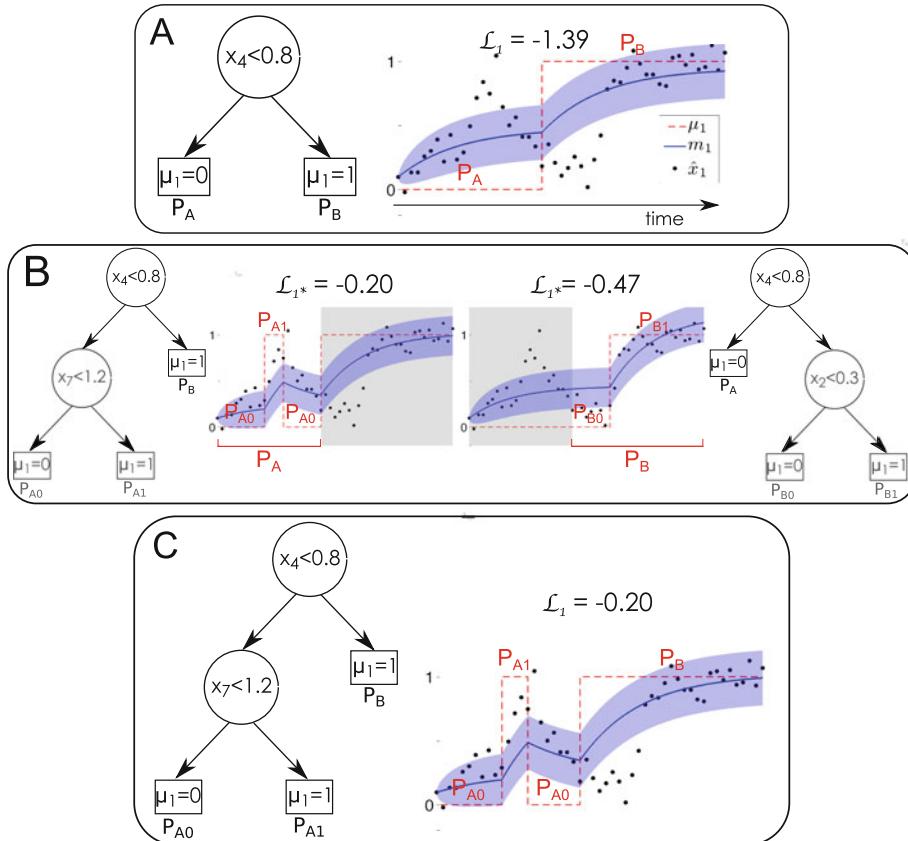


Fig. 4 Growing a jump tree predicting the state μ_1 of the promoter of gene g_1 . **(a)** Each iteration of the jump tree algorithm results in a new tree and a new trajectory μ_1 (dashed red line) yielding a likelihood \mathcal{L}_1 . In this example, the current tree splits the set of observation time points into two subsets P_A and P_B , each one corresponding to a leaf of the tree. The plot also shows the posterior mean \mathbf{m}_1 of the expression of g_1 (solid blue line), with confidence intervals (shaded area), and the observed expression levels of g_1 (black dots). **(b)** For each leaf of the current tree, the optimal split of the corresponding set of time points is identified. **(c)** The leaf for which the optimal split yields the highest likelihood is replaced with a test node. Figure reproduced from [1], by permission of Oxford University Press

test is selected among a number K of random tests, rather than all the possible tests. Each of the K candidate tests is obtained by randomly selecting one candidate regulator (without replacement) and one threshold value. Each tree of the ensemble will thus have its own prediction of $\mu_i(t)$. This prediction is then averaged over the different trees, yielding a probability for the promoter state to be active at time t .

2.2.3 Importance Measure

Once learned, a single tree or an ensemble of trees can be used to compute a score for each candidate regulator that measures the importance of that candidate regulator in the tree-based model. The idea is then to use the importance $w_{j,i}$ of candidate regulator g_j in the model predicting $\mu_i(t)$ as weight for the edge of the GRN that is directed from gene g_j to gene g_i .

Algorithm 2: pickOptimalSplit

input : The expressions \hat{x}_i of target gene g_i at T time points t_1, \dots, t_T ,
 the expressions \hat{X}_R of R candidate regulators at the same time
 points, a subset of time points $P \subset \{t_1, \dots, t_T\}$, and an initial
 promoter state trajectory μ_{init}

output: A test in the form “ $x_j < value$ ” and an updated promoter state
 trajectory

```

1 for each candidate regulator  $g_j \in R$  do
2   for each time point  $t_k \in P$  do
3      $s_{j,k} \leftarrow "x_j < \hat{x}_j(t_k)"$            // definition of a new test
4      $\mu_{j,k} \leftarrow \mu_{init}$ 
5     for each time point  $t_{k'} \in P$  do
6       if  $\hat{x}_j(t_{k'}) < \hat{x}_j(t_k)$  then
7          $\mu_{j,k}(t_{k'}) \leftarrow 0$ 
8       else
9          $\mu_{j,k}(t_{k'}) \leftarrow 1$ 
10      end
11      foreach  $t$  between  $t_{k'}$  and the next observation time point do
12         $\mu_{j,k}(t) \leftarrow \mu_{j,k}(t_{k'})$ 
13      end
14    end
15  end
16  $(j^*, k^*) = \arg \max_{(j,k)} \mathcal{L}_{j,k}$ 
17 return the test  $s_{j^*,k^*}$  and the trajectory  $\mu_{j^*,k^*}$ 

```

The importance measure that is used in Jump3 is based on the likelihood gain at each test node N , obtained during the tree learning:

$$I(N) = \mathcal{L}_{i,N} - \mathcal{L}_i, \quad (11)$$

where \mathcal{L}_i and $\mathcal{L}_{i,N}$ are the log-likelihoods, respectively, obtained before and after splitting the data samples at node N . For a single tree, the importance score $w_{j,i}$ of candidate regulator g_j is then computed by summing the likelihood gains at all the tree nodes where there is a test on that regulator:

$$w_{j,i} = \sum_{k=1}^n I(N_k) \mathbb{1}_{\mathcal{N}_k}(g_j), \quad (12)$$

where n is the number of test nodes in the tree and N_k is the k -th test node. $\mathbb{1}_{\mathcal{N}_k}(g_j)$ is a function that is equal to one if the test at node N_k uses g_j , and zero otherwise. The candidate regulators that are not used in any test node thus obtain an importance score of zero and those ones that appear in tests close to the root node of the tree typically obtain high scores. When using an ensemble of trees, the importance measure of each candidate regulator is simply averaged over all the different trees.

2.2.4 Regulatory Link Ranking

Given the definition of the importance score 12, the sum of the scores of all the candidate regulators, for a single tree, is equal to the total likelihood gain yielded by the tree:

$$\sum_{j \neq i} w_{j,i} = \mathcal{L}_i - \mathcal{L}_{i,0}, \quad (13)$$

where $\mathcal{L}_{i,0}$ is the initial log-likelihood obtained with $\mu_i(t) = 0, \forall t$, and \mathcal{L}_i is the final log-likelihood obtained after the tree has been grown. Some regulatory links of the network may therefore receive a high weight only because they are directed towards a gene for which the overall likelihood gain is high. To avoid this bias, the importance scores obtained from each tree are normalized, so that they sum up to one:

$$w_{j,i} \leftarrow \frac{w_{j,i}}{\mathcal{L}_i - \mathcal{L}_{i,0}} \quad (14)$$

2.2.5 Kinetic Parameters

The on/off model 3 has three kinetic parameters a_i , b_i , and λ_i . Given a promoter state trajectory μ_i , the values of a_i and b_i are optimized in order to maximize the log-likelihood \mathcal{L}_i (in line 8 of Algorithm 1, and once the final μ_i trajectory is obtained from the tree ensemble). Since \mathcal{L}_i is a quadratic function of a_i and b_i (see Eqs. (4) and (9)), these two parameters can be easily optimized using quadratic programming. The value of the decay rate λ_i is more difficult to optimize and is instead directly estimated from the observed expressions of gene g_i , by assuming an exponential decay $e^{-\lambda_i t}$ between the highest and lowest expressions of g_i .

2.3 Computational Complexity

Let us assume for simplicity that each tree contains S test nodes. It can then be shown that the computational complexity of Jump3 is $O(Gn_{trees}KS^2T^2)$, where G is the number of genes, n_{trees} is the number of trees per ensemble, K is the number of randomly chosen candidate regulators at each test node, and T is the number of observation time points. At worst, the complexity of the algorithm is thus quadratic with respect to the number of genes (when $K = G - 1$) and $O(T^4)$ with respect to the number of observations (when $S = T - 1$, i.e., each tree is fully developed with each leaf corresponding to one time point). Jump3 can thus become computationally intensive when the number of data samples is too high, but remains scalable with respect to the network size.

Table 1 gives an idea of the computing times, when using the MATLAB implementation of Jump3. In each case, K was set to the number of candidate regulators and n_{trees} was set to 100. These computing times were measured on an 8GB RAM, 1.7 GHz Intel core i7 computer.

Table 1
Running times of Jump3 for varying dataset sizes

G	n_{TF}	T	Running time
10	10	105	3 min
100	100	210	48 h
1000	40	25	4 h

G: number of genes, n_{TF} : number of candidate regulators, T: number of observations

2.4 Jump3 Performance

The ability of Jump3 to accurately reconstruct network topologies was assessed using several *in silico* networks (among which are the networks of the *DREAM4 In Silico Network challenge* [6, 8]) as well as one synthetic network (the IRMA network [9]), for which the true topologies are known. For these networks, Jump3 always yields competitive, and often better, performance, compared to other existing state-of-the-art network inference methods.

Jump3 was also applied to a real gene expression dataset related to murine macrophages [10], in order to retrieve regulatory interactions that are involved in the immune response. The predicted network (composed of the 500 top-ranked interactions) is highly modular, with a few transcription factors acting as hub nodes, each one regulating a large number of target genes. These hub transcription factors were found to be biologically relevant, comprising some interferon genes, one gene known to be associated with cytomegalovirus infection and several cancer-associated genes.

The interested reader can find additional details and all the results of these experiments in the original paper [1].

3 The Jump3 Code

Jump3 was implemented and tested in MATLAB. A fully documented implementation of Jump3 is available at

<https://github.com/vahuynh/Jump3>.

In this section, we provide a tutorial showing how to run the code using the synthetic example data provided with the implementation; to run it on a different data set, one simply needs to provide a new data file.

3.1 Run Jump3

First of all, add the directory “jump3_code” into MATLAB’s path:

```
path(path, 'path_to_the_directory/jump3_code')
```

A MATLAB file “exampleData.mat” is provided for this tutorial.

```
load exampleData.mat
who
```

```
## Your variables are:
##
## data      genes      obsTimes
```

- `data` contains the (simulated) expression data of 10 genes from 5 time series. This is a toy dataset that was obtained by simulating the on/off model in Eq. (3). Noisy observations at several time points were obtained from the simulated expression time series by adding i.i.d. Gaussian noise.
- `obsTimes` contains the observation time points for each time series of `data`.
- `genes` contains the names of the genes.

3.1.1 Noise Parameters

Jump3 has two noise parameters. These parameters must be put in a structure array (that we will name here `noiseVar`) with two fields that *must* be named `sysNoise` and `obsNoise`.

- `noiseVar.sysNoise` is the variance of the intrinsic noise (σ_{sys}^2); one single value.
- `noiseVar.obsNoise` is the variance of the observation noise ($\sigma_{obs,i,k}^2$); one value for each data value.

The optimal values of the noise parameters are difficult to identify (in most cases, trying to optimize these parameters would result in local optima). Some heuristic should thus be used to set the values of these parameters. For example, a dynamic noise can be used for the observation noise, assuming that the observation noise is higher for higher expression values:

```
noiseVar.obsNoise = cell(1,length(data));
for k=1:length(data)
    % A dynamic noise is used for the observation
    % noise
    noiseVar.obsNoise{k} = (data{k}/10).^2;
    % Replace zero values with a small number
    % to avoid numerical errors
    noiseVar.obsNoise{k}(noiseVar.obsNoise{k}==0)
        = 1e-6;
end
```

The intrinsic noise can be assumed to be much smaller than the observation noise:

```
noiseVar.sysNoise = 1e-4;
```

3.1.2 Run Jump3 with Its Default Parameters

```
[w,exprMean,exprVar,promState,kinParams,kinParamsVar,
  trees] = ...
jump3(data,obsTimes,noisVar);
```

(This should take a few minutes.)

By default, all the genes are candidate regulators, the parameter K of the Extra-Trees (i.e., the number of randomly chosen candidate regulators at each test node) is set to the total number of candidate regulators and the number n_{trees} of trees per ensemble is set to 100.

The algorithm outputs the following:

- **w**: weights of the regulatory links. $w(i, j)$ is the weight of the link directed from gene \mathcal{g}_j to gene \mathcal{g}_i .
- **exprMean**: posterior mean $m_i(t)$ of the expression of each gene \mathcal{g}_i in each time series.
- **exprVar**: posterior variance $c_i(t, t)$ of the expression of each gene \mathcal{g}_i .
- **promState**: posterior state $\mu_i(t)$ of the promoter of each gene \mathcal{g}_i in each time series.
- **kinParams**: (optimized) values of the kinetic parameters a_i , b_i , and λ_i for each gene \mathcal{g}_i .
- **kinParamsVar**: variances of the kinetic parameters a_i and b_i .
- **trees**: ensemble of jump trees predicting the promoter state of each gene \mathcal{g}_i .

3.1.3 Restrict the Candidate Regulators to a Subset of Genes

To guide the network inference, the candidate regulators can be restricted to a subset of genes (for example, the genes that are known to be transcription factors).

```
% Indices of the genes that are used as candidate
  regulators
tfidx = [2 5 6 8 9];
w2 = jump3(data,obsTimes,noisVar,tfidx);
```

In **w2**, the links that are directed from genes that are not candidate regulators have a score equal to 0.

3.1.4 Change the Settings of the Extra-Trees

It is possible to set the values of the two parameters of the Extra-Trees algorithm (K and n_{trees}).

```
% Extra-Trees parameters
K = 3;
ntrees = 50;

% Run the method with these settings
w3 = jump3(data,obsTimes,noisVar,1:10,K,ntrees);
```

3.1.5 Obtain More Information

Additional information about the function `jump3()` can be found by typing this command:

```
help jump3
```

3.2 Write the Predictions

3.2.1 Get the Predicted Ranking of All the Regulatory Links

Given the weight matrix `w` returned by the function `jump3()`, the ranking of regulatory links (where the links are ordered by decreasing order of weights) can be retrieved with the function `getLinkList()`.

```
getLinkList(w)
```

The output will look like this:

```
## G8 G6 0.886894
## G3 G7 0.841794
## G9 G10 0.831725
## G1 G5 0.782503
## G1 G2 0.715546
## G7 G3 0.512323
## ...
```

Each line corresponds to a regulatory link. The first column shows the regulator, the second column shows the target gene, and the last column indicates the weight of the link.

If the gene names are not provided, the i -th gene is named “ G_i .”

Note that the ranking that is obtained will be slightly different from one run to another. This is due to the intrinsic randomness of the Extra-Trees. The variance of the ranking can be decreased by increasing the number of trees per ensemble.

3.2.2 Show Only the Links that are Directed from the Candidate Regulators

The set of candidate regulators can be specified, to avoid displaying the links that are directed from genes that are not candidate regulators and have hence a score of zero.

```
getLinkList(w2, tfidx)
```

3.2.3 Show the Top Links Only

The user can specify the number of top-ranked links to be displayed. For example, to show only the first five links:

```
ntop = 5;
getLinkList(w2, tfidx, {}, ntop)
```

3.2.4 Show the Names of the Genes

The ranking can be shown with the actual gene names, rather than the default names:

```
getLinkList(w2, tfidx, genes, 5)
```

```
## FAM47B RXRA 0.983707
## ZNF618 RFC2 0.839112
## GPX4 CYB5R4 0.808789
## CYB5R4 NPY2R 0.674845
## CYB5R4 GPX4 0.673964
```

3.2.5 Write the Predicted Links in a File

Instead of displaying the edge ranking in the MATLAB console, the user can choose to write it in a file (named here “ranking.txt”):

```
getLinkList(w, 1:10, genes, 0, 'ranking.txt')
```

3.2.6 Obtain More Information

Additional information about the function `getLinkList()` can be found by typing this command:

```
help getLinkList
```

3.3 Plot the Modelling Results

The posterior mean expression of one gene in one time series, as well as the predicted promoter state trajectory, can be plotted with the function `plotPosteriors()`.

```
% Plot the results for the fifth gene in the first
time series
Tsidx = 1;
geneidx = 5;
plotPosteriors(data, obsTimes, exprMean, exprVar,
    promState, ...
    Tsidx, geneidx)
```

Figure 5 shows an example of plot returned by this command.

3.3.1 Obtain More Information

Additional information about the function `plotPosteriors()` can be found by typing this command:

```
help plotPosteriors
```

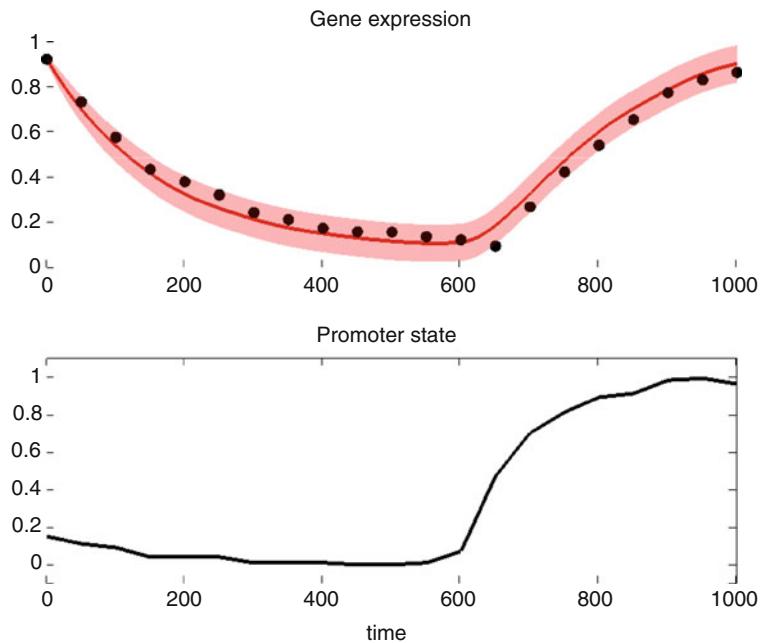


Fig. 5 The top plot shows the posterior mean expression of the target gene (solid red line) with the confidence intervals (shaded red area) as well as the observed expression of the gene (black dots). The bottom plot shows the predicted state of the promoter of the gene

4 Conclusion

Reconstructing GRNs from time series data is a central task in systems biology, and several methods are currently available. In this chapter, we reviewed a recent method, Jump3, which combines some of the advantages of model-based methods with the speed and flexibility of nonparametric, tree-based learning methods. The focus of this chapter is twofold: we first provide a brief introduction to the mathematical aspects of the Jump3 framework, focusing on the practical aspects of extracting relevant statistics from the algorithm's output. We then provide a step-by-step tutorial on how to use the MATLAB implementation of Jump3, freely available at <https://github.com/vahuynh/Jump3>. We hope that this chapter will enhance the usability of Jump3, reinforcing its status as a useful tool within the GRN inference toolkit.

Acknowledgements

VAHT is a Postdoctoral Fellow of the F.R.S.-FNRS. GS acknowledges support from the European Research Council under grant MLCS 306999.

References

1. Huynh-Thu VA, Sanguinetti G (2015) Combining tree-based and dynamical systems for the inference of gene regulatory networks. *Bioinformatics* 31(10):1614–1622
2. Ptashne M, Gann A (2002) Genes and signals. Cold Harbor Spring Laboratory Press, New York
3. Gardiner CW (1996) Handbook of stochastic methods. Springer, Berlin
4. Ocone A, Millar AJ, Sanguinetti G (2013) Hybrid regulatory models: a statistically tractable approach to model regulatory network dynamics. *Bioinformatics* 29(7):910–916
5. Huynh-Thu VA, Irrthum A, Wehenkel L, Geurts P (2010) Inferring regulatory networks from expression data using tree-based methods. *PLoS One* 5(9):e12776
6. Marbach D, Costello JC, Küffner R, Vega N, Prill RJ, Camacho DM, Allison KR, the DREAM5 Consortium, Kellis M, Collins JJ, Stolovitzky G (2012) Wisdom of crowds for robust gene network inference. *Nat Methods* 9(8):796–804
7. Geurts P, Ernst D, Wehenkel L (2006) Extremely randomized trees. *Mach Learn* 36(1):3–42
8. Prill RJ, Marbach D, Saez-Rodriguez J, Sorger PK, Alexopoulos LG, Xue X, Clarke ND, Altan-Bonnet G, Stolovitzky G (2010) Towards a rigorous assessment of systems biology models: the DREAM3 challenges. *PLoS One* 5(2):e9202
9. Cantone I, Marucci L, Iorio F, Ricci MA, Belcastro V, Bansal M, Santini S, di Bernardo M, di Bernardo D, Cosma MP (2009) A yeast synthetic network for in vivo assessment of reverse-engineering and modeling approaches. *Cell* 137(1):172–181
10. Blanc M, Hsieh WY, Robertson KA, Watterson S, Shui G, Lacaze P, Khondoker M, Dickinson P, Sing G, Rodríguez-Martín S, Phelan P, Forster T, Strobl B, Müller M, Riemersma R, Osborne T, Wenk MR, Angulo A, Ghazal P (2011) Host defense against viral infection involves interferon mediated down-regulation of sterol biosynthesis. *PLoS Biol* 9(3):e1000598



Chapter 10

Network Inference from Single-Cell Transcriptomic Data

Helena Todorov, Robrecht Cannoodt, Wouter Saelens, and Yvan Saeyns

Abstract

Recent technological breakthroughs in single-cell RNA sequencing are revolutionizing modern experimental design in biology. The increasing size of the single-cell expression data from which networks can be inferred allows identifying more complex, non-linear dependencies between genes. Moreover, the inter-cellular variability that is observed in single-cell expression data can be used to infer not only one global network representing all the cells, but also numerous regulatory networks that are more specific to certain conditions. By experimentally perturbing certain genes, the deconvolution of the true contribution of these genes can also be greatly facilitated. In this chapter, we will therefore tackle the advantages of single-cell transcriptomic data and show how new methods exploit this novel data type to enhance the inference of gene regulatory networks.

Key words Network inference, Single cell, Gene regulatory networks, Transcriptomics

1 Introduction

Recent technological breakthroughs in single-cell RNA sequencing (scRNA-seq) are revolutionizing modern experimental design in biology. These breakthroughs lie at the basis of myriads of biological discoveries, the most common of which are the identification of novel cell types and the reconstruction of dynamic processes. In the context of network inference (NI), scRNA-seq has several major advantages over more traditional bulk transcriptional profiling techniques such as microarrays and bulk RNA sequencing.

Traditionally, regulatory interactions are inferred from bulk transcriptional profiles, generated by pooling together the RNA transcripts of a supposedly homogeneous population of several thousands of cells, and quantifying the transcript abundance through a microarray or RNA sequencing. Incorrect assumptions on the homogeneity of the pooled cells may lead to the masking of relevant expression patterns in rare cell populations, as expression values are averaged over the whole population (Fig. 1). In

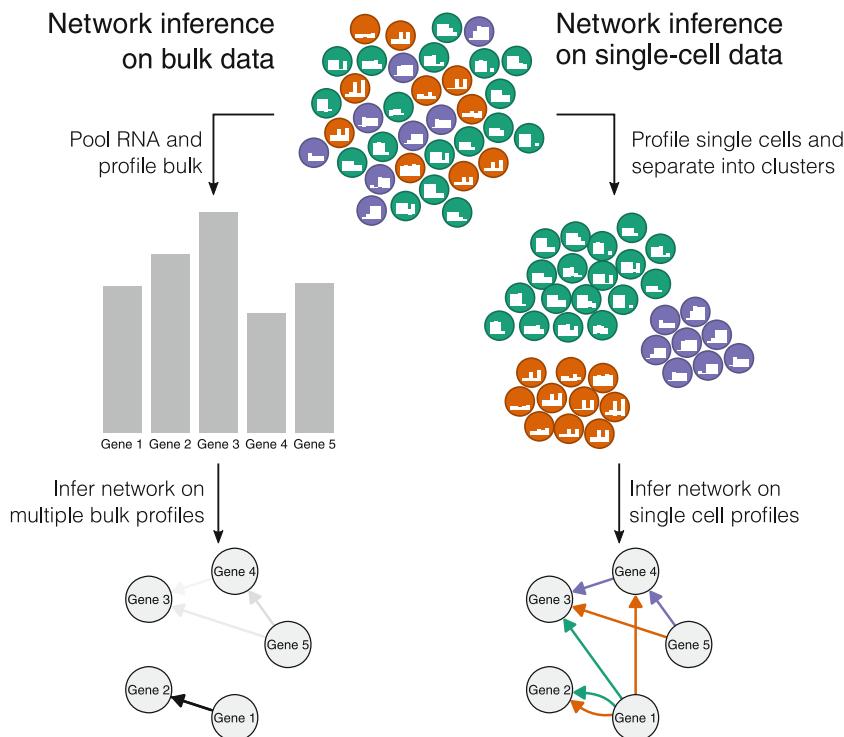


Fig. 1 Bulk expression data return the average expressions of genes among large numbers of cells. In order to infer regulatory networks from this type of data, multiple bulk profiles (resulting from time-series or perturbation experiments) are typically used. On the other hand, sequencing the transcriptome at the single-cell level uncovers the high variability among cells, providing the necessary information to directly infer gene regulatory networks

addition, NI methods rely on a diverse set of time-series and perturbation experiments in order to reliably identify causal regulatory interactions. However, such experiments are expensive and time-consuming, and an inaccurate choice of time points might result in crucial intermediate stages being missed.

One of the main advantages of single-cell transcriptomics is the ability to quantify the exact cellular state of thousands of cells per experiment. The inter-cellular heterogeneity caused by naturally occurring biological stochasticity [1] can be exploited to infer regulatory interactions between transcription factors (TFs) and their target genes (*see* Fig. 1). In this sense, heterogeneity in the cell population will ease the inference of networks, rather than mask condition-specific expression patterns and regulatory interactions.

While single-cell transcriptomics offers many advantages over traditional bulk profiling methods, several computational challenges pertaining to the preprocessing of the data have a big impact on single-cell NI [2]. In this chapter, we will therefore firstly focus on zero-inflation, confounding factors, and scalability problems (Subheading 2). We will then discuss several recent developments in single-cell transcriptomics analysis that present a

high interest to further improve NI methods. In the second part, we will focus on novel unsupervised learning methods that have been proposed for inferring the different cellular states within a heterogeneous cell population. These methods can help to increase the accuracy of NI by deriving differential, dynamic, or profile-specific regulatory networks (Subheading 3). Lastly, single-cell transcriptomics has opened up a gateway to performing high-throughput multi-omics and/or perturbation experiments, which could again revolutionize how gene regulatory networks are being inferred at a high-throughput scale (Subheading 4).

2 Ongoing Computational Challenges

One of the most apparent differences between single-cell and bulk transcriptomics is that for each cell only a few thousands of genes are detected. This zero-inflation can arise through lowly expressed transcripts not being captured during library preparation (referred to as “dropouts”) [3], transcriptional bursting [4, 5], or the gene simply not being expressed in the cell. The problem of dropouts is especially pronounced when using technologies where only a very low number of genes are captured [6]. The shift in distribution caused by zero-inflation can vastly decrease the performance of traditional NI methods, as many zero-inflated genes become highly correlated with each other. There are several possible solutions to the problem of dropouts, although the efficacy of these techniques has not yet been thoroughly explored in the context of single-cell NI. The expression of dropouts events can be imputed using information from highly similar cells in the population [7, 8]. Another approach is to take into account dropouts into the model, for example, using appropriate zero-inflated distributions [9–11] or spike-ins, to estimate technical variance, although these come with their own challenges [12]. However, more complex models add more parameters to the model, requiring a large dataset for parameter estimation and further escalating the scalability problems of single-cell NI as discussed later. Finally, networks can also be inferred on clusters of cells and/or genes, circumventing the dropouts by again investigating the network on respectively population level and gene module level [13].

Another issue is that rapid successive improvements in single-cell profiling technologies have caused an exponential scaling in the number of cells being profiled over the past decade [14]. Traditional NI already requires clever heuristics to predict the regulatory effect between each pair of genes, where the number of samples is yet frequently limited to relatively few samples. In comparison, contemporary single-cell datasets are already reaching sizes of up to hundreds of thousands of single cells. Several approaches simplify the inferred model, for example, by discretizing either the input expression values [15] or outputted regulatory interactions [16] as

binary on/off-state values. Another solution would be to simplify the outputted network by clustering similarly expressed target genes into modules, and clustering cells with similar expression profiles into populations [13]. Finally, a more obvious solution to the scaling problem will be the transition to big data solutions, as shown in a recent example with GRNBoost [17].

3 Integrating Unsupervised Learning and Network Inference

The increasing size of the single-cell expression data from which networks can be inferred allows identifying more complex, non-linear dependencies between genes. In this type of data, the regulatory processes can be inferred directly and accurately from thousands of variable cells. This advantage has been used in methods relying on partial information decomposition [18] or Bayesian NI [19] to infer networks from single-cell transcriptomic data.

However, the processes involved in living organisms are highly dynamic, and it has been long known that regulatory interactions are context-dependent as a result [20]. Consequently, attempting to create an accurate model of those processes by inferring a static regulatory network may have limited relevance. While the presence of context-dependent regulatory interactions was an issue in NI on bulk transcriptomics data, the heterogeneity within single-cell transcriptomics increases the relevance of this issue even further. In addition, strong context-specific interactions are easily masked and thus not detectable by variations in expression when the interaction is not active.

This can be solved with NI methods that take into account the dynamic aspect of the regulome and are able to produce network models with variable regulatory activity. This should improve the detection rate of variable interactions, and allow researchers to explore for which conditions certain interactions are specific. To this end, several approaches have been proposed, which can be broadly classified in three different classes, depending on the output structure they produce: differential, dynamic, and profile-specific networks (Fig. 2). An overview of the existing methods for single-cell network inference is shown in Table 1.

With each of these methodologies, it should be noted that while they produce networks that are specific to certain subsets of the cells' profiles, they still use the information from all available profiles. If a method is to infer a network from cells in only a certain condition, it will infer interactions from noise in the data, rather than the changes that separate that condition from any other. As such, a context-dependent network inferred from only a subset of the profiles is likely to be less accurate than a static network trained on all profiles.

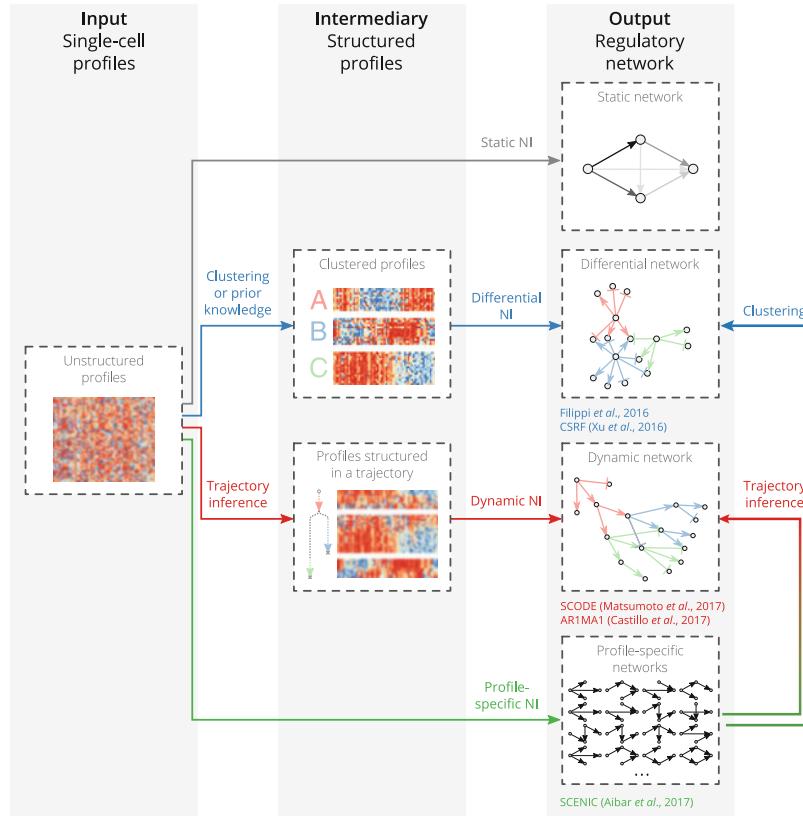


Fig. 2 As regulatory interactions can be context-specific, static NI cannot always accurately model such interactions. Several approaches have been proposed to model context-specific interactions, namely differential, dynamic, and profile-specific NI. Differential NI methods aim to derive different cellular states among the profiles, and infer networks for each of the states. Dynamic NI methods use trajectory inference to create a continuous state graph, and infer the activity of regulatory interactions across the pseudotime in the state graph. Profile-specific networks are the most complex model, where the activity of regulatory interactions is inferred on a per-profile basis. Profile-specific networks can be interpreted individually, or they can be transformed by clustering or trajectory inference to derive differential or dynamic networks, respectively

3.1 Differential Network Inference

Also called case-specific or condition-specific NI, differential NI methods aim to reconstruct one network for each of the given conditions among the transcriptomic profiles. The conditions can be different cellular states or changes in environment, and profiles can be grouped according to prior knowledge or derived through unsupervised clustering. From the resulting networks, one can then investigate the pathways that are differentially activated between conditions (e.g., deregulated pathways between a diseased and healthy condition), or those that are similarly activated between conditions (e.g., similarly activated pathways between two different disease conditions).

Differential NI methods have already been described for bulk -omics data [26], where they have been used, for instance, to

Table 1
Existing tools for single-cell network inference

Method	GRN type	Methodology
PIDC [18]	Global	Partial information decomposition
[19]	Global/differential	Bayesian nonparametric procedure
CSRF [21]	Differential	Random forests
SCNS [15]	Global	Boolean network models, validated with cell ordering
[22]	Global	ODE models, calibrated with cell ordering
[23]	Global	Random forests and ODE
AR1MA1 [24]	Dynamic	First-order autoregressive moving average model
SCODE [25]	Dynamic	ODE
SCENIC [17]	Profile-specific	Random forests and motif analysis

elucidate deregulated mechanisms in different subtypes of leukemia [27]. For single-cell transcriptomics, two pioneering differential NI methods have been proposed. The first method, relying on the random forests algorithm, is case-specific random forests (CSRF) [21]. Random forests have been widely used to assess regulatory networks from bulk expression data. They decompose the construction of the network of N genes into N prediction problems that are addressed with numerous decision trees. In the case of CSRFs, the cells are given specific weights such that similar cells have higher probabilities to be used together in each decision tree. The resulting inferred regulatory networks are thus specific to certain homogeneous groups of cells.

The second method relies on Bayesian Pólya trees [19]. The posterior probabilities of dependence and independence between two genes are computed using Pólya tree priors, to model the unknown distribution of the data. This probabilistic method, for example, helps to identify sets of genes whose expression is dependent under a healthy condition and becomes independent in samples corresponding to a disease state. It has been applied in order to identify changes in gene expression in response to breast cancer [19].

3.2 Dynamic Network Inference

Advances in single-cell transcriptomics have fostered the development of trajectory inference methods [28], which are computational tools to reconstruct dynamic processes such as cellular differentiation and cell cycle (Fig. 3). These methods usually recover the underlying structure of the data by focusing on the similarities between cells. The cells can then be ordered along a trajectory that optimally models the data structure. This bears resemblance to the more traditional time-series experiments with

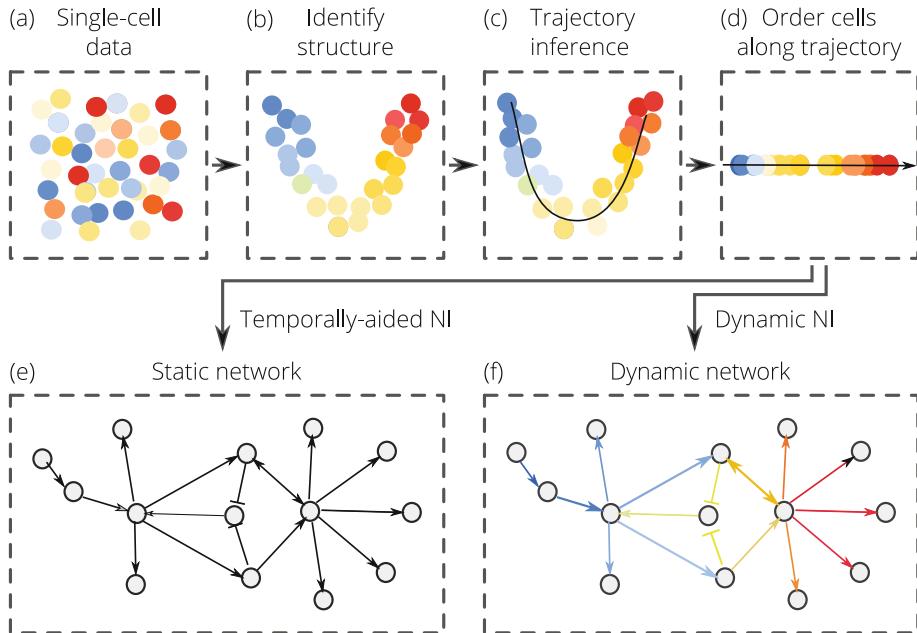


Fig. 3 A common approach to integrating trajectory inference with NI. (a) Unsupervised NI methods start from unstructured single-cell transcriptomics data. (b) The underlying structure of single-cell expression data can be recovered either by reducing the dimensions of the data or by generating a graph. Both these representations will highlight the similarities between the cells in the high-dimensional space. (c) A trajectory can then be inferred from this representation of the data. (d) The cells can be re-ordered along the recovered trajectory. (e) Once the cells have been ordered along a pseudotime axis, static networks can be inferred by taking into account the temporality. For example, when two genes have correlated expressions but one is expressed at slightly earlier stages, then the first gene likely regulates the second. (f) Ordered data can also help to infer dynamic networks, in which the links between genes are specific to certain pseudotime stages

bulk transcriptomics data, albeit at a much higher temporal resolution. Unlike time-series experiments, trajectories are ideally inferred in an unsupervised manner, as not to introduce human bias.

As in time-series experiments, NI methods can exploit the information provided by trajectory inference methods in order to improve the resulting regulatory network. Several NI methods use this information in order to determine the directionality and the type of an interaction. This has helped to construct more accurate boolean networks (SCNS [15]), by selecting the network that most reliably describes the observed cell ordering. Several NI methods use this cell-ordering information to construct ordinary differential equations that are again improved to optimally describe the observed data [22, 23]. While such methods are generally able to improve the accuracy of the produced networks by using information from the trajectory, they suffer from the same limitations as discussed at the start of the section since they produce a single regulatory network that is supposed to be a model for all of the cells in the population.

Instead, dynamic NI methods produce a network of interactions with variable activation levels across the trajectory. They provide useful information on which transcription factors are expressed at the beginning of a developmental process, and which gene interactions occur at later stages, for instance. Dynamic networks are more complex models in comparison to differential and static networks.

Two pioneering methods infer dynamic networks from ordered cells. ARIMA1 [24] relies on a pseudotemporal ordering of the cells to infer the expression of a gene at a time t as the result of the weighted expressions of its regulators at time $t-1$. The potential regulators of a gene are seen as hidden variables with binary expressions (a regulator is either “on” or “off”). The weights of the interactions between a regulator and its target gene are parameters that are optimized until convergence in the Bayesian process. The ARIMA1 method thus returns weighted interactions between genes along the pseudotime. The size of the resulting regulatory network can be trimmed by selecting links between genes that have the highest weights.

The second method, SCODE [25], aims to describe the transcription factors’ expression dynamics along time with a set of ordinary differential equations. As solving these equations requires a lot of computing time and a large amount of memory for large numbers of genes, the dimensions of the data are reduced into z factors (where z is much lower than the number of genes). In these reduced dimensions, the interactions between the z factors can be inferred more rapidly, and then transposed to the original dimensions of the data.

Both methods have been applied to scRNA-seq datasets and sometimes showed better accuracy when inferring regulatory interactions than NI methods that return static networks. While these two methods only support linear trajectories, future methods for inferring dynamic NI methods will likely also support other trajectory structures, such as branching or cyclical.

3.3

Profile-Specific Network Inference

The most complex network models are the profile-specific networks, in which one set of active regulatory interactions is predicted per profile, or cell. Profile-specific networks can be seen as differential networks with one specific network being inferred for each unique cellular state.

Profile-specific NI has its roots in NI on bulk data. In this context it is often referred to as patient-specific or sample-specific NI [16, 29]. It could be used, for example, to investigate deregulated pathways for individual patients in an unbiased approach. For single-cell data, the term “profile-specific” was chosen here to avoid confusion between cells and samples, as the data contains one profile per cell but these likely originate from one sample.

While profile-specific networks can seem daunting to interpret, they can also be interpreted as *regulomics* data, and many of the techniques used for analyzing transcriptomics data (e.g., clustering or visualization) can be exploited in the same way. A differential network can be obtained from profile-specific networks by clustering, and a dynamic network can be obtained by performing NI.

The difference between this approach and more “direct” differential and dynamic NI methods is that the clustering or trajectories were derived from the *regulomes* of the samples, and not directly from transcriptomics data.

A likely advantage could be that NI methods are more robust to batch effects in comparison to clustering and trajectory inference methods. Therefore, by first inferring profile-specific networks, the downstream aggregation could produce more accurate networks. Another advantage could be that the profile-specific networks could be aggregated to non-exclusive biclusters. Each bicluster represents a set of samples for which a set of interactions are similarly active. Such biclusters would allow the unbiased discovery of a set of regulatory interactions important in a subset of all cells, which could be useful for things like disease subtype identification and drug discovery [30].

A method similar to profile-specific NI combined with biclustering is SCENIC [17]. SCENIC first uses GENIE3 [31] to infer a static network, followed by motif discovery to group together target genes into groups called regulons. In a later step, the activity of a regulon is determined for each individual profile by calculating the enrichment of that regulon for the profiles’ expression values. Using motif discovery can aid in significantly improving the accuracy of the network, especially since motif data is now available for almost every transcription factor. However, motifs can be very similar between transcription factor family members and can in some cases be very degenerated. Furthermore, the binding of a transcription factor to DNA requires more than just the presence of a motif, and the presence of complex protein regulatory structures should be investigated to identify robust regulatory effects.

4 Single-Cell Network Inference Using Perturbational Data

NI on bulk transcriptional data benefits greatly from the inclusion of perturbational experiments [32], where one or several regulators have been knocked out or perturbed. This vastly eases the deconvolution of the true contribution of a regulator towards different targets, as perturbed regulators will necessarily be upstream from differentially expressed targets. Techniques in the past were based either on large genetic screens, which have a high cost and require

large numbers of cells, or phenotypic screening, which need a selection criterion and do only observe a limited phenotype, such as cell survival or marker expression.

Several recently developed technologies [33–36] drastically increase the throughput by multiplexing multiple CRISPR/Cas perturbations in one single-cell experiment. This is achieved by generating a library of CRISPR/Cas vectors which specifically knock-out a particular gene. Next, by profiling the RNA of each individual cell, the effect of the perturbation on a regulator can be assessed over the whole transcriptome. By then linking the RNA profile of a cell to a perturbation of a particular regulator, a regulatory network can be inferred.

This technique has two main limitations, although current studies already show some initial proof-of-concepts to solve them. When a regulator is perturbed, both direct and indirect targets will be affected. In principle, this can be overcome by allowing two or more concurrent perturbations per cell, and computationally deconvolving the contribution of each regulator [34]. This has an added advantage that also combinatorial gene regulation can be analyzed. A second limitation is that the technique is not easily applicable *in vivo*, as the guide-RNA (gRNA) vectors have to be transferred to the cells of interest at a relatively high efficiency. One way to solve this is to infect the gRNA vectors into Cas9 transgenic cells *ex vivo*, transferring these cells to a recipient [33], and after some time again purifying the perturbed cells for RNA sequencing.

Perturbational data also comes with several pitfalls. The main technical challenge is to extract the (combination of) genes which were targeted, as the gRNA of the CRISPR/Cas vector is not polyadenylated and will therefore not be sequenced. This issue can be solved technically, by adding a unique barcode to each expressed gRNA that will lead to polyadenylation [33–35] or by directly cloning the gRNA [36]. Several challenges also need to be solved on the computational side. Current techniques mainly try to (1) handle undetected guide barcodes using imputation strategies [33], (2) try to model the regulatory network as a low number of coregulated sets of genes using matrix decomposition [35], (3) include covariates such as cellular state and genotype in the model [34], and (4) try to model the noise distribution underlying the single-cell data [34]. However, it still has to be seen whether current techniques are powerful enough to correctly infer complete single-cell regulatory networks on large-scale single-cell perturbational data, handling both the peculiarities of single-cell transcriptomics and the combinatorial complexity of the regulatory network.

5 Discussion

Single-cell expression data, through the large number and variability of the cells that they contain, have helped to infer more accurate and specific regulatory networks, as was shown in the different sections of this chapter.

Networks can now be reconstructed from different subpopulations of cells in an unbiased way, without prior experimental separation of the cell populations. Moreover, the availability of the expression patterns of every cell yields a deeper understanding of the underlying differentiation processes of the cells. Continuous differentiation dynamics can be reconstructed, which provides a degree of information that could not be reached in time-series experiments on bulk transcriptomics data. Indeed, when reconstructing a developmental trajectory at the single-cell level, all the important transition states can be recovered, providing knowledge of the transcription factors that drive the main phenotypic changes in a differentiation process. This may be used to identify, for instance, the main transcription factors that drive bifurcation processes in differentiation.

The regulatory processes in a cell may be too complex to be inferred from the expression data alone. A correlation between the higher expression of a certain transcription factor and a set of genes may indicate regulatory interactions between those highly expressed genes. However, due to post-transcription and post-translation regulatory processes in the cells, a highly expressed mRNA may also never lead to a functional protein. And even if this protein is synthesized in the cell, several regulatory processes still may prevent it from reaching its gene target, for instance, the modelling of the chromatin.

One future perspective is therefore the integration of different data types to infer more complex but also more accurate regulatory networks. Such studies have already been initiated, by integrating single-cell expression data with chromatin state studies, to set interactions between genes only if a target gene could be reached by the transcription factor [37, 38]. Another method (SCENIC, *see* Subheading 3.3) uses motif enrichment to filter results from a NI algorithm. This approach has the advantage that it does not need to include any extra single-cell data, and that motif data is now available for almost every transcription factor. However, the simplicity of this approach comes with a cost, as motifs can be very similar between transcription factor family members and can in some cases be very degenerated. Furthermore, the binding of a transcription factor to DNA requires more than just the presence of a motif, and the presence of complex protein regulatory structures should be investigated to identify robust regulatory effects.

Other single-cell data types could therefore be integrated in NI to provide a more context-specific view on transcription factor binding. This is not straightforward from a technical standpoint, as (in the ideal case) both the transcriptome and other data type(s) have to be extracted from the same individual cell. Several studies have already demonstrated techniques of extracting the chromatin accessibility at the single-cell level (single-cell ATAC-seq [39, 40] and single-cell DNase I hypersensitive [41]), the methylome [42, 43], chromatin organization (nuclear lamina interactions [44] and Hi-C [45]), and histone modifications [46]. Most of these techniques suffer from a low sensitivity due to the sparsity of the data, although the density of the data is still high enough to cluster similar cells together in an unsupervised way and further work on the resulting unbiased clusters. Some of these techniques have already been combined with single-cell RNA-seq, such as single-cell M&T seq for the parallel extraction of the transcriptome and methylome [47], and the joint profiling of chromatin accessibility, DNA methylation, and transcription simultaneously [48]. A combined single-cell ATAC-seq and transcriptome technique will potentially have the biggest impact to single-cell NI studies, as it can be combined with motif detection to extract a context-specific picture of transcription factor binding, and conversely of context-dependent regulation. Other major developments are recent methods which can profile parts of the proteome and the transcriptome simultaneously [49, 50], although they are currently still limited to proteins for which antibodies are available.

The field of single-cell NI is now starting to become mature, with almost ten different methods reviewed and new methods being published nearly every month. Different approaches have specific advantages and drawbacks, related to scalability with the number of genes and/or cells, prior assumptions about the network, and the kind of network inferred. With the increasing number of methods, it becomes necessary to independently review the advantages and limitations of certain methods, to not only guide researchers towards the best method for their study, but also steer the development of new methods towards better and complex models of gene regulation. Although we already discussed some of the individual characteristics of current methods, a full-blown evaluation study in which speed and accuracy of single-cell NI methods are being put to the test is still necessary. This could be in the form of a competition, as was the case in the past for bulk NI methods [51]. The main challenge of such evaluations will be the development of a good gold standard, which can be easy to obtain from synthetic data, but hard from real data as the real network is not known. Integration of known binding or motif data could be useful in this case, and has indeed already been used to evaluate some methods on a small scale [17]. In contrast to evaluations on static NI methods, such an evaluation will also have to take

into account the context-specificity when evaluating differential, dynamic, or profile-specific NI methods. Networks inferred for the FANTOM5 project could be useful here [52]. We foresee that such an evaluation study will have a profound impact on the field, similar to what previous evaluations have had on static bulk NI methods [51].

References

1. Padovan-Merhar O, Raj A (2013) Using variability in gene expression as a tool for studying gene regulation. *Wiley Interdiscip Rev Syst Biol Med* 5(6):751–759
2. Stegle O, Teichmann SA, Marioni JC (2015) Computational and analytical challenges in single-cell transcriptomics. *Nat Rev Genet* 16(3):133–145
3. Brennecke P, Anders S, Kim JK, Kołodziejczyk AA, Zhang X, Proserpio V, Baying B, Benes V, Teichmann SA, Marioni JC, Heisler MG (2013) Accounting for technical noise in single-cell RNA-seq experiments. *Nat Methods* 10(11):1093–1095
4. Grün D, Kester L, van Oudenaarden A (2014) Validation of noise models for single-cell transcriptomics. *Nat Methods* 11(6):637–640
5. Marinov GK, Williams BA, McCue K, Schroth GP, Gertz J, Myers RM, Wold BJ (2014) From single-cell to cell-pool transcriptomes: stochasticity in gene expression and RNA splicing. *Genome Res* 24(3):496–510
6. Svensson V, Natarajan KN, Ly LH, Miragaia RJ, Labalette C, Macaulay IC, Cvejic A, Teichmann SA (2017) Power analysis of single-cell RNA-sequencing experiments. *Nat Methods* 14(4):381–387
7. Lun ATL, Bach K, Marioni JC (2016) Pooling across cells to normalize single-cell RNA sequencing data with many zero counts. *Genome Biol* 17(1):75
8. Van Dijk D, Sharma R, Nainys J, Yim K, Kathail P, Carr AJ, Burdziak C, Moon KR, Chaffer CL, Pattabiraman D, Bierie B, Mazutis L, Wolf G, Krishnaswamy S, Pe'er D (2018) Recovering Gene Interactions from Single-Cell Data Using Data Diffusion. *Cell* 174(3):716–729.e27
9. Pierson E, Yau C (2015) ZIFA: dimensionality reduction for zero-inflated single-cell gene expression analysis. *Genome Biol* 16(1):241
10. Risso D, Perraudeau F, Gribkova S, Dudoit S, Vert JP (2018) A general and flexible method for signal extraction from single-cell RNA-seq data. *Nat Commun* 9(1):284
11. Berge KVd, Perraudeau F, Soneson C, Love MI, Risso D, Vert JP, Robinson MD, Dudoit S, Clement L (2018) Observation weights to unlock bulk RNA-seq tools for zero inflation and single-cell applications. *Genome Biol* 19(1):24
12. Vallejos CA, Risso D, Scialdone A, Dudoit S, Marioni JC (2017) Normalizing single-cell RNA sequencing data: challenges and opportunities. *Nat Methods* 14(6):565–571
13. De Smet R, Marchal K (2010) Advantages and limitations of current network inference methods. *Nat Rev Microbiol* 8(10):717–729
14. Svensson V, Vento-Tormo R, Teichmann SA (2018) Exponential scaling of single-cell RNA-seq in the past decade. *Nat Protocols* 13(4):599–604
15. Moignard V, Woodhouse S, Haghverdi L, Lilly AJ, Tanaka Y, Wilkinson AC, Buettner F, Macaulay IC, Jawaid W, Diamanti E, Nishikawa SI, Piterman N, Kouskoff V, Theis FJ, Fisher J, Göttgens B (2015) Decoding the regulatory network of early blood development from single-cell gene expression measurements. *Nat Biotechnol* 33(3):269–276
16. Liu X, Wang Y, Ji H, Aihara K, Chen L (2016) Personalized characterization of diseases using sample-specific networks. *Nucleic Acids Res* 44(22):e164–e164
17. Aibar S, González-Blas CB, Moerman T, Huynh-Thu VA, Imrichova H, Hulselmans G, Rambow F, Marine JC, Geurts P, Aerts J, van den Oord J, Atak ZK, Wouters J, Aerts S (2017) SCENIC: single-cell regulatory network inference and clustering. *Nat Methods* 14(11):1083–1086
18. Chan TE, Stumpf MPH, Babtie AC (2017) Gene regulatory network inference from single-cell data using multivariate information measures. *Cell Syst* 5(3):251–267
19. Filippi S, Holmes CC (2016) A Bayesian non-parametric approach to testing for dependence between random variables. *Bayesian Anal* 12(4):919–938

20. Papp B, Oliver S (2005) Genome-wide analysis of the context-dependence of regulatory networks. *Genome Biol* 6(2):206
21. Xu R, Nettleton D, Nordman DJ (2016) Case-specific random forests. *J Comput Graph Stat* 25(1):49–65
22. Ocone A, Haghverdi L, Mueller NS, Theis FJ (2015) Reconstructing gene regulatory dynamics from high-dimensional single-cell snapshot data. *Bioinformatics* 31(12):i89–i96
23. Wei J, Hu X, Zou X, Tian T (2016) Inference of genetic regulatory network for stem cell using single cells expression data. In: 2016 IEEE international conference on bioinformatics and biomedicine (BIBM). IEEE, Piscataway, pp 217–222
24. Castillo MS, Blanco D, Luna IMT, Carrion MC, Huang Y (2017) A Bayesian framework for the inference of gene regulatory networks from time and pseudo-time series data. *Bioinformatics* 34(6):964–970
25. Matsumoto H, Kiryu H, Furusawa C, Ko MSH, Ko SBH, Gouda N, Hayashi T, Nikaido I (2017) SCODE: an efficient regulatory network inference algorithm from single-cell RNA-Seq during differentiation. *Bioinformatics* 33(15):2314–2321
26. Ideker T, Krogan NJ (2012) Differential network biology. *Mol Syst Biol* 8(565):565
27. Gill R, Datta S, Datta S (2014) Differential network analysis in human cancer research. *Curr Pharm Des* 20(1):4–10
28. Cannoodt R, Saelens W, Saeys Y (2016) Computational methods for trajectory inference from single-cell transcriptomics. *Eur J Immunol* 46(11):2496–2506
29. Kuijjer ML, Tung M, Yuan G, Quackenbush J, Glass K (2018) Estimating sample-specific regulatory networks. arXiv preprint arXiv:150506440v2
30. Eren K, Deveci M, Kucuktunc O, Catalyurek UV (2013) A comparative analysis of biclustering algorithms for gene expression data. *Brief Bioinf* 14(3):279–292
31. Huynh-Thu VA, Irrthum A, Wehenkel L, Geurts P (2010) Inferring regulatory networks from expression data using tree-based methods. *PLoS One* 5(9):e12776
32. Marbach D, Costello JC, Küffner R, Vega NM, Prill RJ, Camacho DM, Allison KR, Aderhold A, Allison KR, Bonneau R, Camacho DM, Chen Y, Collins JJ, Cordero F, Costello JC, Crane M, Dondelinger F, Drton M, Esposito R, Foygel R, de la Fuente A, Gertheiss J, Geurts P, Greenfield A, Grzegorczyk M, Haury AC, Holmes B, Hothorn T, Husmeier D, Huynh-Thu VA, Irrthum A, Kellis M, Karlebach G, Küffner R, Lèbre S, De Leo V, Madar A, Mani S, Marbach D, Mordelet F, Ostrer H, Ouyang Z, Pandya R, Petri T, Pinna A, Poultnay CS, Prill RJ, Rezny S, Ruskin HJ, Saeys Y, Shamir R, Sirbu A, Song M, Soranzo N, Statnikov A, Stolovitzky G, Vega N, Vera-Licona P, Vert JP, Visconti A, Wang H, Wehenkel L, Windhager L, Zhang Y, Zimmer R, Kellis M, Collins JJ, Stolovitzky G (2012) Wisdom of crowds for robust gene network inference. *Nat Methods* 9(8):796–804
33. Jaitin DA, Weiner A, Yofe I, Lara-Astiaso D, Keren-Shaul H, David E, Salame TM, Tanay A, van Oudenaarden A, Amit I (2016) Dissecting immune circuits by linking CRISPR-pooled screens with single-cell RNA-seq. *Cell* 167(7):1883–1896
34. Dixit A, Parnas O, Li B, Chen J, Fulco CP, Jerby-Arnon L, Marjanovic ND, Dionne D, Burks T, Raychowdhury R, Adamson B, Norman TM, Lander ES, Weissman JS, Friedman N, Regev A (2016) Perturb-seq: dissecting molecular circuits with scalable single-cell RNA profiling of pooled genetic screens. *Cell* 167(7):1853–1866
35. Adamson B, Norman TM, Jost M, Cho MY, Nuñez JK, Chen Y, Villalba JE, Gilbert LA, Horlbeck MA, Hein MY, Pak RA, Gray AN, Gross CA, Dixit A, Parnas O, Regev A, Weissman JS (2016) A multiplexed single-cell CRISPR screening platform enables systematic dissection of the unfolded protein response. *Cell* 167(7):1867–1882
36. Datlinger P, Rendeiro AF, Schmidl C, Krausgruber T, Traxler P, Klughammer J, Schuster LC, Kuchler A, Alpar D, Bock C (2017) Pooled CRISPR screening with single-cell transcriptome readout. *Nat Methods* 14(3):297–301
37. Äijö T, Bonneau R (2016) Biophysically motivated regulatory network inference: progress and prospects. *Hum Hered* 81(2):62–77
38. Petralia F, Wang P, Yang J, Tu Z (2015) Integrative random forest for gene regulatory network inference. *Bioinformatics* 31(12):i197–i205
39. Buenrostro JD, Wu B, Litzenburger UM, Ruff D, Gonzales ML, Snyder MP, Chang HY, Greenleaf WJ (2015) Single-cell chromatin accessibility reveals principles of regulatory variation. *Nature* 523(7561):486–490

40. Cusanovich DA, Daza R, Adey A, Pliner HA, Christiansen L, Gunderson KL, Steemers FJ, Trapnell C, Shendure J (2015) Multiplex single-cell profiling of chromatin accessibility by combinatorial cellular indexing. *Science* 348(6237):910–914
41. Jin W, Tang Q, Wan M, Cui K, Zhang Y, Ren G, Ni B, Sklar J, Przytycka TM, Childs R, Levens D, Zhao K (2015) Genome-wide detection of DNase I hypersensitive sites in single cells and FFPE tissue samples. *Nature* 528(7580):142
42. Smallwood SA, Lee HJ, Angermueller C, Krueger F, Saadeh H, Peat J, Andrews SR, Stegle O, Reik W, Kelsey G (2014) Single-cell genome-wide bisulfite sequencing for assessing epigenetic heterogeneity. *Nat Methods* 11(8):817–820
43. Farlik M, Sheffield NC, Nuzzo A, Datlinger P, Schönegger A, Klughammer J, Bock C (2015) Single-cell DNA methylome sequencing and bioinformatic inference of epigenomic cell-state dynamics. *Cell Rep* 10(8):1386–1397
44. Kind J, Pagie L, de Vries S, Nahidazar L, Dey S, Bienko M, Zhan Y, Lajoie B, de Graaf C, Amendola M, Fudenberg G, Imakaev M, Mirny L, Jalink K, Dekker J, van Oudenaarden A, van Steensel B (2015) Genome-wide Maps of Nuclear Lamina Interactions in Single Human Cells. *Cell* 163(1):134–147
45. Nagano T, Lubling Y, Stevens TJ, Schoenfelder S, Yaffe E, Dean W, Laue ED, Tanay A, Fraser P (2013) Single-cell Hi-C reveals cell-to-cell variability in chromosome structure. *Nature* 502(7469):59–64
46. Rotem A, Ram O, Shores N, Sperling RA, Goren A, Weitz DA, Bernstein BE (2015) Single-cell ChIP-seq reveals cell subpopulations defined by chromatin state. *Nat Biotechnol* 33(11):1165–1172
47. Angermueller C, Clark SJ, Lee HJ, Macaulay IC, Teng MJ, Hu TX, Krueger F, Smallwood SA, Ponting CP, Voet T, Kelsey G, Stegle O, Reik W (2016) Parallel single-cell sequencing links transcriptional and epigenetic heterogeneity. *Nat Methods* 13(3):229–232
48. Clark SJ, Argelaguet R, Kapourani CA, Stubbs TM, Lee HJ, Alda-Catalinas C, Krueger F, Sanguinetti G, Kelsey G, Marioni JC, Stegle O, Reik W (2018) scNMT-seq enables joint profiling of chromatin accessibility DNA methylation and transcription in single cells. *Nat Commun* 9(1):781
49. Genshaft AS, Li S, Gallant CJ, Darmanis S, Prakadan SM, Ziegler CGK, Lundberg M, Fredriksson S, Hong J, Regev A, Livak KJ, Landegren U, Shalek AK (2016) Multiplexed, targeted profiling of single-cell proteomes and transcriptomes in a single reaction. *Genome Biol* 17(1):188
50. Frei AP, Bava FA, Zunder ER, Hsieh EWY, Chen SY, Nolan GP, Gherardini PF (2016) Highly multiplexed simultaneous detection of RNAs and proteins in single cells. *Nat Methods* 13(3):269–275
51. Marbach D, Prill RJ, Schaffter T, Mattiussi C, Floreano D, Stolovitzky G (2010) Revealing strengths and weaknesses of methods for gene network inference. *Proc Natl Acad Sci U S A* 107(14):6286–6291
52. Marbach D, Lamparter D, Quon G, Kellis M, Kutalik Z, Bergmann S (2016) Tissue-specific regulatory circuits reveal variable modular perturbations across complex diseases. *Nat Methods* 13(4):366–370



Chapter 11

Inferring Gene Regulatory Networks from Multiple Datasets

Christopher A. Penfold, Iulia Gherman, Anastasiya Sybirna,
and David L. Wild

Abstract

Gaussian process dynamical systems (GPDS) represent Bayesian nonparametric approaches to inference of nonlinear dynamical systems, and provide a principled framework for the learning of biological networks from multiple perturbed time series measurements of gene or protein expression. Such approaches are able to capture the full richness of complex ODE models, and can be scaled for inference in moderately large systems containing hundreds of genes. Related hierarchical approaches allow for inference from multiple datasets in which the underlying generative networks are assumed to have been rewired, either by context-dependent changes in network structure, evolutionary processes, or synthetic manipulation. These approaches can also be used to leverage experimentally determined network structures from one species into another where the network structure is unknown. Collectively, these methods provide a comprehensive and flexible platform for inference from a diverse range of data, with applications in systems and synthetic biology, as well as spatiotemporal modelling of embryo development. In this chapter we provide an overview of GPDS approaches and highlight their applications in the biological sciences, with accompanying tutorials available as a Jupyter notebook from <https://github.com/cap76/GPDS>.

Key words Nonlinear dynamical systems, Gaussian process dynamical systems, Causal structure identification, Learning from multiple data sources, Spatiotemporal models

1 Introduction

A major challenge in systems and synthetic biology is the ability to model complex regulatory interactions, providing a compact and accurate representation of a biological system that is capable of predicting behavior under novel conditions. A number of computational approaches have been developed over recent years to address these challenges. *Bottom-up* approaches begin with a detailed mathematical description of the biophysical processes involved in a regulatory pathway, often expressed as a set of differential equations [1] that may explicitly model mRNA transcription and protein translation, as well as transcription factor

diffusion, binding, and degradation. To date, these approaches have been mainly applied to small, well-defined biological systems, and tend to focus on refining network structure or providing detailed kinetic parameterization. In contrast, the so-called *top-down* approaches attempt to *infer* or “reverse-engineer” regulatory networks from high-throughput data sources, often representing the system in terms of (more abstract) statistical relationships. These kinds of representations are typically used for exploratory analysis in systems where the structure of the biological network is uncertain, with a focus on inferring network topology, rather than detailed kinetics. Over recent years, concomitant with the rise of informative datasets, efforts have been made to develop scalable, top-down approaches that are capable of both inferring network structure and capturing the full richness and nonlinearities of the molecular kinetics.

A well-known example of a gene regulatory network (GRN) is the *repressilator* [2], a synthetic three-gene system comprised of three mRNAs and three proteins as shown in Fig. 1. The mutual repression of gene expression in this relatively simple network can give rise to complex nonlinear dynamics, which can be described by the following coupled set of ordinary differential equations (ODEs):

$$\begin{aligned}\dot{m}_1 &= -m_1 + \frac{\alpha_1}{1 + p_3^{n_1}} + \alpha_{0,1}, \\ \dot{p}_1 &= -\beta_1(p_1 - m_1), \\ \dot{m}_2 &= -m_2 + \frac{\alpha_2}{1 + p_1^{n_2}} + \alpha_{0,2}, \\ \dot{p}_2 &= -\beta_2(p_2 - m_2), \\ \dot{m}_3 &= -m_3 + \frac{\alpha_3}{1 + p_2^{n_3}} + \alpha_{0,3}, \\ \dot{p}_3 &= -\beta_3(p_3 - m_3),\end{aligned}\tag{1}$$

where m_i and p_i represent the transcript level and protein level of gene i , respectively, $\alpha_{0,i}$ represents the basal transcription rate for mRNA i when bound by its repressive transcription factor, $\alpha_i + \alpha_{0,i}$ represents the transcription rate for the unbound state, β_i represents the degradation rate of protein i , and n_i represents the Hill coefficient. A complete description of this relatively simple system therefore requires knowledge of the 6 molecular interactions and 12 biophysical parameters. Other regulatory motifs, such as genetic toggle switches [3], can easily be represented as ODE models, and combined with other motifs to build complex genetic circuits capable of exhibiting nonlinear dynamics [4]. Although ODE modelling has shed light on a variety of interesting biological phenomena including the *Arabidopsis thaliana* circadian clock [5–9], and mammalian NFκB signalling [8, 10–12], the

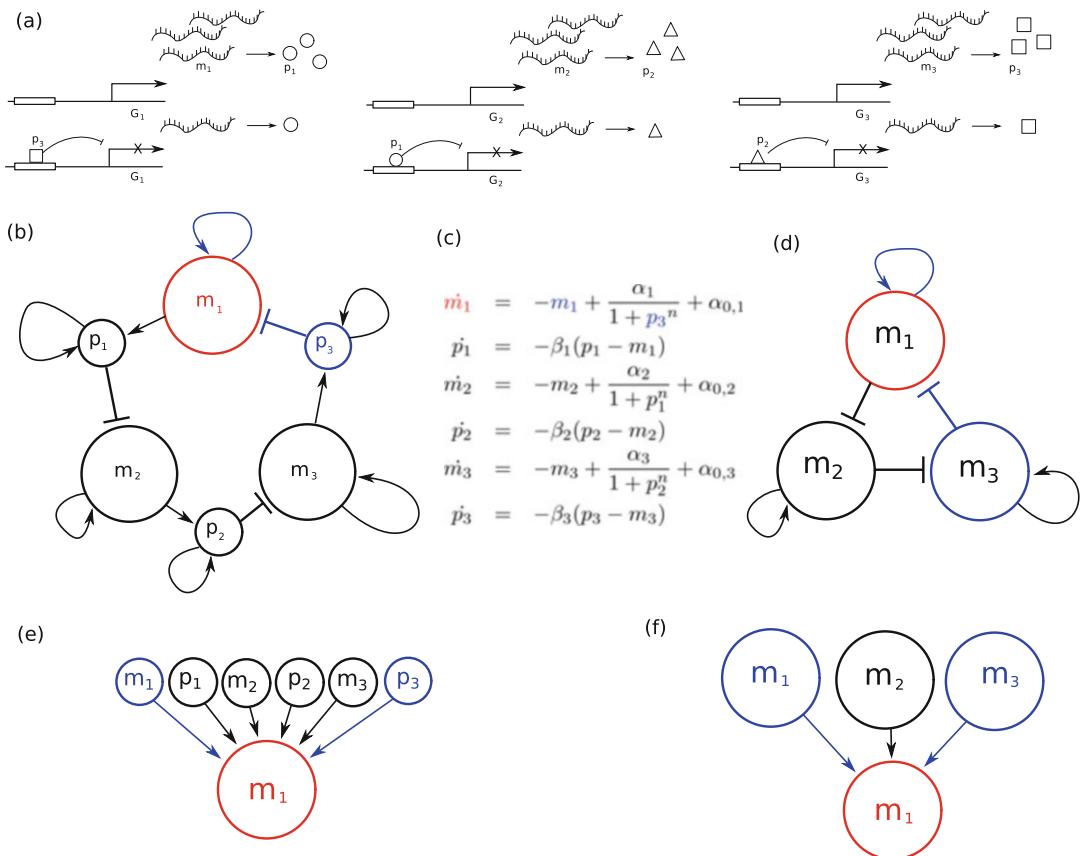


Fig. 1 The repressilator is a synthetic three-gene network. (a) Gene 1 encodes for mRNA 1 (m_1) that is translated into protein 1 (p_1), with gene 2 encoding for mRNA and protein 2 (m_2 , p_2), and gene 3 encoding for mRNA 3 and protein 3 (m_3 , p_3). Within the system, p_1 suppresses the expression of m_2 , p_2 suppresses m_3 , and p_3 represses the expression of m_1 . (b) This system of mutual repression can be represented as a six component network, with kinetics described by the system of ordinary differential equations (c), and evolution of the system is defined by expression levels of the six molecular species [m_1 , p_1 , m_2 , p_2 , m_3 , p_3]. (d) As protein levels are not usually measured, the repressilator is often represented as a three component system of three genes that mutually repress one another; the system would therefore be defined by three observed molecular species [m_1 , m_2 , m_3]. (e) The *parents* of a given gene, i , denoted $\mathcal{P}a(i)$, are defined as the set of regulators with incoming edges, i.e., the molecular species that directly influence the evolution of the ODE system. For mRNA1, the parents are m_1 and p_3 (blue nodes), or molecular species 1 and 6, $\mathcal{P}a(1) = \{1, 6\}$; in the reduced system (f), the parents would be 1 and 3, respectively, $\mathcal{P}a(1) = \{1, 3\}$

quantity of genes modelled in these systems tends to be limited, numbering in the dozens, with hundreds of parameters. Only with advances in Markov chain Monte Carlo (MCMC) [13] and approximate Bayesian computation (ABC) [14, 15], has ODE modelling become more applicable to systems-level biological applications. For an alternative perspective on learning systems of ODEs, see also Chapter 16.

In the biological sciences, data and models such as these ODEs play important complementary roles. Data can be used to validate models, and models can predict data. In fact, in the statistical sense, a model can be thought of as a summary of the distribution over observable data. Since models are never perfect representations of reality in the sciences, they must faithfully capture both uncertainty and noise (unpredictable variability) in their predictions. Probability theory provides a coherent mathematical language for expressing the uncertainty in models. Bayesian statistics, which used to be called “inverse probability” until the twentieth century, refers to the application of probability theory to *infer* unknown quantities (such as model parameters or predictions) from observable data. Bayesian statistics can also be used to compare multiple models (i.e., hypotheses) given the data, and thus can play a fundamental role in scientific hypothesis testing. In the past the computational power required for Bayesian statistical analyses has limited their use in biological data analysis. These constraints have now largely been overcome and the advantages of Bayesian approaches have put them at the forefront of genetic and other biological data analysis [16].

1.1 Bayesian Methods

At the heart of Bayesian analysis is *Bayes’ rule* or theorem. Consider a dataset \mathcal{D} , and a statistical model \mathcal{M} with parameters θ , which explains how the observed data may be described by the parameter(s). The conditional distribution $P(\mathcal{D}|\theta, \mathcal{M})$ is known as the *likelihood*, and represents the probability of the observed data given particular values for the parameters and the underlying model. The *prior* probability distribution over the model parameters, $P(\theta|\mathcal{M})$, represents information about the values of the parameters before we observe any data, given the model. Bayes’ rule combines these probability distributions in the following way:

$$P(\theta|\mathcal{D}, \mathcal{M}) = \frac{P(\mathcal{D}|\theta, \mathcal{M})P(\theta|\mathcal{M})}{P(\mathcal{D}|\mathcal{M})},$$

where $P(\theta|\mathcal{D}, \mathcal{M})$ is known as the *posterior* distribution, the key quantity needed for Bayesian inference. The normalizing constant $P(\mathcal{D}|\mathcal{M})$ is known as the *marginal likelihood* or *evidence*, and is obtained by integrating over the model parameters

$$P(\mathcal{D}|\mathcal{M}) = \int P(\mathcal{D}|\theta, \mathcal{M})P(\theta|\mathcal{M})d\theta.$$

It is the calculation of the marginal likelihood (that is, this integration or summation) that was previously a computational bottleneck in Bayesian analyses. The marginal likelihood may be interpreted as the probability that randomly selected parameters from the prior would generate the dataset observed, \mathcal{D} , and may

be used as a yardstick for model selection; model classes that are too simple are unlikely to generate the observed data, whereas model classes that are too complex can generate many possible datasets, and so, again, they are unlikely to generate our observed dataset at random. If we have several competing models for the data, \mathcal{M}_i , we can define a prior distribution over models $P(\mathcal{M}_i)$, and obtain a posterior distribution over models via Bayes' rule:

$$P(\mathcal{M}_i|\mathcal{D}) = \frac{P(\mathcal{D}|\mathcal{M}_i)P(\mathcal{M}_i)}{P(\mathcal{D})}.$$

Although Bayesian methods generally provide a powerful framework for modelling, representing uncertainty, and combining prior knowledge with data, simple *parametric* models, for example, linear models, are often inadequate for modelling the complexity of real-world biological, physical, and social processes. In the quest for creating more flexible modelling tools with a view to maintaining predictive reliability, recent research has turned to the limit of Bayesian models with infinitely many latent variables and parameters—these are also known as *Bayesian nonparametric models* [17]. The term *nonparametric* is used because the model cannot be represented in terms of its parameters, since there would be an infinite number of them to learn and store, but instead is represented in terms of other quantities derived from the data that encapsulate the effect of these infinite number of parameters. These Bayesian nonparametric models allow much greater flexibility and offer the promise of greater realism than existing approaches, especially for large datasets where the data can “speak for itself.” One such example for inference of dynamical systems is a model based on *Gaussian processes*, described in the next section.

2 Gaussian Process Dynamical Systems (GPDS)

A *dynamical system*, such as a set of ODEs, is a set of component variables whose values change over time. In systems biology applications, these component variables are typically the individual mRNA or protein expression levels associated with a set of genes, and we are interested in understanding the mechanism through which these components evolve over time. For inference, the goal is the identification of the system components which cause, or directly influence, changes in other components, and the corresponding set of functions that dictate *what* those changes look like quantitatively. This *causal structure* defines an interaction map or network which can be represented as a directed graph. In general, an ODE system, such as the repressilator, evolves over continuous time according to:

$$\dot{\mathbf{x}}(t) = \mathcal{g}(\mathbf{x}(t), \mathcal{G}), \quad (2)$$

where $\mathbf{x}(t)$ represents a vector of the expression of the individual mRNA or proteins at time t , \mathcal{G} represents the causal network structure, and $g(\cdot)$ represents the (potentially nonlinear) function governing the evolution over time of \mathbf{x} .

Gaussian process dynamical systems (GPDS) represent a class of probabilistic models able to capture a range of dynamical systems such as the repressilator, in a scalable, nonparametric way. These approaches were initially developed to capture the rich dynamics seen in relatively small systems encountered in the physical sciences [18–25]. Developments by Klemm et al. [26] described their application to the inference of small gene regulatory networks, and their utility was subsequently demonstrated in much larger systems containing hundreds of genes [27, 28].

When the causal network structure, \mathcal{G} , is known, and all molecular species have been measured, a GPDS might simply replace the parameterized ODE model with a set of nonparametric approximations. For the repressilator system, for example, we modify Eq. (1) to give:

$$\begin{aligned}\dot{m}_1 &= f_1(m_1, p_3), \\ \dot{p}_1 &= f_2(p_1, m_1), \\ \dot{m}_2 &= f_3(m_2, p_1), \\ \dot{p}_2 &= f_4(p_2, m_2), \\ \dot{m}_3 &= f_5(m_3, p_2), \\ \dot{p}_3 &= f_6(p_3, m_3),\end{aligned}\tag{3}$$

where $f_i(\cdot)$ denotes an unknown, potentially nonlinear function that describes the dynamics of molecular species i from the expression dynamics of its causal regulators (often called parents), denoted $\mathcal{P}a(i)$. For example, the parental set of mRNA 1 is denoted $\mathcal{P}a(1) = \{m_1, p_3\}$, which may be encoded as a binary indicator $[1, 0, 0, 0, 0, 1]$. The union of parents for all components together define the causal network structure, \mathcal{G} , while the set of functions $f_i(\cdot)$ define the function $g(\cdot)$. In this case, we can assign the unknown function, $f_i(\cdot)$, a suitable prior distribution and use a version of Bayes' rule to learn functions from data:

$$P(f_i|\mathcal{D}) = \frac{P(f_i)P(\mathcal{D}|f_i)}{P(\mathcal{D})},$$

where $P(f_i)$ represents a *prior distribution over functions*. A suitable prior distribution over functions is given by the *Gaussian process* (GP) [29].

2.1 Gaussian Processes

In a Bayesian setting, Gaussian processes are infinite-dimensional generalizations of the Gaussian distribution used to represent prior distributions over functions, allowing us to perform nonlinear regression. Typically we assume some observation data, $\mathbf{y} = (y_1, \dots, y_n)^\top$, at a set of input locations, \mathbf{x} , which represent noisy corruptions of a smooth, nonlinear function, $y = f(\mathbf{x}) + \varepsilon$. For inference, we assign the unknown function a Gaussian process prior, which is completely defined by its *mean function* and *covariance function*, $f(\mathbf{x}) \sim \mathcal{GP}(\mu(\mathbf{x}), K(\mathbf{x}, \mathbf{x}'))$. We can assume a zero-mean function, $\mu(\mathbf{x}) = 0$, without loss of generality, but must specify a covariance function, which defines the general behavior of the nonlinear functions, prior to inference. A significant number of covariance functions exist, encoding a variety of assumptions, including smooth or periodic functions (see [29]), and more complex behavior [30–34].

Throughout this chapter, we will assume a squared exponential covariance function, which encodes for smooth functions, $K(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp((\mathbf{x} - \mathbf{x}')^2 / 2l^2)$, with hyperparameters $\theta = [l, \sigma_f]$, where σ_f defines the process variance, and l defines the function length-scale: smaller values of l will give functions that vary more sharply over \mathbf{x} , while larger values of σ_f will give functions with larger amplitudes.

A Gaussian process prior induces a natural Gaussian prior distribution over the observational data, $p(\mathbf{y}) = \mathcal{N}(\mathbf{y} | \mu(\mathbf{x}), K(\mathbf{x}, \mathbf{x}))$, and we may infer a posterior distribution over \mathbf{y} using Bayes' rule. In Fig. 2 we illustrate Gaussian process regression using a simple example, for a one-dimensional function, $y = f(x)$. In Fig. 2a we indicate the GP prior, showing the prior mean (solid black line) and 95% confidence intervals (CI) for y at various points of x . We can draw samples from the GP prior, shown in Fig. 2b. Given an observation, and using Bayes' rule, we can update the GP prior to yield a posterior Gaussian process, performing nonlinear regression, as shown in Fig. 2c. Observations of y at x can be said to “pin down the GP.” As we increase the number of observations the posterior GP begins to resemble the true underlying function, and the associated 95% confidence intervals become smaller; similarly, sample functions from the posterior GP begin to resemble the true function, as shown in Fig. 2d. In Fig. 2e we show sample prior functions for various values of the hyperparameter l . For a given hyperparameter and set of data, we can generate a posterior GP, shown in Fig. 2f. We can also evaluate the marginal likelihood of the GP: $\mathcal{L}(\theta) = \int P(\mathbf{y}|\mathbf{f}, \mathbf{x})P(\mathbf{f}|\mathbf{x})d\mathbf{f}$, allowing us to automatically select the optimal hyperparameter values. In Fig. 2g we show corresponding marginal likelihoods for the posterior GPs in Fig. 2f. GPs can also be used to specify distributions over functions over higher dimensional spaces, and in Fig. 2h we show an example

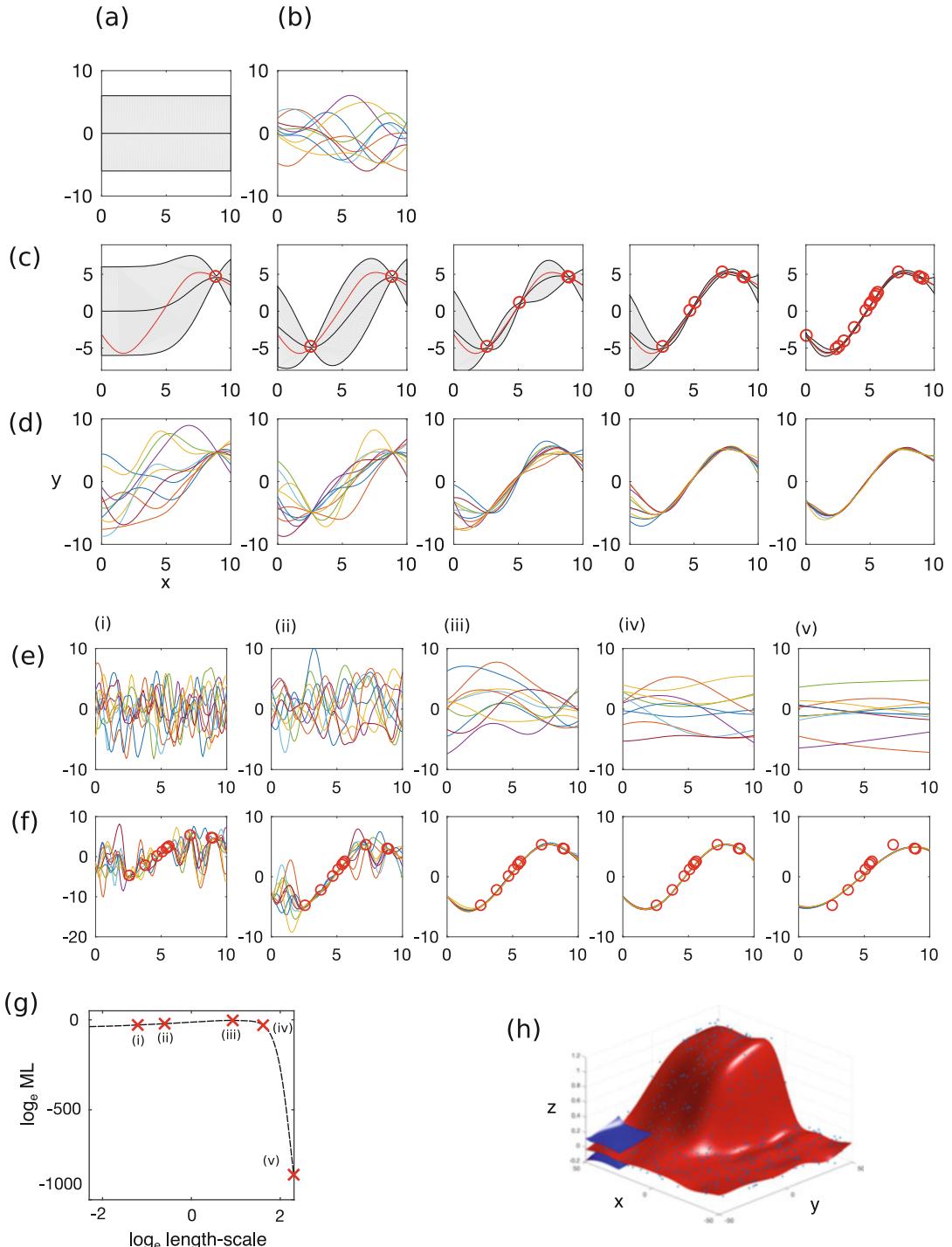


Fig. 2 In a Bayesian setting, Gaussian processes (GPs) are infinite generalizations of the Gaussian distribution used to represent prior distributions over functions, allowing us to perform nonlinear regression. (a) The prior mean (solid black line) and 95% confidence intervals of f at various points of x . (b) Samples from the GP prior. (c) Nonlinear regression using a GP. As we increase the number of observations the posterior GP begins to resemble the true underlying function, and the associated 95% confidence intervals become smaller. (d) Similarly, sample functions from the posterior GP

posterior GP for a two-dimensional function, $z = f(x, y)$, with the posterior mean of the GP indicated by the red surface, and an illustrative 95% confidence intervals shown as blue surfaces for a small region of the space.

In the specific example of the repressor, the data in question corresponds to the expression levels and derivatives of the individual mRNA and protein species, measured at times, $t = (t_1, \dots, t_T)$, under condition s_1 . For simplicity of notation, we can concatenate the data for all molecular species at all time points into matrices $\mathbf{X}^{(s_1)}$ and $\mathbf{Y}^{(s_1)}$:

$$\mathbf{X}^{(s_1)} = \begin{bmatrix} m_1^{(s_1)}(t_1), p_1^{(s_1)}(t_1), \dots, m_3^{(s_1)}(t_1), p_3^{(s_1)}(t_1) \\ \vdots & \ddots & \vdots \\ m_1^{(s_1)}(t_T), p_1^{(s_1)}(t_T), \dots, m_3^{(s_1)}(t_T), p_3^{(s_1)}(t_T) \end{bmatrix},$$

$$\mathbf{Y}^{(s_1)} = \begin{bmatrix} \dot{m}_1^{(s_1)}(t_1), \dot{p}_1^{(s_1)}(t_1), \dots, \dot{m}_3^{(s_1)}(t_1), \dot{p}_3^{(s_1)}(t_1) \\ \vdots & \ddots & \vdots \\ \dot{m}_1^{(s_1)}(t_T), \dot{p}_1^{(s_1)}(t_T), \dots, \dot{m}_3^{(s_1)}(t_T), \dot{p}_3^{(s_1)}(t_T) \end{bmatrix}.$$

Since, in this particular example, the regulators are known, we can focus on learning (a distribution over) the individual functions that govern each of the molecular species. For mRNA 1, for example, the parental set consists of m_1 and p_3 (Fig. 1a, b), and we are interested in learning how $\dot{\mathbf{m}}_1 = (\dot{m}_1(t_1), \dots, \dot{m}_1(t_T))^\top$ is predicted by $\mathbf{m}_1 = (m_1(t_1), \dots, m_1(t_T))^\top$ and $\mathbf{p}_3 = (p_3(t_1), \dots, p_3(t_T))^\top$, in a way that allows us to generalize, so we can predict $\dot{\mathbf{m}}_1^*(t)$ given new observations of $\mathbf{m}_1^*(t)$ and $\mathbf{p}_3^*(t)$, not previously seen by the model. To do so, we first plot columns 1 and 6 of $\mathbf{X}^{(s_1)}$ against column 1 of $\mathbf{Y}^{(s_1)}$, thus mapping the time series into its corresponding *phase space*, as illustrated in Fig. 3a, b. Once we have mapped the time series into the appropriate phase space, inference of the function, $f_1(\cdot)$, becomes a *regression* task, and we can attempt to learn the (potentially) nonlinear function using Bayes' rule with a GP prior.

For a single time series, it seems unlikely there would be sufficient information to accurately capture the full range of dynamics available to mRNA 1, and the posterior GP will therefore have large amount of uncertainty, as illustrated by the large 95% CIs

 **Fig. 2** (continued) begin to resemble the true function. **(e)** Sample prior functions for various values of the hyperparameter l . **(f)** Posterior GPs for a given hyperparameter and set of data. **(g)** The corresponding marginal likelihoods for the posterior GPs indicated in panel (f). **(h)** An example posterior GP for a two-dimensional function, $f(x, y)$, with the mean of the GP shown by the red surface, and illustrative 95% confidence intervals shown by blue surfaces for a small region of the space

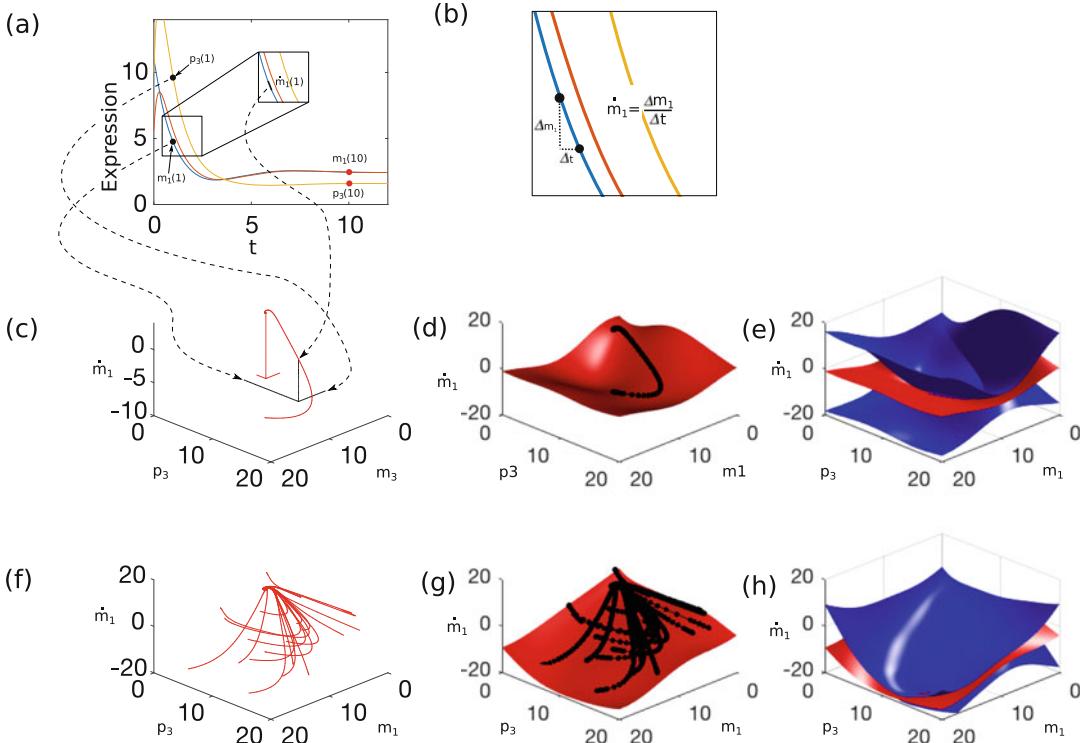


Fig. 3 (a) Example expression profiles for mRNA m_1 (blue) and proteins p_1 (red) and p_3 (yellow) of the repressilator. (b) Typically we will not be able to measure the derivative of the protein or mRNA expression levels directly, but instead must calculate it prior to inference. Previous studies by Äijö and Lähdesmäki [35] calculate this derivative from the expression data, $\dot{y}_i(t) = \Delta y_i(t)/\Delta t$, while others [26, 27] infer this using an independent GP [36]. (c) As we know the regulators of m_1 , our aim is to directly infer the dynamic behavior from the data. We can do this by mapping the expression data into the corresponding *phase space*, which we do by plotting m_1 and p_3 against \dot{m}_1 . In (a, c) we show this mapping explicitly for one data point at time point, $t = 1$. Once the data has been mapped into the corresponding phase space, we can attempt to capture the behavior of the system by fitting a Gaussian process model to the data. (d) Although we can fit a GP to a single time series data, as indicated by the mean function (red), the dataset is unlikely to contain sufficient information to be able to accurately predict behavior at new locations, and the posterior GP has a large amount of uncertainty as indicated by the 95% CIs (e). (f) We can map multiple trajectories, corresponding to different perturbed time series, into this *phase space*. (f, g) When we have enough observations, the GP can accurately capture the behavior of the system, with the mean (g) closely approximating the true underlying function with reduced uncertainty (h)

in Fig. 3e. When we have multiple time series under a range of different initial conditions, $\{s_1, \dots, s_N\}$, however, we can combine the time series together:

$$\mathbf{X} = \begin{bmatrix} \mathbf{X}^{(s_1)} \\ \vdots \\ \mathbf{X}^{(s_N)} \end{bmatrix}, \quad \mathbf{Y} = \begin{bmatrix} \mathbf{Y}^{(s_1)}, \\ \vdots \\ \mathbf{Y}^{(s_N)} \end{bmatrix}.$$

Using this extended set of observations, we can again plot columns 1 and 6 of \mathbf{X} against column 1 of \mathbf{Y} (Fig. 3f) to map these time series into their *phase space*, and perform nonlinear regression using GPs. As the number of perturbed time series increases, the data begins to span a much fuller range of the phase space, and the GP is able to capture the behavior that allows generalization to regions where no observations exist, with less uncertainty (Fig. 3g, h).

2.2 Inference in Unknown Systems

In most circumstances, the regulatory network is not known ahead of inference, or has only been partially elucidated. In this case, the parental set of each component of the network must additionally be inferred alongside a GP model for the molecular kinetics. Ideally, we would search over all possible combinations of putative regulators, possibly eliminating those models we know to be impossible. As an example, we can consider mRNA 1 of the repressilator system, for which there exist six putative regulators: m_1, m_2, m_3, p_1, p_2 , and p_3 . We know that \dot{m}_1 depends on m_1 but not directly on m_2 and m_3 due to the physical nature of the system, but may be unsure about which combinations of proteins p_1, p_2 , and p_3 are regulators. Here we can explore a number of different models, by plotting \dot{m}_1 as a function of m_1 and p_1 (model 1, \mathcal{M}_1 , representing parental set $\mathcal{Pa}_1(1) = \{1, 2\}$), as a function of m_1 and p_2 ($\mathcal{M}_2, \mathcal{Pa}_2(1) = \{1, 4\}$), and as a function of m_1 and p_3 ($\mathcal{M}_3, \mathcal{Pa}_3(1) = \{1, 6\}$) (see Fig. 4a). For each model, we use a GP to capture the molecular kinetics; crucially, as GPs represent a probabilistic model, we can calculate the marginal likelihood (Fig. 4d) for each putative set of parents.

$$P(\mathbf{Y}_i|\mathbf{X}, \mathcal{M}_i) = \int P(\mathbf{Y}_i|\mathbf{X}, f_{\mathcal{M}_i}, \mathcal{Pa}(i)) P(f_i|\mathcal{M}_i) df_{\mathcal{M}_i},$$

where $f_{\mathcal{M}_i}$ represents the distribution over functions associated with model \mathcal{M}_i .

Finally, we can construct a posterior distribution over these various parental sets (or models), using Bayes' rule (Fig. 4e),

$$P(\mathcal{Pa}_i|\mathbf{X}, \mathbf{Y}_i) = P(\mathcal{M}_i|\mathbf{X}, \mathbf{Y}_i), \quad (4)$$

$$= \frac{P(\mathbf{Y}_i|\mathbf{X}, \mathcal{M}_i)}{\sum_{j=1}^3 P(\mathbf{Y}_j|\mathbf{X}, \mathcal{M}_j)}. \quad (5)$$

In this particular example, we only explored a limited number of models (and therefore limited number of parental sets). Naively, we would like to search over more complex combinations of interactions: m_1, p_1 , and p_2 (\mathcal{M}_4); m_1, p_1 , and p_3 (\mathcal{M}_5); m_1, p_2 , and p_3 (\mathcal{M}_6); and m_1, p_1, p_2 , and p_3 (\mathcal{M}_7). While exhaustive searches over all possible combinations of regulators are possible for smaller systems, such as the repressilator, in practice the *number of combinations* scales poorly with the number of putative regulators, and

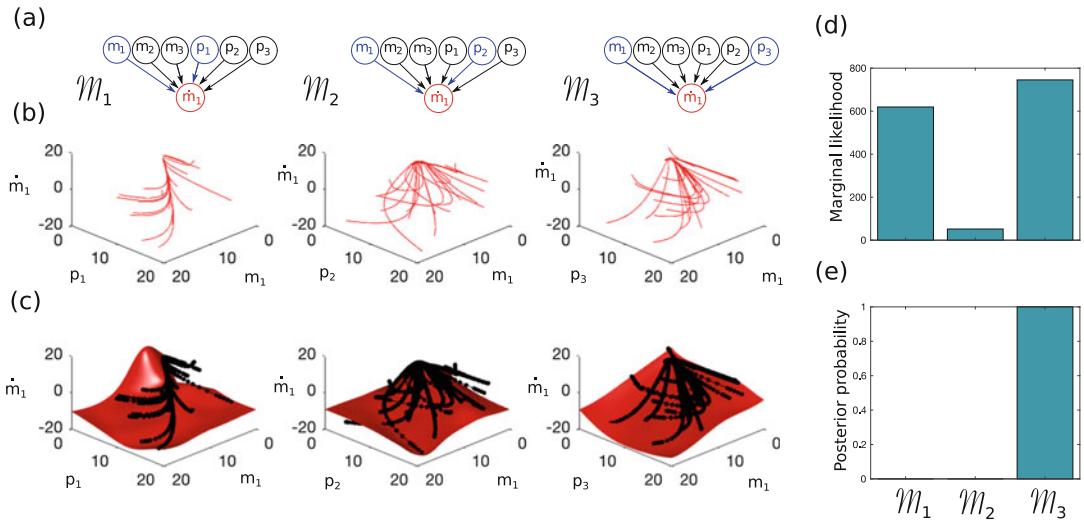


Fig. 4 Typically we do not know the parental set ahead of time, but must additionally infer it from the data. To do so, we assume a number of competing models (a), representing the different possible combinations of regulators for a given component of the network (given gene or protein), with putative regulators denoted as blue nodes in the network. In this example, we are looking to identify the regulators of mRNA 1, and assume three models, \mathcal{M}_1 , \mathcal{M}_2 , and \mathcal{M}_3 , in which the parental set is [1, 4], [1, 5], and [1, 6]. Specifically, we have constructed these models by assuming that the derivative of the expression level is related to expression levels, and that any one of the three proteins can bind and regulate. For each putative set, we can plot the derivative of the component (protein or mRNA) of interest as a function of the various putative parental sets (b), and fit a Gaussian process to each one (c). (d) For each putative model, we can evaluate the marginal likelihood and use Bayes' rule to evaluate a posterior distribution over models. (e) In this particular example, model \mathcal{M}_3 is identified as the best model, representing regulation of \dot{m}_1 by m_1 and p_3 , the correct structure

exhaustive modelling becomes unfeasible for systems with even a moderate number of regulators. Ideally, we could limit the set of putative regulators using prior knowledge; for example, if we explicitly knew that m_3 was a regulator, we could confine our search to include only combinations of regulators that contain m_3 . Alternatively, we could exclude combinations that are likely to be unfeasible, such as eliminating sets containing two or more regulators known to bind the same regulatory motif. For larger systems, where prior information is limited and putative regulators are unknown, we can instead limit the maximum number of simultaneous regulators to be less than some integer, c_{\max} , encoding the assumption that at most c_{\max} regulators are involved in regulating the gene of interest. Allowing up to three simultaneous regulators in the repressilator system by setting $c_{\max} = 3$ would include models \mathcal{M}_1 through to \mathcal{M}_6 , but would eliminate \mathcal{M}_7 . In Table 1 we indicate the number of possible models (parental sets) as a function of the number of putative parents using exhaustive searching versus setting $c_{\max} = 2$ and $c_{\max} = 5$. Limiting the parental sets in this manner reduces the computational scaling from factorial to polynomial in the number of putative parents.

Table 1

The number of combinations of regulators, i.e., number of models N_M , using exhaustive searching scales factorially with the number of putative transcription factors, N_P

N_P	N_M		
	Exhaustive	$c_{\max} = 2$	$c_{\max} = 5$
10	1023	55	637
50	1.13×10^{15}	1275	2,369,935
100	1.27×10^{30}	5050	7.94×10^7

Limiting the maximum number of simultaneous regulators by setting c_{\max} reduces this to polynomial scaling

2.3 Gaussian Process Hyperparameters

For each of the models we search over, there exists a corresponding Gaussian process model for the molecular kinetics. The fit of each GP model depends on the type of covariance function, and the value of the *hyperparameters*, θ , which must be appropriately set. Studies by Äijö and Lähdesmäki [35] assume that each of the GP models has its own independent set of hyperparameters, which can be tuned to maximize the marginal likelihood for that model. The hyperparameter for parental set k of gene i is thus set to:

$$\hat{\theta}_k(i) = \arg_{\theta_k(i)} \max \left(P_{\theta_k(i)}(\mathbf{Y}_i | \mathbf{X}, \mathcal{M}_k) \right).$$

Depending on the type of covariance function used, this results in nN_i hyperparameters, where n represents the number of *hyperparameters* in the covariance function and N_i the total number of parental sets for gene i . For most covariance functions n is typically small, but, as we have seen from Table 1, N can be excessively large. An alternative approach is to instead tie the hyperparameters, such that the values are identical for each parental set [26, 27]. Here the hyperparameters are optimized as:

$$\hat{\theta}(i) = \arg_{\theta(i)} \max \left(\sum_{j=1}^3 P_{\theta(i)}(\mathbf{Y}_i | \mathbf{X}, \mathcal{M}_j) \right).$$

Although the tying of hyperparameters can be considered advantageous, reducing the number of hyperparameters to just n , direct optimization via gradient descent is no longer possible, and must instead be approximated using an expectation-maximization (EM) algorithm [26, 27].

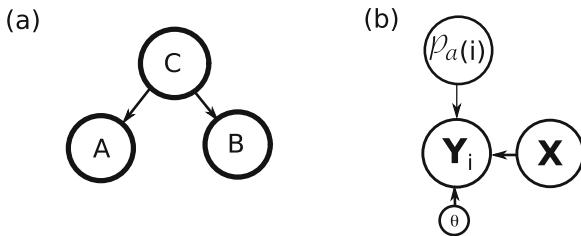


Fig. 5 (a) Graphical model representing conditional independencies. (b) A graphical model for learning the regulators and hyperparameters for component i using CSI

2.4 Causal Structure Identification

Although learning a GP model for a single component of the network is not, in itself, necessarily useful, we can efficiently, and in parallel, learn a GP model for each of the components of the network; for the repressilator this represents one for each of the three mRNA and three protein species. A distribution over the causal network can then be constructed from the distributions over the regulators of each individual component, to yield at a complete statistical description of the system, capable of predicting behavior under novel conditions:

$$\mathbb{P}(\mathcal{G}|\mathbf{X}, \mathbf{Y}) = \prod_{i=1}^N \mathbb{P}_{\hat{\theta}(i)}(\mathcal{P}a(i)|\mathbf{Y}_i, \mathbf{X}). \quad (6)$$

This construction constitutes the Causal Structure Identification (CSI) algorithm outlined by Klemm et al. [26] and Penfold and Wild [27], which has been implemented as a MATLAB GUI [37] or as a cloud computing service via the CyVerse Discovery Environment [38].

2.4.1 Graphical Models

We can represent learning of the individual regulators for each gene by a *graphical model*, as shown in Fig. 5. Graphical models are a graphical representation of conditional independence relationships between a set of random variables in a probabilistic model. A random variable A is conditionally independent from B given C if $P(A, B|C) = P(A|C)P(B|C)$, or, equivalently, $P(A|B, C) = P(A|C)$. For example, in medical diagnosis, the probability of observing a symptom (A) may be conditionally independent of the results of a test for a disease (B), if we are given (i.e., we know) that the patient suffers from the disease (C). Using these conditional independences the joint probability of all the random variables in the model may be factored into a product of the conditional probabilities. For the medical example given above,

$$P(A, B, C) = P(C)P(A|C)P(B|C).$$

In a graphical model, each variable or hyperparameter in the model is represented by a *node* in the graph (circle), and the conditional independence relationships by directed arrows in the graph. We draw arrows from the variables on the right-hand side of the conditionals in a factorization of a joint probability distribution to the variables on the left-hand side, calling the right-hand side variables the *parents* of the left-hand side variables (called the *children*). The graphical model corresponding to the factorization above is shown in Fig. 5a.

2.5 Simulation with Gaussian Process Dynamical Systems

Gaussian process dynamical systems represent Bayesian nonparametric approaches to inferring a dynamical system; more than simply inferring a network structure, they provide a comprehensive framework for predicting (simulating) the behavior of the system under novel situations and perturbations. Since for each gene i , we have a set of GP models, we can make predictions about \dot{x}_i^* given a new vector of observations, denoted x^* , as a weighted mixture of Gaussian densities. A faster approximation would instead use the *maximum a posterior* (MAP) network, corresponding to a simulation from the single most likely dynamical system. Predictions would then correspond to a Gaussian distribution.

In Fig. 6 we illustrate the use of a GPDS to simulate expression time series, given starting concentrations for the molecular species not used to train the model. In Fig. 6a–c we show a phase space plot of the trajectories of three mRNA species m_1 , m_2 , and m_3 (black, green, and blue), simulated from a GPDS model with randomly initialized starting concentrations, trained using five time series (not shown). In Fig. 6d–f we show the same simulated time series, compared to the true values from the corresponding ODEs. In general, the time series simulated using the GPDS closely match the ODEs, demonstrating the GPDS's nonparametric ability to capture the kinetics of the system without the need to explicitly infer various parameters of the ODE. This ability to simulate from the system means that the approach can readily be used alongside approximate Bayesian computation (ABC) [14] to infer the behavior of the system under novel perturbations, for example, for gene knockouts, where partial observations have been made. Alternatively, these approaches could be used to simulate the behavior of the system following synthetic network rewiring, defined as the rearrangement of connections in a network, which can be achieved *in vivo* or *in vitro* by swapping regulatory regions, or the promoters of genes.

3 Hierarchical Gaussian Process Dynamical Systems

GPDS models are able to capture the full richness of complex dynamical systems provided that there exists sufficient training data for the GPs to capture the dynamics of each individual component

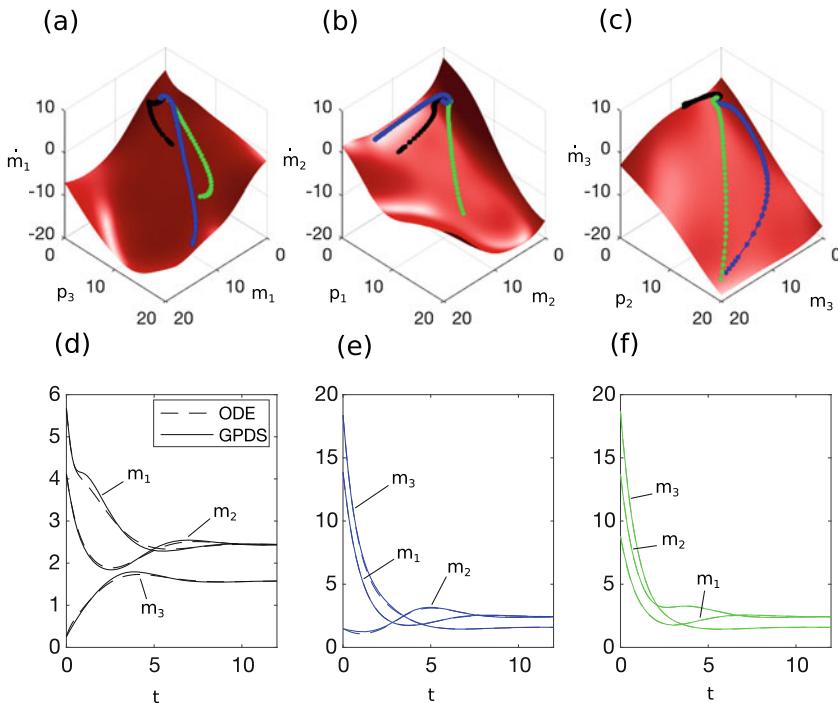


Fig. 6 GPDS can be used for simulation. Here we show the molecular kinetics of three mRNAs; m_1 , m_2 , and m_3 (a, b, and c, respectively), simulated from a GPDS model learnt using five other perturbed time series. Once we have a complete set of GPDS models, we can simulate the system from new starting concentrations not used for training. In (d–f) we show the same three simulated time series for the GPDS (solid lines) and the ODE model (dashed lines), with the trajectories shown in the corresponding *phase space* (a–c) using identical colors

of the system, and for the marginal likelihoods to correctly distinguish between putative sets of regulators [26, 27, 35]. The GPDS model outlined in Subheading 1, however, makes a number of assumptions that may not hold for all applications. Specifically, the model assumes that the kinetic parameters—or rather their proxy in the form of the posterior GP—are similar across all biological conditions (note that as we have a full posterior GP we can allow for some differences in the kinetics between time series), which is not necessarily the case if, for example, the mRNA or protein transcription or degradation rates can be altered under specific conditions [39, 40]. The model further assumes that the network structure itself is invariant to the biological condition; that is, the topology of the network remains the same in all samples s_1 to s_N . Again, this may not always be the case, and the effective network topology can vary due to changes in the epigenetic landscape, or the presence of inhibitors or cofactors, which can alter patterns of transcription factor binding. A notable motivating example can be found in early human embryogenesis, where the transcription factor SOX17 has been identified as a key regulator of both

endoderm fate [41–44] and primordial germ cell fate [44, 45], targeting a different, albeit partially overlapping, set of genes in the two cell types.

It is therefore of interest to develop approaches capable of inferring context-dependent networks, identifying both the similarities and differences in the GRNs associated with two or more subsets of data. A naive approach is to infer an independent GPDS for each subset of data, as indicated by the graphical models in Fig. 7a. Any similarities or differences in the networks can then be evaluated using appropriate statistical tests. However, in many cases alterations to the network structure may be limited to a few key nodes and edges, with the majority of the network unaltered across the conditions. The assumption of independence would needlessly dilute the available data, increasing the uncertainty and the propensity for overfitting. On the other hand, assuming the dynamical systems to be identical and combining the data together for inference is contrary to our initial assumptions and patently wrong.

Rather than treating the data independently or jointly, a more principled approach is to adopt a hierarchical modelling strategy [46–48], in which each dataset (or subset of data) is allowed its own generative GPDS, with the causal structure constrained to favor similar topologies across the networks. Hierarchical approaches have been developed for the GPDS model [47] that do exactly this, with a graphical model shown in Fig. 7b. As an illustrative example, we consider a dataset in which we have N time series observations, with the first $k - 1$ generated from a wild-type repressor system, and a second set of observations generated from a modified repressor system in which we have introduced an additional repressive link from m_1 to m_3 via the binding of p_1 to the promoter of gene 3 (Fig. 7c). We first split the observation data as follows:

$$\mathbf{X}^{(S_1)} = \begin{bmatrix} \mathbf{X}^{(1)} \\ \vdots \\ \mathbf{X}^{(k-1)} \end{bmatrix}, \quad \mathbf{Y}^{(S_1)} = \begin{bmatrix} \mathbf{Y}^{(1)}, \\ \vdots \\ \mathbf{Y}^{(k-1)} \end{bmatrix},$$

$$\mathbf{X}^{(S_2)} = \begin{bmatrix} \mathbf{X}^{(k)} \\ \vdots \\ \mathbf{X}^{(N)} \end{bmatrix}, \quad \mathbf{Y}^{(S_2)} = \begin{bmatrix} \mathbf{Y}^{(k)}, \\ \vdots \\ \mathbf{Y}^{(N)} \end{bmatrix}.$$

We must therefore infer two dynamical systems, and thus two causal network structures. Each node in each network is allowed its own (potentially unique) parental set structure, with the parental set for node i in network j denoted $\mathcal{P}a^{(j)}(i)$. As in previous models, a GP model can be fitted for node i in network j by first plotting $\mathbf{Y}_i^{(S_j)}$ as a function of $\mathbf{X}_{\mathcal{P}a^{(j)}(i)}^{(S_j)}$, allowing calculation of the

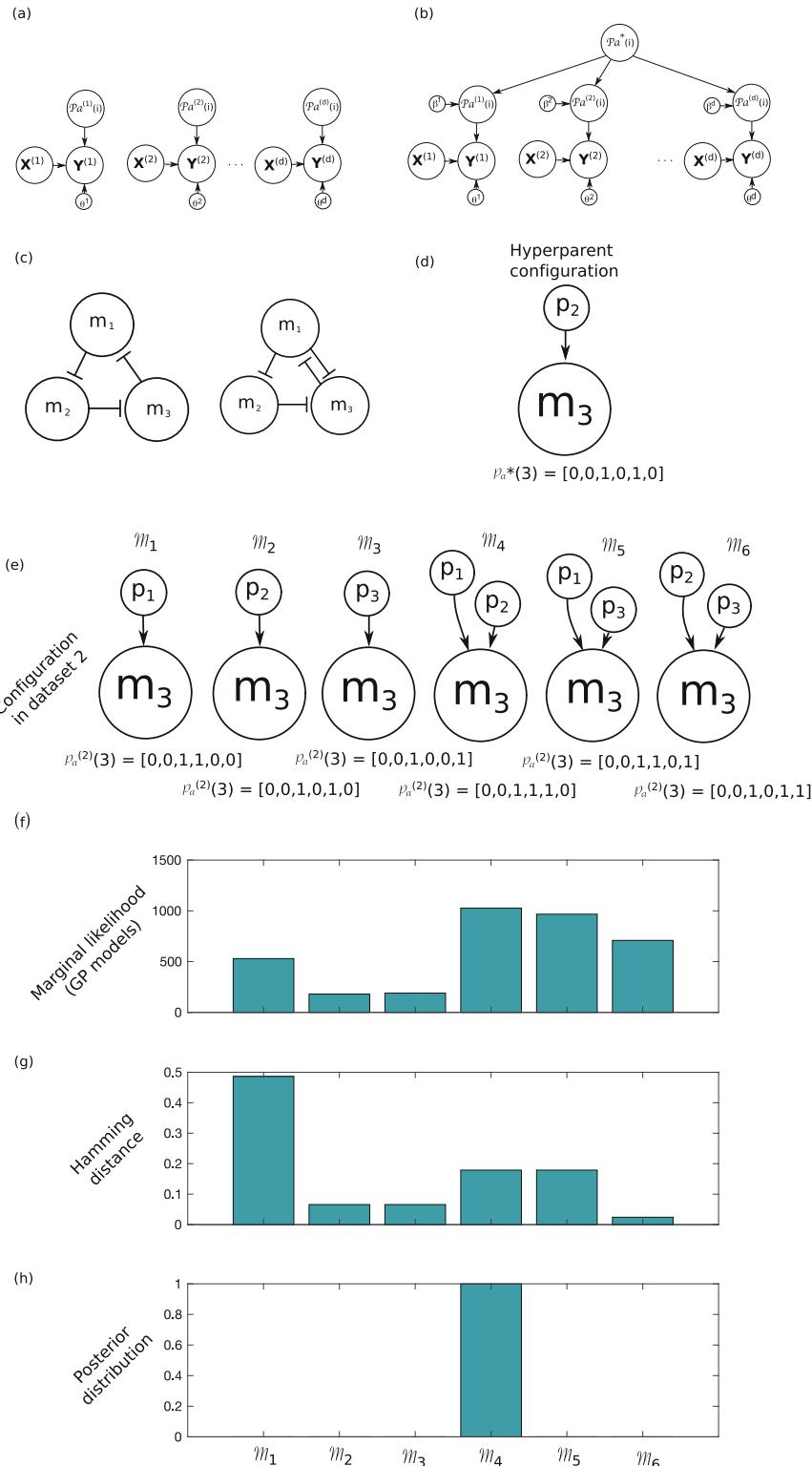


Fig. 7 (a) When there are multiple sets of time series, where the different sets are expected to have different network structures, we can use an independent GPDS for each of the datasets

likelihood of $\mathcal{P}a^{(j)}(i)$. Finally we assume the existence of a latent *hyperparent* set, $\mathcal{P}a^*$, usually defined over the same set of nodes as the individual parental sets. Conditional on $\mathcal{P}a^*$, the likelihood of $\mathcal{P}a^{(S_j)}(i)$ is weighted to favor similarity to the hyperparent set via a Gibbs distribution:

$$\mathbb{P}(\mathcal{P}a^{(j)}(i) | \mathcal{P}a^*(i), \beta) \propto \exp(-\beta^{(j)} ||\mathcal{P}a^{(j)}(i) - \mathcal{P}a^*(i)||). \quad (7)$$

where $|| \cdot ||$ denotes the Hamming distance. An intuitive way of thinking about this is to first consider the case where the hyperparents are fixed. In the repressilator system, for example, we might fix the hyperparent for node 3, $\mathcal{P}a^*(3) = \{1, 6\}$, which can be encoded in binary as $[0, 0, 1, 0, 1, 0]$ (Fig. 7d). The hyperparent thus induces a prior distribution over the parental sets of node 3 in networks 1 and 2 that favors molecular species 3 and 5 as the regulators of node 3. For network 2, we can consider a number of possible models for the regulation of node i (Fig. 7e), and using Bayes' rule, combine the likelihood from the GP models (Fig. 7f) with the prior distribution arising from Eq. (7) (Fig. 7g) to yield a posterior distribution over the models (Fig. 7h). In this case, while the prior assumption was for a simpler model, with only p_2 (note we have dropped the implicit self-regulation from m_3 for notational simplicity) regulating m_3 in the rewired repressilator network, the likelihood was sufficiently in favor of both p_1 and p_3 as dual regulators of m_3 , and allowed the correct regulation to be picked out in the posterior distribution (Fig. 7h). In much the same way, we can evaluate the posterior distribution for node i in the wild-type network.

Fig. 7 (continued) (b) When networks are expected to be similar, a hierarchical approach instead allows each dataset to have its own unique structure, albeit with the networks constrained to favor similarity. (c) An example of rewiring of the repressilator adds an extra link between gene 1 and gene 3. (d) Initially we have some guess for the hyperparent structure. (e) Conditional on our initial guess of the hyperparent structure, we can attempt search through different models representing putative sets of regulators in dataset 2. (f) Each putative set of regulators in dataset 2 again has a corresponding GP model for the dynamics, for which we can compute the marginal likelihood. (g) Each model is weighted by its distance from the hypernetwork, $\exp(\beta d(\mathcal{P}a, \mathcal{P}a^*))$, to yield the final posterior distribution for the parental set associated with dataset 2 (h). In this case, we favored model $\mathcal{P}a = \{p_1, p_2\}$, which corresponds to the true network. We can now fix this parental set in dataset 2, and attempt to update network 1 in the same way. Finally, the hyperparent can then be updated conditional on parental sets in datasets 1 and 2

For most applications, the hyperparent for any given node is not fixed, but additionally inferred. In the same way that we update node i in network j conditional on the hyperparent, we can, in turn, update the hyperparent conditional on the parental sets in the individual datasets, resulting in an iterative algorithm that updates the parental sets for each network and hypernetwork. In this case, the hyperparent no longer represents a prior distribution, but can instead be thought of in terms of a model average, or else in terms of feature selection, highlighting connections that exist in a large number of the individual networks.

Within this model the parameter $\beta^{(j)}(i)$ controls the level of influence that the hyperparent exerts on the individual parental sets and vice versa. If we set $\beta^{(j)}(i) = 0$, the hyperparent has no influence on the parental sets of node i in dataset j , and the conditional posterior distribution is determined completely by the likelihoods from the GP models—when all β parameters are set to 0, the hierarchical model becomes a set of independent GPDS models (Fig. 7a). Likewise, when $\beta^{(j)}(i)$ is large, the hyperparent becomes increasingly important, and for sufficiently large values, the parental set in the individual datasets will be forced to be identical to the hyperparent set.

The ability to automatically tune the β parameters within the model is therefore a significant advantage, allowing different datasets to exert more or less influence on the other network structures depending on the amount of information they share. This is particularly important where, for example, one of the datasets has been corrupted, or has been generated by a network with vastly different topology to the other datasets. In this case, without tuning β , all parental sets would contribute equally to the hyperparent and vice versa, and the hypernetwork would therefore be less suitable for doing feature selection; on the other hand, by down weighting the impact on the corrupted or vastly different dataset, allows the hypernetwork to be primarily influenced by the remaining datasets, highlighting the commonalities. This advantage has been demonstrated in earlier work [47], which demonstrated the superiority of hierarchical modelling compared to independent inference.

Such approaches had proven useful for identifying context-dependent differences in plant transcriptional response to pathogens, allowing the joint leveraging of data from multiple biotic and abiotic stresses alongside yeast-one-hybrid networks [47, 49], with software available within the CSI MATLAB GUI [37] or via the CyVerse Discovery Environment [38].

4 Orthologous Gaussian Process Dynamical Systems

While the hierarchical GPDS outlined in Subheading 3 offers a principled approach to learning the structure and kinetics of closely related dynamical systems, it is primarily intended for inference when nodes in the individual networks corresponded in a 1:1 fashion. This is primarily the case when inferring networks in the same species under different conditions. Another key challenge is to infer dynamical systems for closely related species where nodes do not necessarily correspond 1:1 due to events such as gene or chromosome duplication, hybridization, or other evolutionary changes. Despite such changes, the gene regulatory networks (GRNs) of related species are still expected to share common topological features with one another by virtue of a shared ancestry, and there is expected to be a good deal of mutually informative information between the species (Fig. 8a). Consequently, the joint inference of GRNs from gene expression datasets collected from different species should result in better overall accuracy in the inferred networks, due to the increased amount of data from which to learn the shared components of the networks[50–53]. Likewise, the leveraging of gene regulatory networks that have been experimentally verified in one species into a related species should also improve the accuracy of the networks in the second species (Fig. 8b). Both the tasks of joint inference and network leveraging are related and require the leveraging of data and information between species, either in the form of multiple time series gene expression datasets, as is the case for joint inference (JI), or a combination of time series gene expression data with experimentally verified networks during network leveraging (NL). Due to the increasing availability of heterogeneous datasets in a range of species, flexible approaches to network inference that can be adapted to both JI and NL tasks should be particularly useful and would allow vast amounts of data and information available in model organisms to be translated into more complex or medically or economically relevant ones.

A number of approaches for JI and NL for GPDS were developed in Penfold et al. [28]. The first approach is based on the hierarchical models outlined in the previous section (Fig. 8b). In this case, the probability of a network associated with species j , denoted $\mathcal{G}^{(j)}$, is determined by the GP likelihood, weighted by its similarity to the hypernetwork, \mathcal{G}^* , via a Gibbs distribution:

$$\mathbb{P}(\mathcal{G}^{(j)} | \mathcal{G}^*, \beta^{(j)}) \propto \exp(-\beta^{(j)}(\mathcal{E}_0 - K(\mathcal{G}^{(j)}, \mathcal{G}^*))),$$

where $K(\cdot, \cdot)$ is a measure of similarity between two graphs, and \mathcal{E}_0 is a constant representing the maximum possible similarity. Crucially, we note a key difference compared to earlier hierarchical

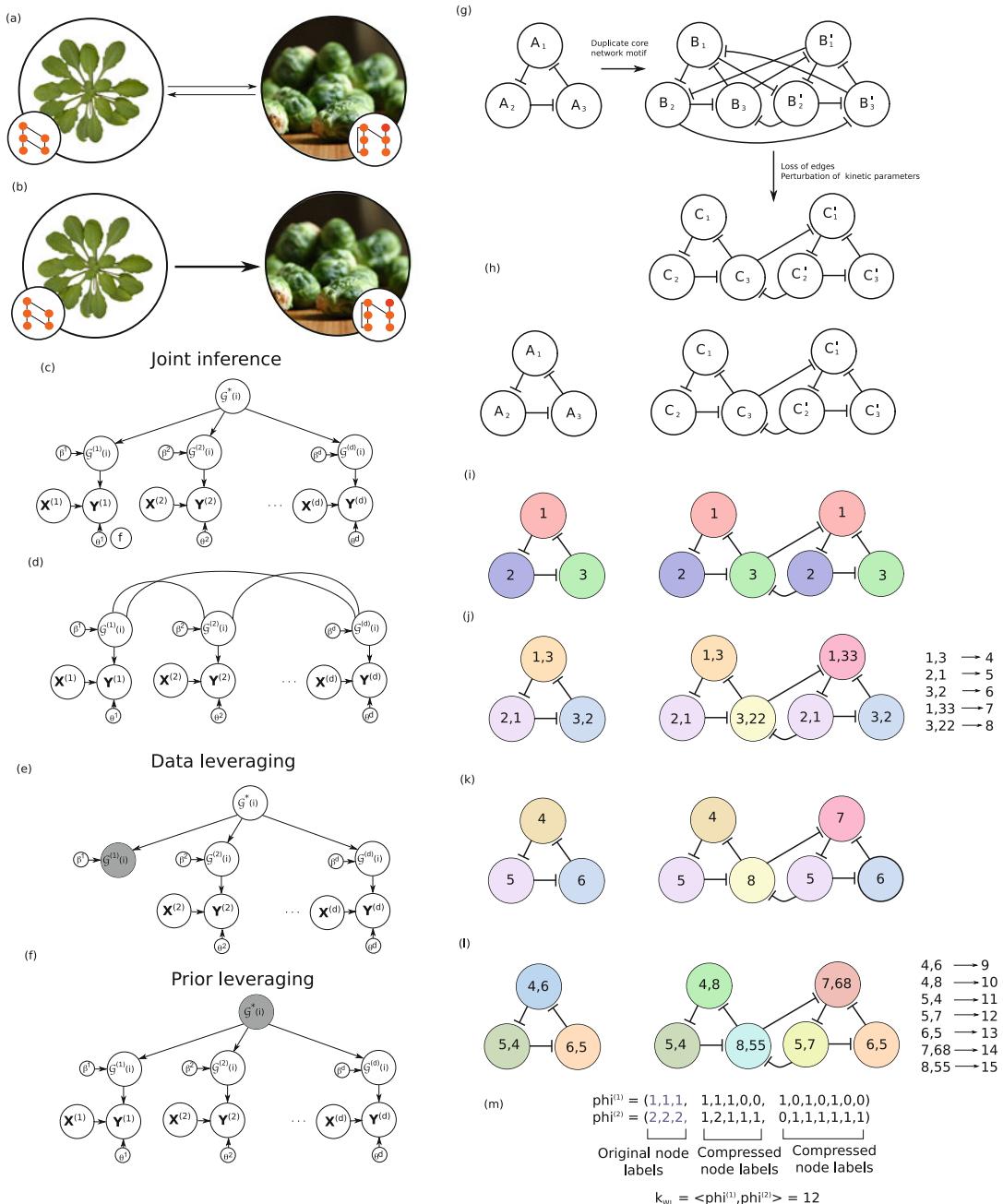


Fig. 8 The orthologous GPDS can be used for a number of applications. **(a)** This includes joint inference of networks in multiple species, where data is available in both species. Inference can be achieved similarly to other hierarchical models, by inference of a constraining hypernetwork **(b)** or by allowing the networks to directly constrain one another **(c)**. Earlier studies show that constraining networks via a hypernetwork is better. **(d)** OGPDS can also be used for leveraging information between species, this includes **(e)** leveraging information about networks derived in one species into a related species and **(f)** leveraging joint prior information into all species **(f)**. In both cases, the networks to be leveraged are fixed, and inference proceeds as normal. **(g)** As an illustrative example, of the WL kernel, we can assume duplication of the repressilator system in species

models; network similarity is computed between networks as a whole, rather than component-wise among the parental sets. This distinction provides a basis for dealing with networks with duplicated or novel nodes, as we can use *graph kernels* to evaluate the similarity between any two graphs. Informally, graph kernels represent a function of two graphs that quantifies their similarity [54], and fall broadly into three categories: (1) those based on similarities of the walks and paths in the respective graphs; (2) those based on similarities in limited-sized subgraphs, and (3) those based on sub-tree patterns. Previous studies evaluated a number of such graph kernels for both joint inference and the leveraging of information between species, and found that the Weisfeiler-Lehman (WL) kernel offered the best overall performance [28]. The WL kernel evaluates the similarity between two graphs or more graphs by comparing sub-tree patterns [54].

To illustrate the WL kernel, we first return to the repressilator system. In this case, we assume a three component network in species *A* where the three genes A_1 , A_2 , and A_3 mutually repress one another via a repressilator motif (Fig. 8g). A second, intermediate species, *B*, arose following duplication of the core repressilator motif in species *A*, resulting in six genes B_1 , B_2 , B_3 , B_4 , B_5 , and B_6 , which form a complex repressive motif. Finally, a third species, *C*, emerges following evolutionary rewiring of the network in species *B*. As an illustrative example, we can compare the network similarity of species *A* with that of species *C* (Fig. 8h). Using the WL kernel, we first assign a label to each node in the network, with nodes that are paralogues or orthologues of one another assigned the same label (Fig. 8i).

Each node in the graph is initially relabelled using the following convention: the updated label contains the original node label followed by an ordered list of labels based on which nodes are directly upstream. For example, the node for gene A_2 in network 1 was originally assigned a label 2. As this node is directly downstream of a node with label 1, it will be relabelled 2, 1 (Fig. 8j), and updated labels compressed into a new unique set of node labels

Fig. 8 (continued) *A* results in a network of six genes, B_1 , B_2 , B_3 , B'_1 , B'_2 , B'_3 , with subsequent evolutionary loss of links and perturbation of dynamics leading to a new species, *C*. **(h)** Here we are interested in comparing the similarity of the networks in the two species. To do so, each node is assigned a label, with paralogues or orthologues assigned an identical label. **(i)** Nodes are relabelled according to which nodes they're regulated by **(j)**, resulting in a new set of node labels, but retaining the network structure. **(k)** Relabelled nodes are assigned compressed node labels which can, once again, be updated according to which nodes are upstream **(l)**. Finally, after a predetermined number of iterations, a vector containing the counts of each compressed node labels can be constructed for each network, and the inner product used to evaluate similarity **(m)**

(Fig. 8k). This process can be repeated (Fig. 8l). For example, for species A , the initial graph, denoted $\mathcal{G}_0(1) = (V, E, l)$, may be relabelled according to incoming connections to generate a new graph $\mathcal{G}_1(1) = (V, E, l_1)$. Node A_1 , for example, which is regulated by A_3 , is initially assigned label 1, and subsequently relabelled 1, 3. After each round of relabelling, the nodes and edges are conserved, but the updated labels propagate information about the immediate neighborhood of each node. The b th round of relabelling for graph i generates the graph $\mathcal{G}_b(i) = (V, E, l_b)$. At each step of the relabelling, we can construct a vector for each graph containing the count of each (compressed) node label (Fig. 8m), with $K(\cdot, \cdot)$ evaluated between two graphs as the inner product.

The orthologous GPDS model was initially implemented in MATLAB [28], and used to leverage *S. cerevisiae* networks to inform inference in *S. pombe*, where it highlighted a novel role for Gas1 (SPAC19B12.02c), a 1,3-beta-glucanosyltransferase, in cell cycle regulation [28]. Software for implementing inference in orthologous nonlinear dynamical systems is available via the CyVerse Discovery Environment [38].

5 Inference in Spatiotemporal Systems

Many spatiotemporal models of embryogenesis use detailed biophysical descriptions based on partial differential equations, such as reaction-diffusion models [55–57]. Inference based on these approaches can be computationally intensive, and does not scale well to complex or understudied organisms, or systems with many molecular species. Bayesian nonparametric approaches based on GPDS can naturally be used to capture the regulatory structure and spatiotemporal dynamics of embryogenesis, jointly leveraging the observations over multiple spatial positions in a structured way.

In a simple one-dimensional representation of embryo development, we assume that measurements of gene or protein expression levels are made at various points along a spatial axis, for example, along the *anterior-posterior* axis, with $x_i^{(j)}(t)$ used to denote the expression of molecular species i in cell j at time t , with corresponding derivative $\dot{x}_i^{(j)}(t)$. Component i then evolves as:

$$\dot{x}_i^{(j)}(t) = f_i(\underbrace{\mathbf{x}_{\mathcal{P}a(i)}^{(j)}(t - \tau_1)}_{\text{Local production (regulation by parents)}}, \underbrace{\mathbf{x}_i^{(j-1)}(t - \tau_2), \mathbf{x}_i^{(j+1)}(t - \tau_2)}_{\text{Net diffusion from adjacent cells}}), \quad (8)$$

where τ_1 and τ_2 represent characteristic time scales denoting transcriptional or translational and diffusion delays, respectively. This model makes a number of clear assumptions: first, it assumes that the concentrations of the individual molecular species within

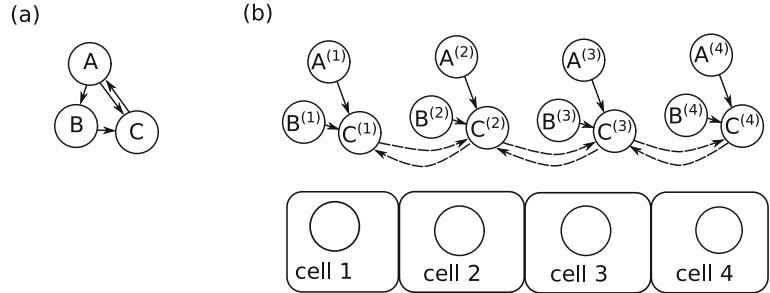


Fig. 9 (a) A modified repressilator network, with additional repression from gene A to C . (b) The core network can function in all cells within the developing embryo, with molecular species potentially diffusing between cells. For illustration purposes, here we show the parental sets associated with gene C

an individual cell are reasonably well mixed, allowing us to replace a spatially continuous description with a coupled set of ODE models, one for each of the individual cells (or spatial points of measurement); second, the model assumes that, within cell j , the rate of change of molecular species i with respect to time is determined by the local production of the protein or mRNA within the cell from transcription or translation, and by the net diffusion of the molecular species from the two adjacent cells. We can illustrate a spatiotemporal model using a modified repressilator motif. In Fig. 9a we indicate the modified repressilator composed of three genes that mutually repress one another, and indicate the causal parental sets for gene C in a four cell embryo (Fig. 9b).

As with earlier GPDS models, if the regulators of mRNA or protein i , denoted $\mathcal{P}a(i) = \{pa_1, \dots, pa_n\}$, are known explicitly then, for a given cell j , we are interested in explaining the observation $\mathbf{y}_i^{(j)} = (\dot{x}_i^{(j)}(t_1), \dots, \dot{x}_i^{(j)}(t_T))^\top$, as a function of the inputs:

$$\mathbf{x}_i^{(j)} = \begin{bmatrix} x_{pa_1}^{(j)}(t_1 - \tau_1), & \dots, & x_{pa_n}^{(j)}(t_1 - \tau_1), & x_i^{(j-1)}(t_1 - \tau_2), & x_i(j+1)(t_1 - \tau_2) \\ \vdots & & \ddots & & \vdots \\ x_{pa_1}^{(j)}(t_T - \tau_1), & \dots, & x_{pa_n}^{(j)}(t_T - \tau_1), & x_i^{(j-1)}(t_T - \tau_2), & x_i^{(j+1)}(t_T - \tau_2) \end{bmatrix}.$$

Note that here the input matrix is composed of two parts: in blue we indicate the expression levels of the parent regulators, which control the rate of production of molecular species i , with the terms in red denoting the concentration of molecular species i in the adjacent cells.

It is unlikely that observation of a single time series in a single cell will yield sufficient information to capture the full range of kinetics possible in this system. However, if we can assume the molecular parameters and network topology are invariant to position, we can stack the measurements in the c cells:

$$\mathbf{X}_i = \begin{bmatrix} \mathbf{X}_i^{(1)} \\ \vdots \\ \mathbf{X}_i^{(c)} \end{bmatrix}, \quad \mathbf{y}_i = \begin{bmatrix} \mathbf{y}_i^{(1)} \\ \vdots \\ \mathbf{y}_i^{(c)} \end{bmatrix}. \quad (9)$$

If we have multiple perturbations, then these too can be stacked, provided that we can assume invariance of the network and molecular kinetics to the perturbation. Since it is not clear what form the unknown function, $f_i(\cdot)$, should take, we can again assume a GP prior, and capture the molecular kinetics directly from the observational data. Likewise, when the structure of network remains unknown, we can employ the same combinatorial search outlined previously, using Bayes' rule to infer a posterior distribution over the parental sets.

The spatiotemporal GPDS (STGPDS) model can naturally be extended to two-dimensional and three-dimensional systems, with the trivial change that diffusion occurs from an increased number of adjacent cells, rather than the two we considered in the one-dimensional case.

5.1 Segmentation Gene Networks in *Drosophila melanogaster*

To demonstrate inference with a spatiotemporal GPDS we have inferred the regulatory network and molecular kinetics involved in *Drosophila melanogaster* embryo segmentation, a crucial process in the establishment of the *Drosophila* body plan. To do so we used data available from the FlyEx database [58, 59], which contained measurements of protein expression based on fluorescence at ~ 2300 locations along the anterior-posterior and dorsal-ventral axes over 5 time points. This included measurements of key proteins bicoid (bcd), caudal (cad), kruppel (Kr), giant (gt), hunchback (hb), knirps (kni), and tailless (tll). A STGPDS model was inferred using 150 data points randomly selected from the 2300 stacked observations. In Fig. 10a, b we compare the inferred network with a previously elucidated network [60]. In total, the *maximum a posteriori* network—that is, the single best network—captured 11 connections that appeared in the literature network, identifying the importance of cad, tll, and bcd as drivers of the expression patterns, as well as interactions between kni and Kr. While a number of connections were missing, these preliminary results suggest that a significant proportion of the network can be recovered using a subset of the data.

5.2 Simulation of *Drosophila* Segmentation Network

As previously noted, GPDS are well suited to predicting the molecular dynamics of a system. In Fig. 10c we plot the simulated expression data at all ~ 2300 spatial locations at time point 3, for hb, Kr, and gt compared to the true expression levels along the anterior-posterior axis, while in Fig. 10d we indicate the inferred and observed expression pattern along the dorsal-ventral axis. In general the true dynamics are reasonably well captured for the

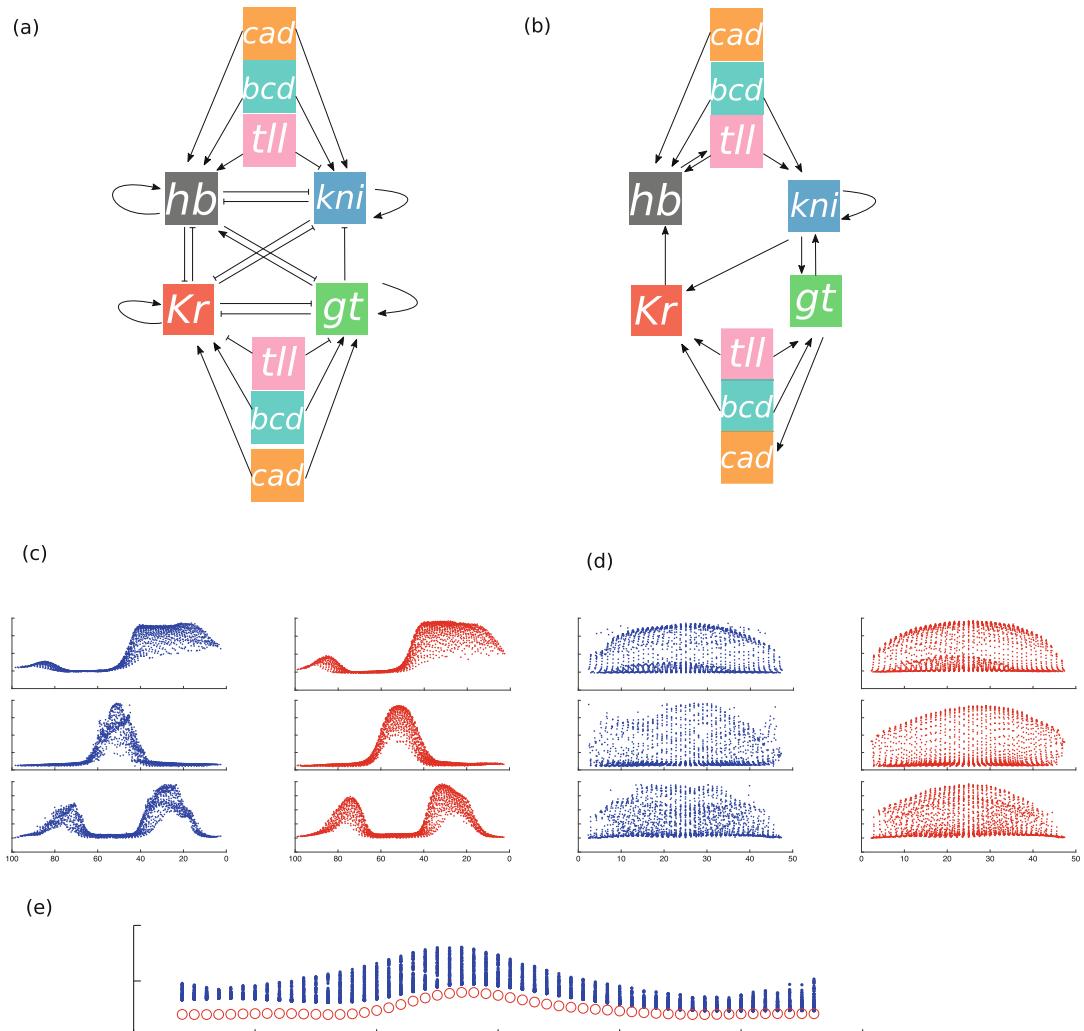


Fig. 10 (a) *Drosophila* segmentation network from Kozlov et al. [60]. (b) The inferred *Drosophila* network using a STGPDS with 150 randomly selected training data points from a set of 2D measurements at 5 time points. (c) Inferred spatial profiles for *hb*, *Kr*, and *gt* at time point 3 (blue) versus observed profiles (red) along the anterior-posterior axis. (d) Inferred spatial profiles for *hb*, *Kr*, and *gt* at time point 3 (blue) versus observed profiles (red) along the dorsal-ventral axis. (e) Inferred *kni* profiles along the anterior-posterior axis using partial observations of *hb* and *gt* in a *Kr* mutant line (blue) versus the observed values (red)

majority of proteins. While network reconstruction was not in perfect agreement with the literature network, it appeared that sufficient information was captured to allow good prediction of the wild-type time series.

A key goal of GPDS modelling is the prediction of expression dynamics under novel conditions, for example, following knockout (KO) of key genes. To demonstrate this we use the STGPDS model to predict protein expression for *kni* using partial observations of *hb* and *gt* in a *Kr* mutant line. Specifically, we note that protein

expression levels for the genes hb, gt, and kni in the knockout mutant line were available at 3 time points. We assume that the starting expression levels for cad, bcd, and tll were identical to that of wild-type embryos due to a lack of regulation by hb of these three proteins in the literature network (Fig. 10a). The expression profiles for hb and gt at time point 1 were set to their observed expression profiles in the Kr mutant, while the expression profile for gt was assumed to be a constant, random level across the cells. The initial expression profile for hkb was taken as a random sample from a Gaussian process. The STGPDS model from Subheading 5.1 was used to simulate sample expression profiles for all genes at time points 2 and 3, which were accepted or rejected based on how well the simulated trajectories for hb and gt matched the observed values. In Fig. 10e we show the inferred expression profile for kni at time point 3 versus the true profile in the mutant line. We note that even though the model had not seen the expression pattern of kni in the mutant line, it was nonetheless able to predict its behavior using the wild-type dynamical system, and observations of other genes in the mutant line.

6 Outlook

GPDS models represent flexible approaches for the inference of nonlinear dynamical systems, capable of inferring the network structure and providing unique insights into complex biological processes. Crucially, by leveraging multiple datasets, these methods are able to capture the molecular kinetics of biological systems in a way that generalizes and allows predictions under novel conditions. In this chapter we have shown one such use, predicting the effect of gene knockouts in *Drosophila* embryogenesis.

As well as obvious applications in systems and developmental biology, the ability to predict dynamics under novel perturbations is of particular interest in synthetic biology, an emerging discipline tasked with constructing novel biological systems for a particular purpose by using existing underlying biological parts. Its principles are based on engineering and applications range from designing and constructing genetic circuits, RNA, proteins, or even viruses, cells, and microbes for therapeutic, environmental, or industrial purposes [61, 62] or to gain insights into fundamental principles of biology.

While re-purposing networks by rewiring has proven to be a viable approach to increasing recombinant protein production [63], enhancing tolerance to environmental conditions [64], or significantly increasing the efficacy of chemotherapy [65], the applications have been mostly based on exhaustive searches of network structures. The use of GPDS modelling for the rational redesign of networks should greatly speed up the design process and reduce the costs of biological implementation.

Acknowledgements

CAP is supported by the Wellcome Trust (grant 083089/Z/07/Z). IG is supported by EPSRC/BBSRC research grant EP/L016494/1. AS is supported by a 4-year Wellcome Trust PhD Scholarship and Cambridge International Trust Scholarship. DLW acknowledges support from the Engineering and Physical Science Research Council (grant EP/R014337/1).

CAP, IG, and AS BBSRC-EPSRC funded OpenPlant Synthetic Biology Research Centre (BB/L014130/1) through the Open-Plant Fund scheme. CAP and AS also thank M. Azim Surani for his support.

References

1. Kholodenko BN, Kiyatkin A, Bruggeman FJ, Sontag E, Westerhoff HV, Hoek JB (2002) Untangling the wires: a strategy to trace functional interactions in signaling and gene networks. *Proc Natl Acad Sci* 99(20):12841–12846
2. Elowitz MB, Leibler S (2000) A synthetic oscillatory network of transcriptional regulators. *Nature* 403(6767):335–338
3. Gardner TS, Cantor CR, Collins JJ (2000) Construction of a genetic toggle switch in *Escherichia coli*. *Nature* 403(6767):339–342
4. Zak DE, Gonye GE, Schwaber JS, Doyle FJ (2003) Importance of input perturbations and stochastic gene expression in the reverse engineering of genetic regulatory networks: insights from an identifiability analysis of an in silico network. *Genome Res* 13(11):2396–2405
5. Locke J, Millar A, Turner M (2005) Modelling genetic networks with noisy and varied experimental data: the circadian clock in *Arabidopsis thaliana*. *J Theor Biol* 234(3):383–393
6. Pokhilko A, Mas P, Millar AJ (2013) Modelling the widespread effects of *toc1* signalling on the plant circadian clock and its outputs. *BMC Syst Biol* 7(1):23
7. Fogelmark K, Troein C (2014) Rethinking transcriptional activation in the *Arabidopsis* circadian clock. *PLoS Comput Biol* 10(7):e1003705
8. Domijan M, Rand DA (2015) Using constraints and their value for optimization of large ode systems. *J R Soc Interface* 12(104):20141303
9. De Caluwé J, Xiao Q, Hermans C, Verbruggen N, Leloup JC, Gonze D (2016) A compact model for the complex plant circadian clock. *Front Plant Sci* 7:74
10. Ashall L, Horton CA, Nelson DE, Paszek P, Harper CV, Sillitoe K, Ryan S, Spiller DG, Unitt JF, Broomhead DS et al (2009) Pulsatile stimulation determines timing and specificity of NF- κ B-dependent transcription. *Science* 324(5924):242–246
11. Wang Y, Paszek P, Horton CA, Yue H, White MR, Kell DB, Muldoon MR, Broomhead DS (2012) A systematic survey of the response of a model nf- κ b signalling pathway to tnf α stimulation. *J Theor Biol* 297:137–147
12. Jonak K, Kurpas M, Szoltysek K, Janus P, Abramowicz A, Puszynski K (2016) A novel mathematical model of atm/p53/nf- κ b pathways points to the importance of the DDR switch-off mechanisms. *BMC Syst Biol* 10(1):75
13. Calderhead B, Girolami M, Lawrence ND (2009) Accelerating Bayesian inference over nonlinear differential equations with Gaussian processes. In: Advances in neural information processing systems, pp 217–224
14. Toni T, Welch D, Strelkowa N, Ipsen A, Stumpf MP (2009) Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems. *J R Soc Interface* 6(31):187–202
15. Liepe J, Kirk P, Filippi S, Toni T, Barnes CP, Stumpf MP (2014) A framework for parameter estimation and model selection from experimental data in systems biology using approximate Bayesian computation. *Nat Protoc* 9(2):439–456

16. Beaumont MA, Rannala B (2004) The Bayesian revolution in genetics. *Nat Rev Genet* 5(4):251–261
17. Hjort N, Holmes C, Müller P, Walker S (eds) (2010) Bayesian nonparametrics. Cambridge University Press, Cambridge
18. Murray-Smith R, Johansen TA, Shorten R (1999) On transient dynamics, off-equilibrium behaviour and identification in blended multiple model structures. In: 1999 European control conference (ECC). IEEE, Piscataway, pp 3569–3574
19. Murray-Smith R, Girard A (2001) Gaussian process priors with ARMA noise models. In: Irish signals and systems conference, Maynooth, pp 147–152
20. Girard A, Rasmussen CE, Candela JQ, Murray-Smith R (2003) Gaussian process priors with uncertain inputs application to multiple-step ahead time series forecasting. In: Advances in neural information processing systems, pp 545–552
21. Leithead W, Solak E, Leith D (2003) Direct identification of nonlinear structure using Gaussian process prior models. In: European control conference (ECC), 2003. IEEE, Piscataway, pp 2565–2570
22. Sbarbaro D, Murray-Smith R (2005) Self-tuning control of non-linear systems using Gaussian process prior models. In: Switching and learning in feedback systems. Springer, Berlin, pp 140–157
23. Cunningham J, Ghahramani Z, Rasmussen CE (2012) Gaussian processes for time-marked time-series data. In: International conference on artificial intelligence and statistics, pp 255–263
24. Frigola R, Lindsten F, Schön TB, Rasmussen CE (2014) Identification of Gaussian process state-space models with particle stochastic approximation EM. *IFAC Proc Vol* 47(3):4097–4102
25. Frigola R, Chen Y, Rasmussen CE (2014) Variational Gaussian process state-space models. In: Advances in neural information processing systems, pp 3680–3688
26. Klemm S et al (2008) Causal structure identification in nonlinear dynamical systems. Department of Engineering, University of Cambridge, Cambridge
27. Penfold CA, Wild DL (2011) How to infer gene networks from expression profiles, revisited. *Interface Focus* 1(6):857–870
28. Penfold CA, Millar JB, Wild DL (2015) Inferring orthologous gene regulatory networks using interspecies data fusion. *Bioinformatics* 31(12):i97–i105
29. Williams CK, Rasmussen CE (2006) Gaussian processes for machine learning, vol 2. MIT Press, Cambridge
30. Lloyd JR, Duvenaud D, Grosse R, Tenenbaum JB, Ghahramani Z (2014) Automatic construction and natural-language description of nonparametric regression models. Preprint. arXiv:14024304
31. Yang J, Penfold CA, Grant MR, Rattray M (2016) Inferring the perturbation time from biological time course data. *Bioinformatics* 32:2956–2964
32. Penfold CA, Sybirna A, Reid J, Huang Y, Wernisch L, Grant M, Ghahramani Z, Surani MA (2017) Nonparametric Bayesian inference of transcriptional branching and recombination identifies regulators of early human germ cell development. *bioRxiv* p 167684
33. Penfold CA, Sybirna A, Reid J, Huang Y, Wernisch L, Ghahramani Z, Grant M, Surani MA (2018) Branch-recombinant Gaussian processes for analysis of perturbations in biological time series. *Bioinformatics*, 34(17):i1005–i1013
34. Boukouvalas, Alexis, Hensman J, Rattray M (2018) BGP: identifying gene-specific branching dynamics from single-cell data with a branching Gaussian process. *Genome biology* 19:1:65
35. Äijö T, Lähdesmäki H (2009) Learning gene regulatory networks from gene expression measurements using non-parametric molecular kinetics. *Bioinformatics* 25(22):2937–2944
36. Solak E, Murray-Smith R, Leith WE, Leith DJ, Rasmussen CE (2003) Derivative observations in Gaussian process models of dynamic systems. In: Advances in neural information processing systems, pp 1057–1064
37. Penfold CA, Shifaz A, Brown PE, Nicholson A, Wild DL (2015) Csi: a nonparametric Bayesian approach to network inference from multiple perturbed time series gene expression data. *Stat Appl Genet Mol Biol* 14(3):307–310
38. Polanski K, Gao B, Mason SA, Brown P, Ott S, Denby KJ, Wild DL (2017) Bringing numerous methods for expression and promoter analysis to a public cloud computing service. *Bioinformatics* 1:3

39. Rabani M, Levin JZ, Fan L, Adiconis X, Raychowdhury R, Garber M, Gnrke A, Nusbaum C, Hacohen N, Friedman N et al (2011) Metabolic labeling of RNA uncovers principles of RNA production and degradation dynamics in mammalian cells. *Nat Biotechnol* 29(5):436–442
40. Li L, Nelson C, Fenske R, Trösch J, Pruzinská A, Millar AH, Huang S (2017) Changes in specific protein degradation rates in *Arabidopsis thaliana* reveal multiple roles of lon1 in mitochondrial protein homeostasis. *Plant J* 89(3):458–471
41. D'Amour KA, Agulnick AD, Eliazer S, Kelly OG, Kroon E, Baetge EE (2005) Efficient differentiation of human embryonic stem cells to definitive endoderm. *Nat Biotechnol* 23(12):1534–1541
42. Wang P, Rodriguez RT, Wang J, Ghodasara A, Kim SK (2011) Targeting SOX17 in human embryonic stem cells creates unique strategies for isolating and analyzing developing endoderm. *Cell Stem Cell* 8(3):335–346
43. Viotti M, Nowotschin S, Hadjantonakis AK (2014) SOX17 links gut endoderm morphogenesis and germ layer segregation. *Nat Cell Biol* 16(12):1146–1156
44. Kobayashi T, Zhang H, Tang WW, Irie N, Withey S, Klisch D, Sybirna A, Dietmann S, Contreras DA, Webb R et al (2017) Principles of early human development and germ cell program from conserved model systems. *Nature* 546:416–420
45. Irie N, Weinberger L, Tang WW, Kobayashi T, Viukov S, Manor YS, Dietmann S, Hanna JH, Surani MA (2015) SOX17 is a critical specifier of human primordial germ cell fate. *Cell* 160(1):253–268
46. Werhli AV, Husmeier D (2008) Gene regulatory network reconstruction by Bayesian integration of prior knowledge and/or different experimental conditions. *J Bioinform Comput Biol* 6(03):543–572
47. Penfold CA, Buchanan-Wollaston V, Denby KJ, Wild DL (2012) Nonparametric Bayesian inference for perturbed and orthologous gene regulatory networks. *Bioinformatics* 28(12):i233–i241
48. Oates CJ, Korkola J, Gray JW, Mukherjee S et al (2014) Joint estimation of multiple related biological networks. *Ann Appl Stat* 8(3):1892–1919
49. Hickman R, Hill C, Penfold CA, Breeze E, Bowden L, Moore JD, Zhang P, Jackson A, Cooke E, Bewicke-Copley F et al (2013) A local regulatory network around three NAC transcription factors in stress responses and senescence in *Arabidopsis* leaves. *Plant J* 75(1):26–39
50. Kashima H, Yamanishi Y, Kato T, Sugiyama M, Tsuda K (2009) Simultaneous inference of biological networks of multiple species from genome-wide data and evolutionary information: a semi-supervised approach. *Bioinformatics* 25(22):2962–2968
51. Gholami AM, Fellenberg K (2010) Cross-species common regulatory network inference without requirement for prior gene affiliation. *Bioinformatics* 26(8):1082–1090
52. Zhang X, Moret BM (2010) Refining transcriptional regulatory networks using network evolutionary models and gene histories. *Algorithms Mol Biol* 5(1):1
53. Joshi A, Beck Y, Michoel T (2015) Multi-species network inference improves gene regulatory network reconstruction for early embryonic development in *Drosophila*. *J Comput Biol* 22(4):253–265
54. Shervashidze N, Schweitzer P, Leeuwen EJV, Mehlhorn K, Borgwardt KM (2011) Weisfeiler-Lehman graph kernels. *J Mach Learn Res* 12(Sep):2539–2561
55. Turing A (1952) The chemical basis of morphogenesis. *Phil Trans R Soc Lond B* 237:37–72
56. Kondo S, Miura T (2010) Reaction-diffusion model as a framework for understanding biological pattern formation. *Science* 329(5999):1616–1620
57. Müller P, Rogers KW, Jordan BM, Lee JS, Robson D, Ramanathan S, Schier AF (2012) Differential diffusivity of nodal and lefty underlies a reaction-diffusion patterning system. *Science* 336(6082):721–724
58. Pisarev A, Poustelnikova E, Samsonova M, Reinitz J (2008) Flyex, the quantitative atlas on segmentation gene expression at cellular resolution. *Nucleic Acids Res* 37(Suppl 1):D560–D566
59. Poustelnikova E, Pisarev A, Blagov M, Samsonova M, Reinitz J (2004) A database for management of gene expression data in situ. *Bioinformatics* 20(14):2212–2221
60. Kozlov K, Gursky V, Kulakovskiy I, Samsonova M (2014) Sequence-based model of gap gene regulatory network. *BMC Genomics* 15(12):S6

61. Purnick PE, Weiss R (2009) The second wave of synthetic biology: from modules to systems. *Nat Rev Mol Cell Biol* 10(6):410–422
62. Khalil AS, Collins JJ (2010) Synthetic biology: applications come of age. *Nat Rev Genet* 11(5):367–379
63. Windram OP, Rodrigues RT, Lee S, Haines M, Bayer TS (2017) Engineering microbial phenotypes through rewiring of genetic networks. *Nucleic Acids Res* 45(8):4984–4993
64. Isalan M, Lemerle C, Michalodimitrakis K, Horn C, Beltrao P, Rainieri E, Garriga-Canut M, Serrano L (2008) Evolvability and hierarchy in rewired bacterial gene networks. *Nature* 452(7189):840–845
65. Lee MJ, Albert SY, Gardino AK, Heijink AM, Sorger PK, MacBeath G, Yaffe MB (2012) Sequential application of anticancer drugs enhances cell death by rewiring apoptotic signaling networks. *Cell* 149(4):780–794



Chapter 12

Unsupervised GRN Ensemble

Pau Bellot, Philippe Salembier, Ngoc C. Pham, and Patrick E. Meyer

Abstract

Inferring gene regulatory networks from expression data is a very challenging problem that has raised the interest of the scientific community. Different algorithms have been proposed to try to solve this issue, but it has been shown that 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. In this chapter, a common framework to standardize already proposed consensus methods is presented, and based on this framework different proposals are introduced and analyzed in two different scenarios: Homogeneous and Heterogeneous. The first scenario reflects situations where the networks to be aggregated are rather similar because they are obtained with inference algorithms working on the same data, whereas the second scenario deals with very diverse networks because various sources of data are used to generate the individual networks. A procedure for combining multiple network inference algorithms is analyzed in a systematic way. The results show that there is a very significant difference between these two scenarios, and that the best way to combine networks in the Heterogeneous scenario is not the most commonly used. We show in particular that aggregation in the Heterogeneous scenario can be very beneficial if the individual networks are combined with our new proposed method ScaleLSum.

Key words Consensus network algorithms, Meta-analysis, Gene regulatory networks, Gene expression data

1 Introduction

Inferring gene regulatory networks from expression data is a very challenging problem that has seen a continuously rising interest in the last years and presumably in the years to come due to its applications in biomedical and biotechnological research.

Several studies have compared performances of network inference algorithms [1–9], reaching the conclusion that none of the methods is the best across all types of data and datasets. Furthermore, the different algorithms have specific biases towards the recovery of different regulation patterns, i.e., mutual information (MI) and correlation-based algorithms can recover feed-forward

loops most reliably, while regression and Bayesian networks can more accurately recover linear cascades than MI and correlation-based algorithms [7, 10].

These observations suggest that different network inference algorithms have different strengths and weaknesses [11]. Therefore, combining multiple network inference algorithms emerges as a natural idea to infer a more accurate gene regulatory network (GRN), leading to a consensus among *Homogeneous* networks. We use the term Homogeneous here to refer to networks obtained from the same experimental data through the use of different algorithms. However, there are other situations where different networks describing the same cells are derived from different original datasets coming from very different technologies such as ChIP data, microarray, and RNA-seq. We will refer to this situation as the *Heterogeneous* scenario.

Behind all state-of-the-art consensus network algorithms one can identify a normalization step followed by an aggregation step. The main contributions of this chapter are (1) to systematically analyze the combination of normalization and aggregations strategies, creating in some cases new consensus network algorithms and (2) to evaluate the performances of the algorithms in both the *Homogeneous* and the *Heterogeneous* scenarios, and finally we will also present some alternatives to ensemble GRNs. As will be seen, the conclusions highlight a clear difference in terms of expected performances and algorithm selection in both cases. In order to precisely measure the algorithms performances and to control the degree of homogeneity without depending on any specific network inference algorithm, we rely on a synthetic network generation method presented in this paper. With this approach we can have a full control on the homogeneity and the characteristics of the individual networks.

2 Methods

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* regulate 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 has 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, Thomas et al. [12] made an analysis of different network topologies (motifs) that are commonly inferred incorrectly. Furthermore, Krishnan et al. [13] pointed that

some particular topologies are impossible to be inferred from gene expression data without any further information (knockdowns, existing interactions, etc.).

Several network inference methods have been compared in several papers [7–9, 14]. The main conclusion of [7, 9] 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. As a result, this approach is receiving more and more attention from the scientific community.

In order to create a consensus network from initial individual network inference algorithms, several strategies have been used. Assuming that each algorithm provides a score for each edge of the network, the simplest strategy consists of computing the average of the scores across the N individual networks [15].

Other strategies do not rely on the actual edge scores but on their rank defined in an ordered list. The method proposed in [7] is based on rank averaging: if e_{ij} denotes an edge connecting genes i and j and $r_n(e_{ij})$ the rank of the edge for network n , the final rank is computed with:

$$r(e_{ij}) = \sum_{1 \leq n \leq N} r_n(e_{ij}) \quad (1)$$

The consensus network is obtained from the list composed of the sorted values of summed ranks $r(e_{ij})$. This method called *RankSum* is also known as *Borda* count.

TopkNet [10] is based on the observations made in [7], which showed that integration of algorithms with high-diversity outperforms the integration of algorithms with low-diversity [7]. First, for each individual network the predictions are ranked, and then the final rank for each edge is the result of applying a rank filter of order k , which returns the k -th-greatest value (RankOrderFilter_k) over all the N values:

$$r(e_{ij}) = \text{RankOrderFilter}_k \{ r_1(e_{ij}), \dots, r_N(e_{ij}) \} \quad (2)$$

The computation of a rank value in the *RankSum* and the *TopkNet* algorithms can be viewed as a normalization of the score values before their aggregation with the sum or the rank order filter. Based on this observation, we present an analysis of potential normalization and aggregation strategies in the following section.

We argue that this strategy is the most general one, since the networks may come from different kinds of data. Furthermore, this strategy can be even applied when the networks are constructed from ChIP data or even from the literature.

2.1 Systematic Consensus Analysis

As previously mentioned, 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 will be referred to as “Normalization.” For example, [7, 10] use *Rank*, whereas [15] does not perform any normalization which can be seen as the *Identity* normalization.

Then, the second step is the “Aggregation” of the N different edge scores into one consensus score for every edge. In [7] and [15] this process is done through the *Sum* process, while [10] uses the rank order filter.

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

2.1.1 Normalization

Four different normalization techniques will be analyzed. Let us call $t_n(e_{ij})$ the normalized 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:

$$t_n(e_{ij}) = s_n(e_{ij}) \quad (3)$$

Rank

The *Rank* replaces the numerical score $s_n(e_{ij})$ by their 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:

$$t_n(e_{ij}) = r_n(e_{ij}) \quad (4)$$

Scale

A classical normalization 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:

$$t_n(e_{ij}) = \frac{s_n(e_{ij}) - \mu_n}{\sigma_n} \quad (5)$$

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 normalization does not assure a limited range of values.

ScaleL

The previous proposals normalize the network only taking into consideration the score values. *ScaleL* is an extension of the last normalizing method. This method takes into account the local context of the scores of gene i and j for computing the normalized 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):

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

$$\text{and } \zeta_j = \frac{s_n(e_{ij}) - \mu_{s_j}}{\sigma_{s_j}} \quad (6)$$

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:

$$\mu_{s_i} = \frac{1}{G} \sum_{l=1}^G s_n(e_{il}), \quad (7)$$

$$\sigma_{s_i} = \sqrt{\frac{1}{G-1} \sum_{l=1}^G (s_n(e_{il}) - \mu_{s_i})^2} \quad (8)$$

where G is the number of genes in the network. Note that this rule is related to the CLR network inference method [1] which can be interpreted as a normalization strategy as pointed out in [16]. This normalization 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.

2.1.2 Aggregation

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

Sum

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

$$a(e_{ij}) = \sum_{n=1}^N t_n(e_{ij}). \quad (9)$$

Median

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

$$a(e_{ij}) = \text{Median} \{ t_1(e_{ij}), \dots, t_N(e_{ij}) \} \quad (10)$$

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

Table 1
State-of-the-art consensus network algorithms

Normalization	Aggregation	Name	Reference
<i>Identity</i>	<i>Sum</i>	<i>IdSum</i>	[15]
<i>Rank</i>	<i>Sum</i>	<i>RankSum</i>	[7]
<i>Rank</i>	<i>Median</i>	<i>RankMed</i>	[10]

2.2 Consensus Network Algorithms

The combination of four possible normalization strategies with two possible aggregation rules gives rise to eight 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 1. The other methods have not been reported in the literature and, to our knowledge, they are studied here for the first time.

3 Homogeneous Versus Heterogeneous Network Scenario

The *Homogeneous* scenario reflects the case where the individual networks have been inferred with different algorithms but with the same type of data like gene expression as in [7]. In order to get an estimation of the degree of homogeneity that might be expected in this type of scenario, we have downloaded the networks from the supplemental information of [7]. To measure the homogeneity 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, it is possible to reconstruct GRNs from different kinds of data. As an example a physical, regulatory network is one where edges represent a physical interaction between a transcription factor (TF) and a target gene (TG) as detected in 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 TFs and their targets are supported by functional changes in the gene expression [17], 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

Table 2
Synthetic networks used in this study and their characteristics

Network	Name	Topology	Experiments	Genes	Edges
<i>Rogers1000</i>	R1	Power-law tail topology	1000	1000	1350
<i>SynTReN</i> ₃₀₀	S1	<i>E. coli</i>	800	300	468
<i>SynTReN</i> ₁₀₀₀	S2	<i>E. coli</i>	1000	1000	4695
<i>GNW</i> ₁₅₆₅	G1	<i>E. coli</i>	1565	1565	7264
<i>GNW</i> ₂₀₀₀	G2	Yeast	2000	2000	10,392

data. We thus refer to this scenario as *Heterogeneous* scenario. This situation reflects the case where the individual networks have been inferred from very different data types, and the consequently individual networks are very different, hardly having any edge in common. In [15] 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 of averaging the links across different dataset networks. The final list is computed by sorting these scores decreasingly.

To get an estimation of the expected correlation in this scenario, we have downloaded the networks from supplemental material of [15] and converted them to vectors. Afterwards, we have computed the correlation between the networks obtaining a mean correlation of 0.06.

4 Data

In this section, we present the data that belong to either Homogeneous or Heterogeneous group in order to test the methods described in Subheading 2.

4.1 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 importantly, 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 network (True_{Net}). We use the networks proposed in [9], whose characteristics are detailed in Table 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 with 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.

Heterogeneous 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 subsamplings 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.

4.1.1 Subsampling Strategy

The networks of the two scenarios are generated with Algorithm 1, which is illustrated with a toy example in Fig. 1. A toy True_{Net} is shown in Fig. 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 v_n . In Fig. 1b a particular case of v_n is shown, in this case $\tau = 50$ so v_n has 10 edges. Then, $m\%$ of errors (both false positives and false negatives) are introduced. Following the example and with $m = 30$, this

```

Input:  $\text{True}_{\text{Net}}, N, \tau, m$ 
Output:  $N$  individual synthetic networks  $(\{\eta_n\}_{n=1}^N)$ 
 $n \leftarrow 1;$ 
while  $n \leq N$  do
    if Heterogeneous then
         $| v_n \leftarrow \text{Subsample } \tau \% \text{ edges from the } \text{True}_{\text{Net}};$ 
    else
        if  $n == 1$  then
             $| v_1 \leftarrow \text{Subsample } \tau \% \text{ edges from the } \text{True}_{\text{Net}};$ 
        else
             $| v_n \leftarrow v_1;$ 
        end
    end
     $\eta_n \leftarrow \text{Introduce } m \% \text{ errors in the network } v_n;$ 
    Add noise to  $\eta_n$ ;
     $n \leftarrow n + 1$ 
end

```

Algorithm 1: Generate N individual synthetic networks

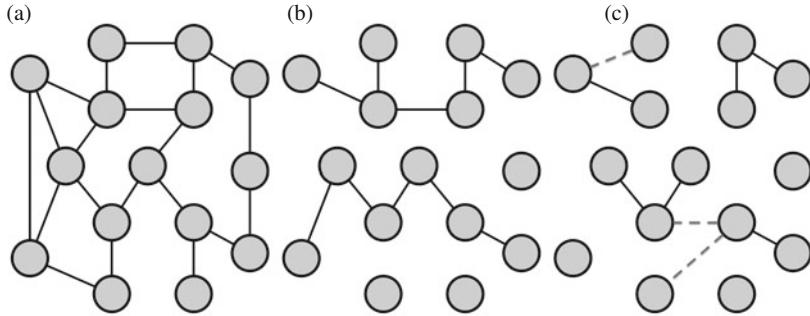


Fig. 1 Toy example illustration of the first steps of Algorithm 1. **(a)** True_{Net}. **(b)** Subsampled network v_n . **(c)** v_n with errors

will introduce three false negatives and three false positives to obtain η_n . The resulting network is presented in Fig. 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 score values in order to ensure non-negative scores in the networks. This step introduces more false positives with a low confidence (if the standard deviation of the noise denoted as δ is small) and also introduces variability of the scores in the true “recovered” edges.

4.2 DREAM5 Data

In a second step, we will use the predictions of 35 real GRN inference algorithms on the first simulated dataset of DREAM5 [7]. We use the inferred networks by the participants of the challenge. These networks are obtained from the same *in silico* data with different approaches. So we can consider them in the *Homogeneous* scenario. However, these networks present a particularity, the different algorithms present a high variability in their performances (see Fig. 3).

4.3 Real Data

Finally, the proposed consensus procedures are tested on public datasets 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 Ratio (FPR), as most genes probably interact with much more TFs than currently documented ones. Moreover, predictions for tran-

scription 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 rewards methods that tend to reproduce current knowledge and penalizes those that could find new results [18].

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

4.3.1 *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 [19, 20] contains the largest and best-known information on transcriptional regulation in *E. coli*. Thus, it has been used as a GS to evaluate the accuracy of the variously constructed networks.

We use different *E. coli* datasets:

1. *Ecoli1*: Many Microbe Microarrays Database (“M3D”) [21] 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 [22] (GSE33147), which contains 96 microarrays of laboratory adaptive evolution experiments using Affymetrix *E. coli* Antisense Genome Arrays.
3. *Ecoli3*: Expression data from [23, 24] which contains 217 arrays that measure 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 are presented in [25]. Then, we use the different consensus strategies to integrate them.

4.3.2 *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 [26].

We have selected the data used in [15], since it provides a Heterogeneous scenario with networks of the same organism that comes from different kinds of data. There is a total of six networks that comes from both functional and physical regulatory interactions.

Table 3
Algorithm 1 parameters to generate the experimental setup for synthetic networks

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

5 Results

As mentioned before, the performances of the various algorithms are benchmarked with both real and synthetic networks.

In [9] we have proposed to evaluate only the best $x\%$ of the total predictions, using the area under the precision-recall curve ($AUPR_{x\%}$). In [25], it is analyzed how the $AUPR_{x\%}$ behaves as a function of x , and we proposed to use the value of $AUPR_{5\%}$ as our metric.

As previously, we use $AUPR_5$ of the consensus network. However 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 normalize the $AUPR_5$ with respect to the mean $AUPR_5$ of the individual networks ($\mu_{AUPR_{5;ind}}$):

$$AUPR_{5;norm} = \frac{AUPR_5}{\mu_{AUPR_{5;ind}}} \quad (11)$$

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

5.1 Results on Synthetic Networks

In the case of synthetic networks, the individual networks are generated with Algorithm 1 with the parameters being specified in Table 3. 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 on Table 2, and this procedure is repeated ten times in order to have different runs of Algorithm 1 and therefore different pools of individual networks.

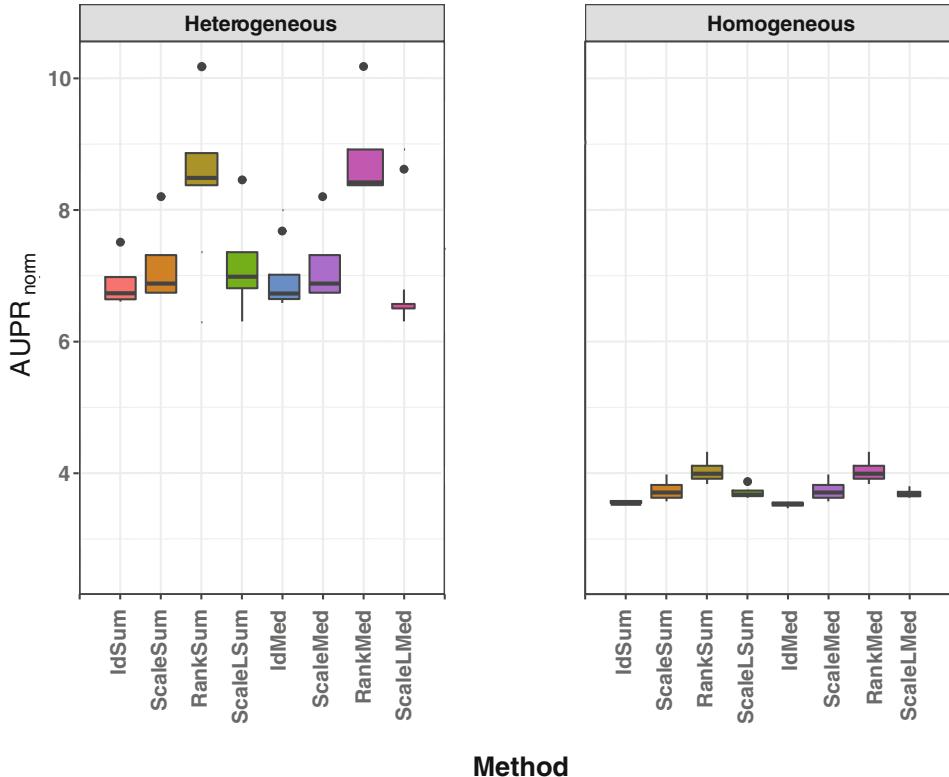


Fig. 2 Boxplots of $AUPR_{5;norm}$ performance of consensus methods on synthetic generated networks

Figure 2 presents the boxplots of $AUPR_{5;norm}$ 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. By observing the figure we can confirm that the *IdSum* method that was used in [15] in a *Heterogeneous* scenario is a good choice for this case. However, using the *Rank* as normalization 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 result in improving the inference compared to the average individual networks as the $AUPR_{5;norm}$ is around 4 for all consensus methods. This conclusion is in line with previous publications such as [10, 15]. However, there are few differences in the *Homogeneous* scenario, but we can still see that *Rank* is performing better than *Id*. Therefore, we think that the best option is to use the *RankSum* [7] or *IdSum* [15], as they are already part of the state-of-the-art and have a simpler normalization 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 still room for improvement. In average, the results improve by 8.5 outperforming the *Homogeneous*' results, which has an average $AUPR_{5,norm}$ of 3.7.

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.

5.2 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 (Fig. 3). 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 generalize across different

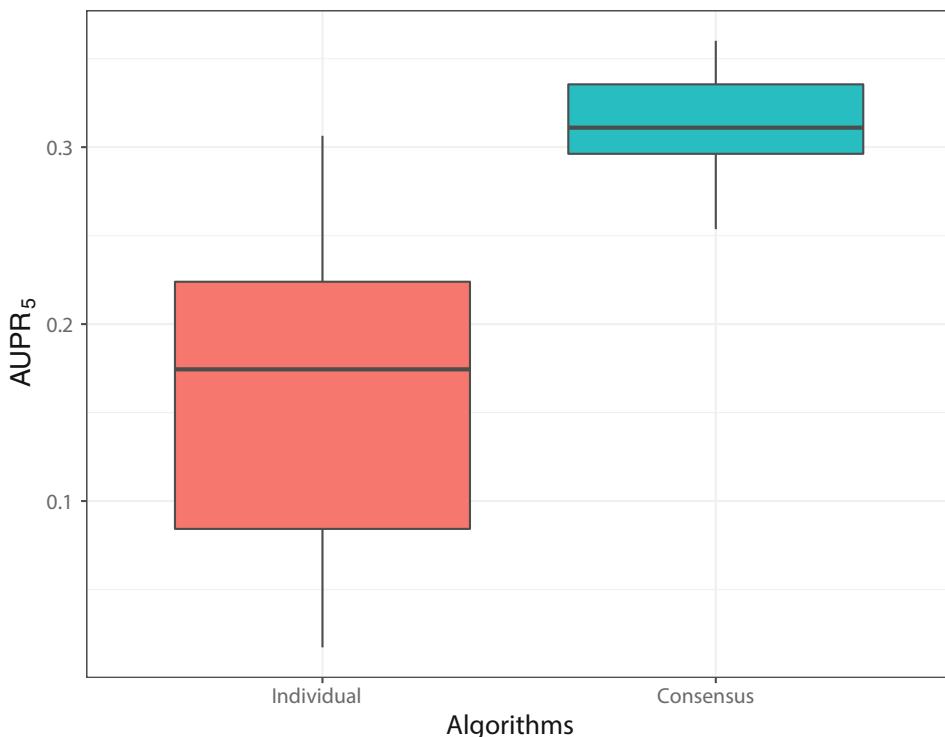


Fig. 3 Boxplots of AUPR₅ performance of individual networks and consensus methods on DREAM5

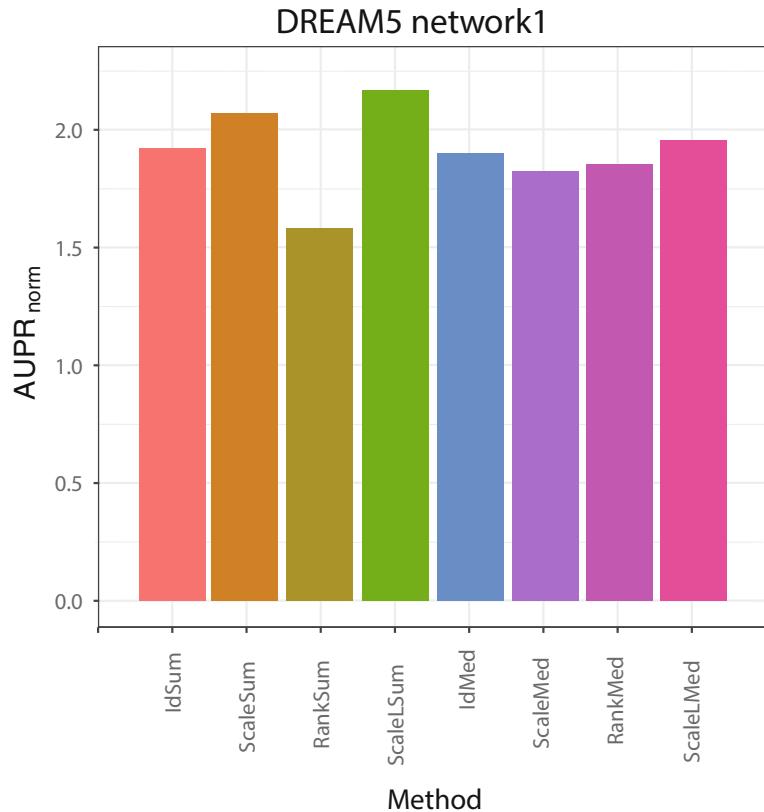


Fig. 4 Boxplots of AUPR₅ performance of individual networks and consensus methods on DREAM5

networks. Moreover, it is difficult to know in a real situation which one of the GRN algorithms has the best performance. Furthermore, in Fig. 3, we can see that the mean value of AUPR₅ of individual networks is 0.17, while the mean value of AUPR₅ of the consensus network is 0.31. Therefore, the averaged AUPR_{5;norm} is 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 Fig. 4. In this case, our proposal *ScaleLSum* 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 normalization step since *ScaleSum* also almost obtains the same performance.

5.3 Results on Real Data

To finish the analysis of consensus methods and their limits, we will evaluate them on the real data described in Subheading 4.3.

With the three different *E. coli* microarray data, we have inferred GRNs with the ten different methods described in [25], containing among others GENIE3 [27], CLR [1], MRNET [2],

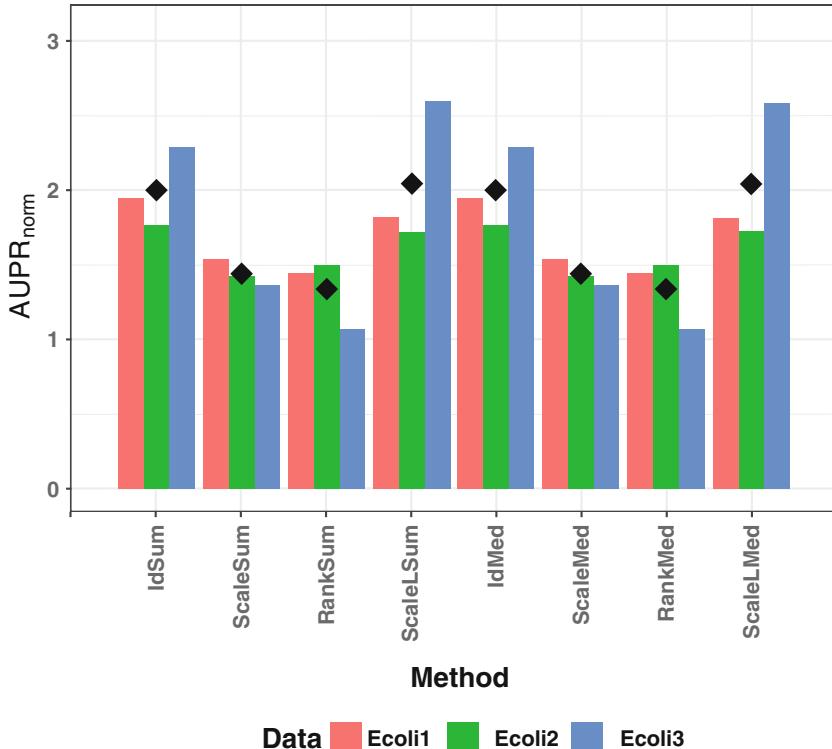


Fig. 5 AUPR_{5;norm} performance of consensus methods on *E. coli* datasets

and C3Net [5]. Afterwards, we have integrated these networks with the different consensus proposals and evaluated them. Figure 5 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 confirms the conclusions reached in the previous subsection. *ScaleLsum* is the best consensus method. In general, we have found that different ranges of values between methods usually needs a normalization as we explain through the chapter. However, in this particular case the different individual GRN inference methods return networks with a similar range of values for the edge scores. This fact could explain why *Id* normalization can be competitive compared with *Rank*, *Scale*, or *ScaleL*.

The previous results reflect that in a *Homogeneous* scenario *ScaleLsum* is the best alternative among the studied consensus methods. In order to have a larger picture, we evaluate the performance of consensus methods in a real *Heterogeneous* scenario with FlyNet data shown in Fig. 6. Observing the figure we can confirm that the *IdSum* method that was used in the original work [15] is the best choice for this case.

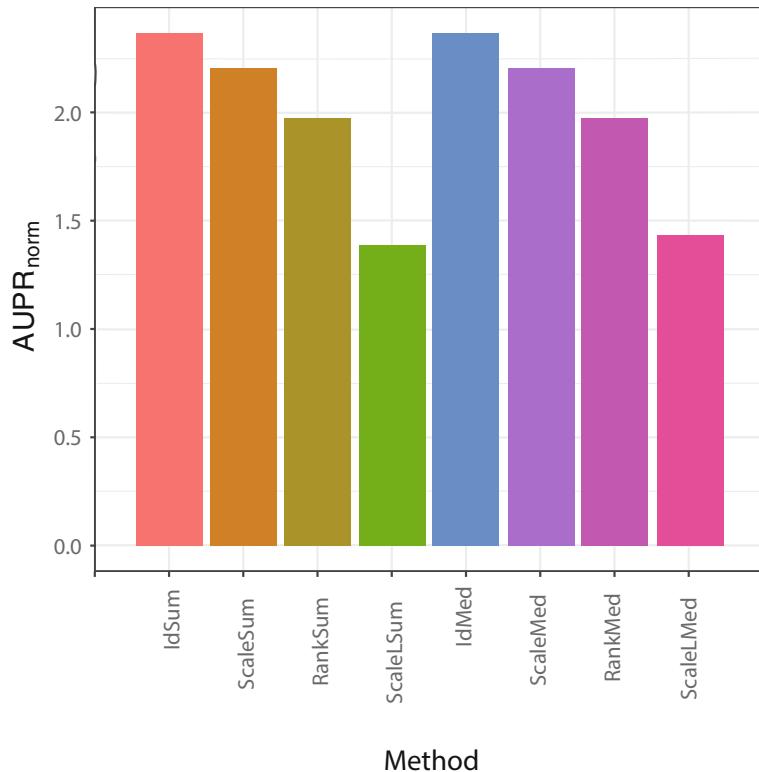


Fig. 6 AUPR_{5;norm} performance of consensus methods on FlyNet

6 Other Unsupervised GRN Ensemble Options

In this chapter, we have presented different approaches to generate a consensus network to try to improve the inferred network. In this section, we present other approaches that does not fit in the presented framework of normalization and aggregation.

6.1 Assembling Pairwise Matrices

In [28] a meta-analysis approach for inferring a GRN 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 signif-

icantly). Thus, to infer a network from various expression data, the approach consists in combining mutual information matrices (MIMs) estimated independently from every single dataset. Then, a GRN is created from inferring the aggregated Mutual Information Matrix using one of the information-theoretic-based GRN inference methods. We refer to the interested reader to [29] for a review of such GRN MIM-based methods.

6.2 Topological Ensemble

In [30] 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 [31]) 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 [32] this cost is minimized several times, obtaining different re-ranked lists. These lists are averaged to get the final output ranking. The cost function penalizes the modification of the original ranking and rewards better structural properties. Ruyssinck et al. [30] proposes to use the frequency of graphlet G4 among the graphlets of four nodes (see Fig. 7).

In [7], it is shown that some algorithms perform better than others. 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).

$$a(e_{ij}) = \sum_{n=1}^N w_n \cdot t_n(e_{ij}). \quad (12)$$

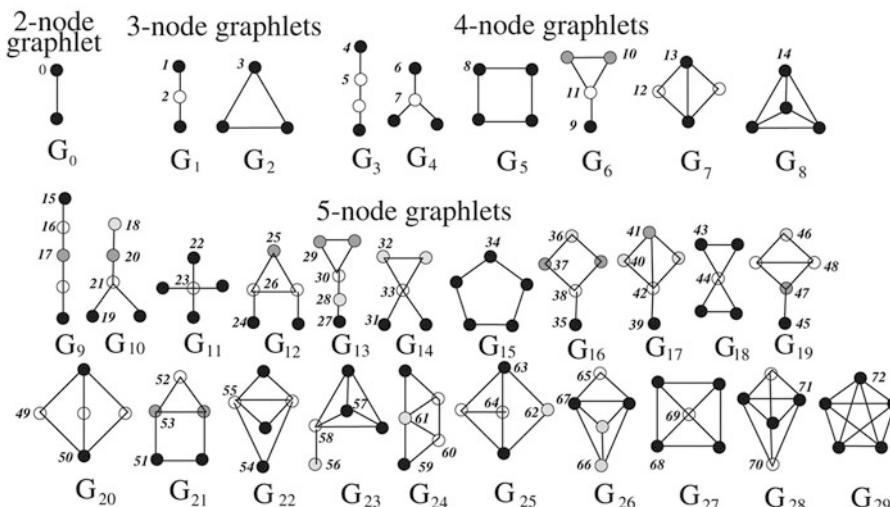


Fig. 7 The 73 automorphism orbits for the graphlets up to five nodes. In a particular graphlet G_i , $i \in \{0, 1, \dots, 29\}$, nodes that belong to the same orbit have the same gray color. Figure taken from [31]

These weights can be learned in a supervised manner as in [15]. The authors propose a logistic regression-based binary classifier, where the class label represents the presence or absence of an edge.

Bellot [25] proposed to estimate these weights in an unsupervised manner, through 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. It concluded that the proposal *ScaleLWsum* seems to be a valuable choice and the best in some of the analyzed real data.

7 Conclusions

In this chapter, we have proposed a framework for combining and integrating different inferred networks. It has been defined as a two-step process, consisting in a normalization 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 analyzed 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 normalization 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 *Heterogeneous* 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 can conclude that *ScaleLSum* is interesting in some cases but maybe not all of them, however, it seems that *IdSum* is a safer option since it obtains one of the best consensus across all different studied cases.

References

1. Faith JJ, Hayete B, Thaden JT, Mogno I, Wierzbowski J, Cottarel G, Kasif S, Collins JJ, Gardner TS (2007) Large-scale mapping and validation of *Escherichia coli* transcriptional regulation from a compendium of expression profiles. *PLoS Biol* 5(1):e8
2. Meyer PE, Kontos K, Lafitte F, Bontempi G (2007) Information-theoretic inference of large transcriptional regulatory networks. *EURASIP J Bioinform Syst Biol*, pp 8–8
3. Meyer P, Kontos K, Bontempi G (2007) Biological network inference using redundancy

- analysis. In: Bioinformatics research and development, pp 16–27
4. Meyer PE, Marbach D, Roy S, Kellis M (2010) Information-theoretic inference of gene networks using backward elimination. In: BIOCOMP, pp 700–705
 5. Altay G, Emmert-Streib F (2010) Inferring the conservative causal core of gene regulatory networks. *BMC Syst Biol* 4(1):132
 6. Altay G, Emmert-Streib F (2010) Revealing differences in gene network inference algorithms on the network level by ensemble methods. *Bioinformatics* 26(14):1738–1744
 7. Marbach D, Costello JC, Küffner R, Vega NM, Prill RJ, Camacho DM, Allison KR, Kellis M, Collins JJ, Stolovitzky G et al (2012) Wisdom of crowds for robust gene network inference. *Nat Methods* 9(8):796–804
 8. Maetschke SR, Madhamshettiar PB, Davis MJ, Ragan MA (2013) Supervised, semi-supervised and unsupervised inference of gene regulatory networks. *Brief Bioinform* p bbt034
 9. Bellot P, Olsen C, Salembier P, Oliveras-Vergés A, Meyer PE (2015) Netbenchmark: a bioconductor package for reproducible benchmarks of gene regulatory network inference. *BMC Bioinf* 16(1):312
 10. Hase T, Ghosh S, Yamanaka R, Kitano H (2013) Harnessing diversity towards the reconstructing of large scale gene regulatory networks. *PLoS Comput Biol* 9(11):e1003361
 11. Marbach D, Prill RJ, Schaffter T, Mattiussi C, Floreano D, Stolovitzky G (2010) Revealing strengths and weaknesses of methods for gene network inference. *Proc Natl Acad Sci* 107(14):6286–6291
 12. Thomas S, Marbach D, Floreano D (2011) GeneNetWeaver: in silico benchmark generation and performance profiling of network inference methods. *Bioinformatics* 27(16):2263–2270
 13. Krishnan A, Giuliani A, Tomita M (2007) Indeterminacy of reverse engineering of gene regulatory networks: the curse of gene elasticity. *PLoS One* 2(6):e562
 14. Emmert-Streib F, Glazko GV, Altay G, Simoes RdM (2012) Statistical inference and reverse engineering of gene regulatory networks from observational expression data. *Front Genet* 3:8
 15. Marbach D, Roy S, Ay F, Meyer PE, Candeias R, Kahveci T, Bristow CA, Kellis M (2012) Predictive regulatory models in drosophila melanogaster by integrative inference of transcriptional networks. *Genome Res* 22(7):1334–1349
 16. Bellot P, Meyer PE (2014) Efficient combination of pairwise feature networks. In: NCW2014 ECML
 17. Capaldi AP, Kaplan T, Liu Y, Habib N, Regev A, Friedman N, O'Shea EK (2008) Structure and function of a transcriptional network activated by the MAPK Hog1. *Nat Genet* 40(11):1300–1306
 18. De Smet R, Marchal K (2010) Advantages and limitations of current network inference methods. *Nat Rev Microbiol* 8(10):717–729
 19. Gama-Castro S, Salgado H, 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 Res* 39:D98–D105
 20. Salgado H, Martínez-Flores I, Lopez-Fuentes A (2012) Extracting regulatory networks of Escherichia coli from RegulonDB. *Methods Mol Biol* 804:179–195
 21. Faith J, Driscoll M, Fusaro V (2008) Many Microbe Microarrays Database: uniformly normalized Affymetrix compendia with structured experimental metadata. *Nucleic Acids Res* 36:D866–D870
 22. Fong S, Joyce A, Palsson B (2005) Parallel adaptive evolution cultures of Escherichia coli lead to convergent growth phenotypes with different gene expression states. *Genome Res* 15:1365–1372
 23. Sangurdekar D, Srienc F (2006) A classification based framework for quantitative description of large-scale microarray data. *Genome Biol* 7:R32
 24. Xiao G, Wang X, Khodursky A (2011) Modeling three-dimensional chromosome structures using gene expression data. *J Am Stat Assoc* 106:61–72
 25. Bellot P (2017) Study of gene regulatory networks inference methods from gene expression data. PhD thesis, Universitat Politècnica de Catalunya
 26. Halfon M, Gallo S, Bergman C (2008) REDfly 2.0: an integrated database of cis-regulatory modules and transcription factor binding sites in Drosophila. *Nucleic Acids Res* 36:D594–D598
 27. Huynh-Thu VA, Irrthum A, Wehenkel L, Geurts P (2010) Inferring regulatory networks from expression data using tree-based methods. *PloS One* 5(9):e12776

28. Pham NC, Haibe-Kains B, Bellot P, Bontempi G, Meyer PE (2016) Study of meta-analysis strategies for network inference using information-theoretic approaches. In: Biological knowledge discovery and data mining
29. Meyer PE, Olsen C, Bontempi G (2011) Transcriptional network inference based on information theory. In: Applied statistics for network biology: methods in systems biology. Wiley-Blackwell, Weinheim, pp 67–89
30. Ruyssinck J, Demeester P, Dhaene T, Saeys Y (2016) Netter: re-ranking gene network inference predictions using structural network properties. *BMC Bioinf* 17(1):76
31. Pržulj N (2007) Biological network comparison using graphlet degree distribution. *Bioinformatics* 23(2):e177–e183
32. Hwang CR (1988) Simulated annealing: theory and applications. *Acta Appl Math* 12: 108–111



Chapter 13

Learning Differential Module Networks Across Multiple Experimental Conditions

Pau Erola, Eric Bonnet, and Tom Michoel

Abstract

Module network inference is a statistical method to reconstruct gene regulatory networks, which uses probabilistic graphical models to learn modules of coregulated genes and their upstream regulatory programs from genome-wide gene expression and other omics data. Here, we review the basic theory of module network inference, present protocols for common gene regulatory network reconstruction scenarios based on the Lemon-Tree software, and show, using human gene expression data, how the software can also be applied to learn differential module networks across multiple experimental conditions.

Key words Gene regulatory network inference, Module networks, Differential networks, Bayesian analysis

1 Introduction

Complex systems composed of a large number of interacting components often display a high level of modularity, where independently functioning units can be observed at multiple organizational scales [1]. In biology, a module is viewed as a discrete entity composed of many types of molecules and whose function is separable from that of other modules [2]. The principle of modularity plays an essential role in understanding the structure, function, and evolution of gene regulatory, metabolic, signaling, and protein interaction networks [3]. It is therefore not surprising that functional modules also manifest themselves in genome-wide data. Indeed, from the very first studies examining genome-wide gene expression levels in yeast, it has been evident that clusters of co-expressed genes, i.e., sharing the same expression profile over time or across different experimental perturbations, reveal important information about the underlying biological processes [4, 5]. Module network inference takes this principle one step

further and aims to infer simultaneously co-expression modules and their upstream regulators [6, 7]. From a statistical perspective, modularity allows to reduce the number of model parameters that need to be determined, because it is assumed that genes belonging to the same module share the same regulatory program, and therefore allows to learn more complex models, in particular nonlinear probabilistic graphical models [8], than would otherwise be possible.

While module networks were originally introduced to infer gene regulatory networks from gene expression data alone [6], the method has meanwhile been extended to also include expression quantitative trait loci data [9, 10], regulatory prior data [11], microRNA expression data [12], clinical data [13], copy number variation data [14, 15], or protein interaction networks [16]. Furthermore, the method can be combined with gene-based network inference methods [17, 18]. Finally, the module network method has been applied in numerous biological, biotechnological, and biomedical studies [19–29].

An area of interest that has received comparatively limited attention to date concerns the inference of *differential* module networks. Differential networks extend the concept of differential expression, and are used to model how co-expression, regulatory, or protein–protein interaction networks differ between two or more experimental conditions, cell or tissue types, or disease states [30, 31]. Existing differential network inference methods are mainly based on pairwise approaches, either by testing for significant differences between correlation values in different conditions or by estimating a joint graph from multiple data sets simultaneously using penalized likelihood approaches [32–35]. The inference of differential module networks is more challenging, because it requires a matching or comparable set of modules across the conditions of interest. A related problem has been addressed in a study of the evolutionary history of transcriptional modules in a complex phylogeny, using an algorithm that maps modules across species and allows to compare their gene assignments [36].

In this chapter, we review the theoretical principles behind module network inference, explain practical protocols for learning module networks using the Lemon-Tree software [15], and show in a concrete application on human gene expression data how the software can also be used to infer differential module networks using a similar principle as in [36].

2 Module Network Inference: Theory and Algorithms

2.1 The Module Network Model

Module networks are probabilistic graphical models [7, 8] where each gene g_i , $i \in \{1, \dots, G\}$, is represented by a random variable X_i taking continuous values. In a standard probabilistic graphical model or Bayesian network, it is assumed that the distribution

of X_i depends on the expression level of a set of regulators \mathcal{P}_i (the “parents” of gene i). If the causal graph formed by drawing directed edges from parents to their targets is acyclic, then the joint probability distribution for the expression levels of all genes can be written as a product of conditional distributions:

$$p(x_1, \dots, x_G) = \prod_{i=1}^G p(x_i | \{x_j : j \in \mathcal{P}_i\}). \quad (1)$$

In data integration problems, we are often interested in explaining patterns in one data type (e.g., gene expression) by regulatory variables in another data type (e.g., transcription factor binding sites, single nucleotide or copy number variations, etc.). In this case, the causal graph is bipartite, and the acyclicity constraint is satisfied automatically.

In a module network, we assume that genes are partitioned into *modules*, such that genes in the same module share the same parameters in the distribution function (1). Hence, a module network is defined by a partition of $\{1, \dots, G\}$ into K modules \mathcal{A}_k , a collection of parent genes \mathcal{P}_k for each module k , and a joint probability distribution:

$$p(x_1, \dots, x_G) = \prod_{k=1}^K \prod_{i \in \mathcal{A}_k} p(x_i | \{x_j : j \in \mathcal{P}_k\}). \quad (2)$$

In a module network, only one conditional distribution needs to be parameterized per module, and hence it is clear that if $K \ll G$, the number of model parameters in Eq. (2) is much smaller than in Eq. (1). Moreover, data from genes belonging to the same module are effectively pooled, leading to more robust estimates of these model parameters. This is the main benefit of the module network model.

In principle, any type of conditional distribution can be used in Eq. (2). For instance, in a linear Gaussian framework [8], one would assume that each gene is normally distributed around a linear combination of the parent expression levels. However, the pooling of genes into modules allows for more complex, nonlinear models to be fitted. Hence, it was proposed that the conditional distribution of the expression level of the genes in module k is normal with mean and standard deviation depending on the expression values of the parents of the module through a regression tree (the “*regulatory program*” of the module) [6] (Fig. 1). The tests on the internal nodes of the regression tree are usually defined to be of the form $x \geq v$ or not, for a split value v , where x is the expression value of the parent associated to the node.

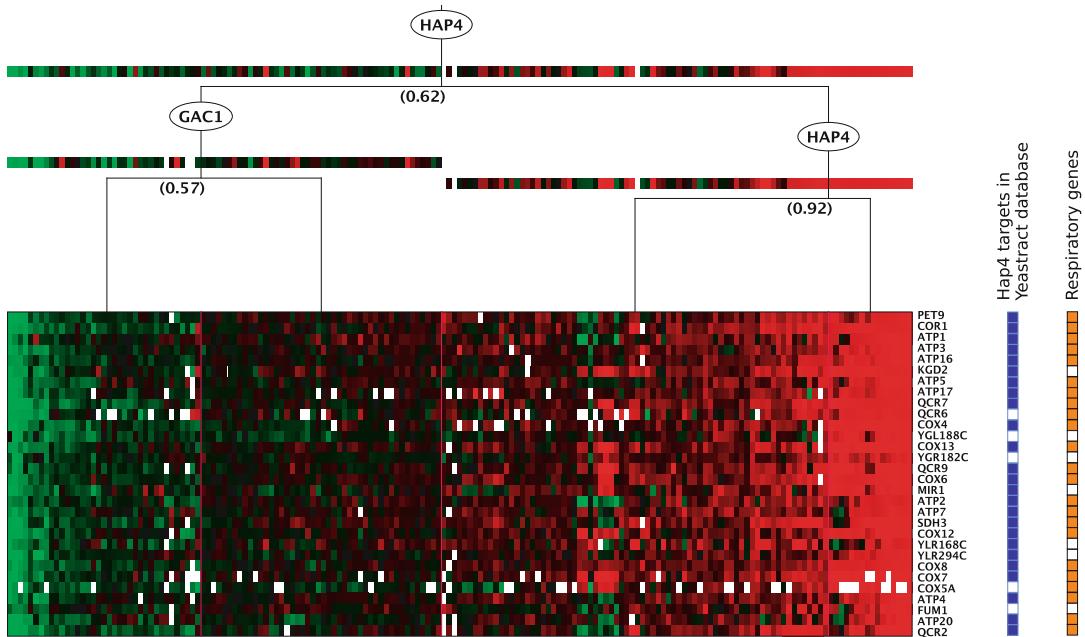


Fig. 1 Example of a module and regulatory decision tree inferred from yeast data, with Hap4 assigned as a top regulator. Genes known to be regulated by Hap4 in YEASTRACT are marked in blue and those involved in respiration are marked in orange. Adapted from Joshi *et al.*, *Module networks revisited: computational assessment and prioritization of model predictions*, *Bioinformatics*, 2009, 25(4):490–496 [37], by permission of Oxford University Press

Given a module network specification \mathcal{M} , consisting of gene module assignments, regulatory decision trees, and normal distribution parameters at the leaf nodes, the probability density of observing an expression data matrix $\mathbf{X} = (x_{im}) \in \mathbb{R}^{G \times N}$ for G genes in N samples is given by:

$$\begin{aligned} P(\mathbf{X} | \mathcal{M}) &= \prod_{m=1}^N \prod_{k=1}^K \prod_{i \in \mathcal{A}_k} p(x_{im} | \{x_{jm}: j \in \mathcal{P}_k\}) \\ &= \prod_{k=1}^K \prod_{\ell=1}^{L_k} \prod_{i \in \mathcal{A}_k} \prod_{m \in \mathcal{E}_\ell} p(x_{im} | \mu_\ell, \sigma_\ell), \end{aligned}$$

where L_k is the number of leaf nodes of module k 's regression tree, \mathcal{E}_ℓ denotes the experiments that end up at leaf ℓ after traversing the regression tree, and (μ_ℓ, σ_ℓ) are the normal distribution parameters at leaf ℓ . The Bayesian model score is obtained by taking the log-marginal probability over the parameters of the normal distributions at the leaves of the regression trees with a normal-gamma prior:

$$\begin{aligned}
S &= \sum_k S_k = \sum_k \sum_{\ell} S_k(\mathcal{E}_\ell) \\
S_k(\mathcal{E}_\ell) &= -\frac{1}{2} R_0^{(\ell)} \log(2\pi) + \frac{1}{2} \log \left(\frac{\lambda_0}{\lambda_0 + R_0^{(\ell)}} \right) \\
&\quad - \log \Gamma(\alpha_0) + \log \Gamma(\alpha_0 + \frac{1}{2} R_0^{(\ell)}) \\
&\quad + \alpha_0 \log \beta_0 - (\alpha_0 + \frac{1}{2} R_0^{(\ell)}) \log \beta_1
\end{aligned} \tag{3}$$

where $R_q^{(\ell)}$ are the sufficient statistics at leaf ℓ ,

$$R_q^{(\ell)} = \sum_{m \in \mathcal{E}_\ell} \sum_{i \in \mathcal{A}_k} x_{i,m}^q, \quad q = 0, 1, 2,$$

and

$$\beta_1 = \beta_0 + \frac{1}{2} \left[R_2^{(\ell)} - \frac{(R_1^{(\ell)})^2}{R_0^{(\ell)}} \right] + \frac{\lambda_0 (R_1^{(\ell)} - \mu_0 R_0^{(\ell)})^2}{2(\lambda_0 + R_0^{(\ell)}) R_0^{(\ell)}}.$$

Details of this calculation can be found in [38, 39].

2.2 Optimization Algorithms

The first optimization strategy proposed to identify high-scoring module networks was a **greedy hill-climbing** algorithm [6]. This algorithm starts from an initial assignment of genes to co-expression clusters (e.g., using k-means), followed by assigning a new regulator to each module by iteratively finding the best (if any) new split of a current leaf node into two new leaf nodes given the current set of gene-to-module assignments, and reassigning genes between modules given the current regression tree, while preserving acyclicity throughout. The decomposition of the Bayesian score [Eq. (3)] as a sum of leaf scores of the different modules allows for efficient updating after every regulator addition or gene reassignment.

An improvement to this algorithm was found, based on the observation that the Bayesian score depends only on the assignment of samples to leaf nodes, and not on the actual regulators or tree structure that induce this assignment [39]. Hence, a **decoupled greedy hill-climbing** algorithm was developed, where first the Bayesian score is optimized by two-way clustering of genes into modules and samples into leaves for each module, and then a regression tree is found for the converged set of modules by hierarchically merging the leave nodes and finding the best regulator to explain the split below the current merge. This algorithm achieved comparable score values as the original one, while being considerably faster [39].

Further analysis of the greedy two-way clustering algorithm revealed the existence of multiple local optima, in particular for moderate to large data sets (~ 1000 genes or more), where considerably different module assignments result in near-identical scores. To address this issue, a **Gibbs sampler** method was developed, based on the Chinese restaurant process [40], for sampling from the posterior distribution of two-way gene/sample clustering solutions [41]. By sampling an ensemble of multiple, equally probable solutions, and extracting a core set of “tight clusters” (groups of genes which consistently cluster together), gene modules are identified that are more robust to fluctuations in the data and have higher functional enrichment compared to the greedy clustering strategies [37, 41].

Finally, the Gibbs sampling strategy for module identification was complemented with a probabilistic algorithm, based on a logistic regression of sample splits on candidate regulator expression levels, for sampling and ensemble averaging of regulatory programs, which resulted in more accurate regulator assignments [37].

3 The Lemon-Tree Software Suite for Module Network Inference

3.1 Lemon-Tree Software Package

Lemon-Tree is a software suite implementing all of the algorithms discussed in Subheading 2.2. Lemon-Tree has been benchmarked using large-scale tumor data sets and shown to compare favorably with other module network inference methods [15]. Its performance has been carefully assessed also in an independent study not involving the software authors [42]. Lemon-Tree is self-contained, with no external program dependencies and is entirely coded in the JavaTM programming language. Users can download a pre-compiled version of the software, or alternatively they can download and compile the software from the source code, which is available on the GitHub repository (<https://github.com/eb00/lemon-tree>). Note that there is also a complete wiki on the Lemon-Tree GitHub (<https://github.com/eb00/lemon-tree/wiki>), with detailed instruction on how to download, compile, use the software, what are the default parameters, and an extended bibliography on the topic of module networks.

Lemon-Tree is a command-line software, with no associated graphical user interface at the moment. The different steps for building the module network are done by launching commands with different input files that will generate different output files. All the command-line examples below are taken from the Lemon-Tree tutorial that users are encouraged to download and reproduce by themselves.

The purpose of the Lemon-Tree software package is to create a module network from different types of “omics” data. The end

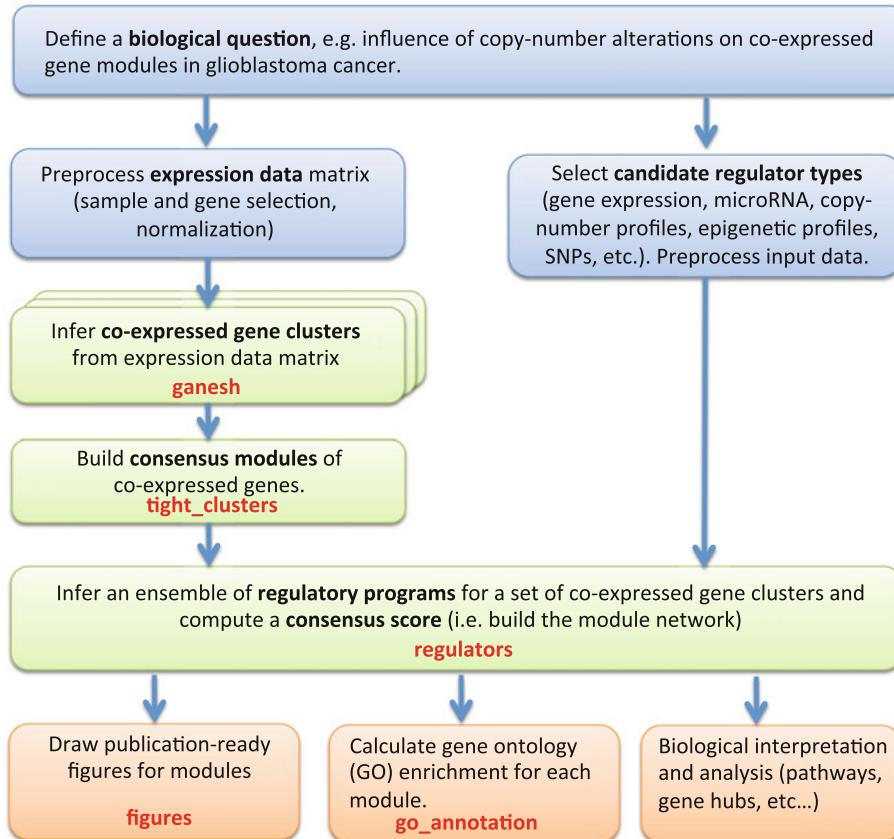


Fig. 2 Flow chart for module network inference with Lemon-Tree. This figure shows the general workflow for a typical integrative module network inference with Lemon-Tree. Blue boxes indicate the pre-processing steps that are done using third-party software such as R or user-defined scripts. Green boxes indicate the core module network inference steps done with the Lemon-Tree software package. Typical post-processing tasks (orange boxes), such as GO enrichment calculations, can be performed with Lemon-Tree or other tools. The Lemon-Tree task names are indicated in red (see main text for more details). Figure reproduced from [15] under Creative Commons Attribution License

result is a set of gene clusters (co-expressed genes), and their associated “regulators.” The regulators can be of different types, for instance, mRNA expression, copy-number profiles, variants (such as single nucleotide variants), or even clinical parameter profiles can be used. There are three fundamental steps or tasks to build a module network with Lemon-Tree (Fig. 2):

- Generate several cluster solutions (“ganesh” task).
- Merge the different cluster solutions using the fuzzy clustering algorithm (“tight_clusters” task).
- Assign regulators to each cluster, producing the module network (“regulators” task).

3.2 Ganesh Task

The goal of this task is to cluster genes from a matrix (rows) using a probabilistic algorithm (Gibbs sampling) [41]. This step is usually done on the mRNA expression data only, although some other data type could be used, for instance, proteomic expression profiles. We first select genes having non-flat profiles, by keeping genes having a standard deviation above a certain value (0.5 is often used as the cutoff score, but this value might depend on the data set). The data is then centered and scaled (by row) to have a mean of 0 and a standard deviation of 1. To find one clustering solution, the following command can be used (the command is spread here over multiple lines, but should be entered on a single line without the backslash characters):

```
java -jar lemontree.jar -task ganesh \
-data_file data/expr_matrix.txt \
-output_file ganesh_results/cluster1
```

The clustering procedure should be repeated multiple times, using the same command, only changing the name of the output file. For instance, we could generate 5 runs, named cluster1, cluster2, cluster3, cluster4, and cluster5, with the same command, just by changing the name of the output file.

3.3 Tight Clusters Task

Here, we are going to generate a single, robust clustering solution from all the individual solutions generated at the previous step, using a graph clustering algorithm [43]. Basically, we group together genes that frequently co-occur in all the solutions. Genes that are not strongly associated to a given cluster will be eliminated.

```
java -jar lemontree.jar -task tight_clusters \
-data_file data/expr_matrix.txt \
-cluster_file cluster_file_list \
-output_file tight_clusters.txt \
-node_clustering true
```

The “cluster_file” is a simple text file, listing the location of all the individual cluster files generated at the “ganesh” step. By default, the tight clusters procedure is keeping only clusters that have a minimum of 10 genes (this can be easily changed by overriding a parameter in the command).

3.4 Revamp Task

This task is aimed at maximizing the Bayesian co-expression clustering score of an existing module network while preserving the initial number of clusters. A threshold can be specified to avoid that genes are reassigned if the score gain is below this threshold and allowing the systematic tracking of the conservation and divergence of modules with respect to the initial partition. This task can be used to optimize an existing module network obtained with a different clustering algorithm, or to optimize an existing module

network for a different data matrix, e.g., a subset of samples as presented in Subheading 4.

```
java -jar lemontree.jar -task revamp \
-data_file data/expr_matrix.txt \
-cluster_file cluster_file.txt \
-reassign_thr 0.0 \
-output_file revamped_clusters.txt \
-node_clustering true
```

The “cluster_file.txt” is a simple text clustering file, like the one obtained in “tight_clusters” step, and “reassign_thr” is the score gain threshold that must be reached to move a gene from one cluster to another. By default, this reassignment threshold is set to 0.

3.5 Regulators Task

In this task, we assign sets of “regulators” to each of the modules using a probabilistic scoring, taking into account the profile of the candidate regulator and how well it matches the profiles of co-expressed genes [37]. The candidate regulators can be divided into two different types, depending on the nature of their profiles: continuous or discrete. The first type can be, for example, transcription factors or signal transducers mRNA expression profiles (selected from the same matrix used for detecting co-expressed genes), microRNA expression profiles, or gene copy-number variants profiles (CNVs). For the latter, the numerical values will be integers, such as the different clinical grades characterizing a disease state (discrete values), or single nucleotide variant profiles (SNVs, characterized by profiles with 0/1 values). In all cases, the candidate regulator profiles must have been made on the same samples as the tight clusters defined previously. Missing values are allowed, but obviously they should not constitute the majority of the values in the profile. Note that a patch to the regulator assignment implementation identified in [42] is included in Lemon-Tree version 3.0.5 or above.

Once the list of candidate regulators is established, the assignment to the clusters can be made with a single command like this:

```
java -jar lemontree.jar -task regulators
-data_file data/expr_matrix.txt \
-reg_file data/reg_list.txt \
-cluster_file tight_clusters.txt \
-output_file results/reg_tf
```

The “reg_file” option is a simple text list of candidate regulators that are present in the expression matrix. If the regulators are discrete, it is mandatory to add a second column in the text file, describing the type of the regulator (“c” for continuous or “d” for discrete). The profiles for co-expressed genes and for all the

regulators must be included in the matrix indicated by the `data_file` parameter.

Note that this command will create four different output files, using the “`output_file`” parameter as the prefix for all the files.

- `reg_tf.topreg.txt`: Top 1% regulators assigned to the modules.
- `reg_tf.allreg.txt`: All the regulators assigned.
- `reg_tf.randomreg.txt`: Regulators assigned randomly to the modules.
- `reg_tf.xml.gz`: xml file containing all the regulatory trees used for assigning the regulators.

The regulators text files all have the same format: three columns representing, respectively, the regulator name, the module number, and the score value.

3.6 Figures Task

This task is creating one figure per module. The figure represents the expression values color coded with a gradient ranging from dark blue (low expression values) to bright yellow (high expression values). All the module genes are in the lower panel, while the top regulators for the different classes or types of regulators (if any) are displayed in the upper panel. A regulation tree is represented on top of the figure, with the different split points highlighted in the figure as vertical red lines. The name of each gene is displayed on the left of the figure.

```
java -jar lemontree.jar \
-task figures \
-top_regulators reg_files.txt \
-data_file data/all.txt \
-reg_file data/reg_list.txt \
-cluster_file tight_clusters.txt \
-tree_file results/reg_tf.xml.gz
```

Note that the “`top_regulators`” parameter is a simple text file listing the different top regulator files and their associated clusters. Such a file could be, for instance, the file `reg_tf.topreg.txt` mentioned in the previous paragraph. All figures are generated in the `eps` (encapsulated postscript) format, but it is relatively easy to convert this format to other common formats such as `pdf`.

3.7 GO Annotation Task

The goal of this task is to calculate the GO (Gene Ontology) category enrichment for each module, using code from the BiNGO package [44]. We have to specify two GO annotation files that are describing the GO codes associated with the genes (“`gene_association.goa_human`”) and another file describing the GO graph (“`gene_ontology_ext.obo`”). These files can be downloaded for various organisms from the GO website (<http://www.geneontology.org>). We also specify the set of genes that should be

used as the reference for the calculation of the statistics, in this case the list of all the genes that are present on the microarray chip (file “all_gene_list”). The results are stored in the output file “go.txt.”

```
java -jar lemontree.jar \
-tasks go_annotation \
-cluster_file tight_clusters.txt \
-go_annot_file gene_association.goa_human \
-go_ontology_file gene_ontology_ext.obo \
-go_ref_file all_gene_list \
-output_file go.txt
```

4 Differential Module Network Inference

4.1 Differential Module Network Model

Assume that we have expression data in T different conditions (e.g., experimental treatments, cell or tissue types, and disease stages or states), with N_t samples in each condition $t \in \{1, \dots, T\}$, and wish to study how the gene regulatory network differs (or not) between conditions. We define a differential module network as a collection of module networks $\{\mathcal{M}_1, \dots, \mathcal{M}_T\}$, one for each condition, subject to constraints, and model gene expression levels for G genes as:

$$p(x_1, \dots, x_G | \mathcal{M}_1, \dots, \mathcal{M}_T) = \prod_{t=1}^T p(x_1, \dots, x_G | \mathcal{M}_t), \quad (4)$$

where each factor is a model of the form of Eq. (2). Hence, if gene i is assigned to modules $\{k_{i1}, \dots, k_{iT}\}$ in each module network, its parent set is the union $\mathcal{P}_i = \bigcup_{t=1}^T \mathcal{P}_{k_{it}}$. If the graph mapping these parent sets to their targets is acyclic, Eq. (4) defines a proper Bayesian network. If the individual factors $p(x_1, \dots, x_G | \mathcal{M}_t)$ are the usual Gaussians with parameters depending on the parent expression levels in that module network, their product remains a Gaussian. By Bayes’ theorem, we can write, for a concatenated data matrix $\mathbf{X} = (\mathbf{X}_1, \dots, \mathbf{X}_T)$,

$$p(\mathcal{M}_1, \dots, \mathcal{M}_T | \mathbf{X}) \propto p(\mathcal{M}_1, \dots, \mathcal{M}_T) \prod_{t=1}^T p(\mathbf{X}_t | \mathcal{M}_t) \quad (5)$$

If we assume independence, $p(\mathcal{M}_1, \dots, \mathcal{M}_T) = \prod_t p(\mathcal{M}_t)$, then optimization of, or sampling from, Eq. (5) is the same as inferring module networks independently in each condition, but this will reveal little of the underlying relations between the conditions. Instead, we assume that there exists a conserved set of modules across all conditions, but their gene and regulator assignment may differ in each condition. This results in the following constraints:

1. The number of modules must be the same in each module network, i.e., $p(\mathcal{M}_1, \dots, \mathcal{M}_T) = 0$ unless $K_1 = \dots = K_T = K$.
2. Module networks with more similar gene and/or regulator assignments are more likely a priori,

$$\log p(\mathcal{M}_1, \dots, \mathcal{M}_T) = - \sum_{k=1}^K \sum_{t,t'} \left[\lambda_{t,t'} f(\mathcal{A}_k^{(t)}, \mathcal{A}_k^{(t')}) + \mu_{t,t'} g(\mathcal{P}_k^{(t)}, \mathcal{P}_k^{(t')}) \right], \quad (6)$$

where f and g are distance functions on sets (e.g., Jaccard distance), and $\lambda_{t,t'}$ and $\mu_{t,t'}$ are penalty parameters that encode the relative a priori similarity between conditions.

4.2 Optimization Algorithm

For simplicity, we assume here that $\mu_{t,t'} = 0$ and $\lambda_{t,t'} = \lambda$ for all (t, t') in Eq. (6), i.e., we will only constrain the gene assignments, and uniformly so for all condition pairs; the complete model will be treated in detail elsewhere. More general forms of $\lambda_{t,t'}$ can be used, for instance, to mimic the model of [36], where conditions represented different species and gene reassessments were constrained by a phylogenetic tree. With a fixed λ , instead of modeling λ and f explicitly, we observe that the effect of including f in the model (5) is to impose a penalty on gene reassessments: starting from identical modules in all conditions, a gene reassignment in condition t increases the posterior log-likelihood only if its increase in $\log p(\mathbf{X}_t \mid \mathcal{M}_t)$ is sufficiently large to overcome the penalty induced by Eq. (6). This can be modeled equivalently by setting a uniform module reassignment score threshold as an external parameter. Hence, we propose the following heuristic optimization algorithm for differential module network inference using Lemon-Tree:

1. Create a concatenated gene expression matrix $\mathbf{X} = (\mathbf{X}_1, \dots, \mathbf{X}_T)$ and learn a set of co-expression modules using tasks “ganesh” (Subheading 3.2) and “tight clusters” (Subheading 3.3). This results in a set of module networks $(\mathcal{M}_1, \dots, \mathcal{M}_T)$ with identical module assignments and empty parent sets.
2. Set a reassignment threshold value and use task “revamp” (Subheading 3.4) to maximize the Bayesian co-expression clustering score $\log p(\mathbf{X}_t \mid \mathcal{M}_t)$ [cf. Eq. (3)] for each condition independently, but subject to the constraint that gene reassessments must pass the Bayesian score difference threshold.
3. Assign regulators to each module for each condition independently using task “regulators” (Subheading 3.5).

4.3 Reconstruction of a Differential Module Network Between Atherosclerotic and Non-atherosclerotic Arteries in Cardiovascular Disease Patients

To illustrate the differential module network inference algorithm, we applied it to 68 atherosclerotic (i.e., diseased) arterial wall (AAW) samples and 79 non-atherosclerotic (i.e., non-diseased) internal mammary artery (IMA) samples from the Stockholm Atherosclerosis Gene Expression study [45–47], using 1803 genes with variance greater than 0.5 in the concatenated data. The STAGE study was designed to study the effect of genetic variation on tissue-specific gene expression in cardiovascular disease [46]. According to the systems genetics paradigm, genetic variants in regulatory regions affect nearby gene expression (“*cis*-eQTL effects”), which then causes variation in downstream gene networks (“*trans*-eQTL effects”) and clinical phenotypes [47, 48]. We therefore considered as candidate regulators the tissue-specific sets of genes with significant eQTLs [46] and present in our filtered gene list (668 AAW and 964 IMA genes, 267 in common), and ran the “regulators” task on each set of modules independently.

As expected, independent clustering of the two data sets results in different numbers of modules, and an inability to map modules unambiguously across tissues (Fig. 3a). In contrast, application of the differential module network optimization algorithm (Subheading 4.2) results in a one-to-one mapping of modules, whose average overlap varies smoothly as a function of the reassignment threshold value (Fig. 3b).

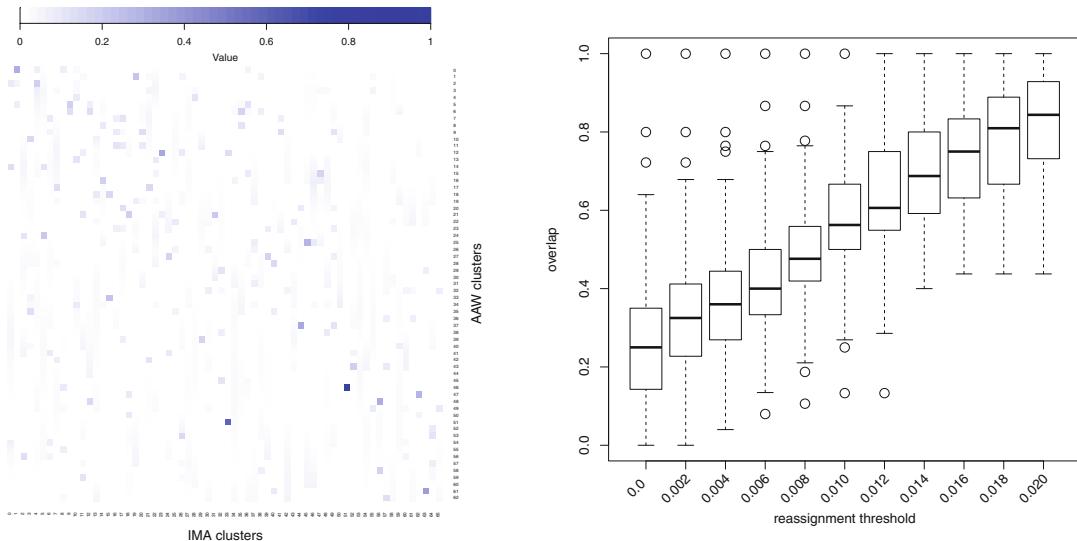


Fig. 3 Differential module network inference on STAGE AAW and IMA tissues. **(a)** Independent clustering of tissue-specific data results in poorly identifiable module relations between tissues. Shown is the pairwise overlap fraction for all pairs of modules inferred in AAW (rows) and IMA (columns). **(b)** Joint clustering of data across both tissues using the “revamp” task in Lemon-Tree results in a one-to-one mapping of modules with a tunable level of overlap. Shown are the module overlap distributions (boxplots) at different values for the tuning parameter

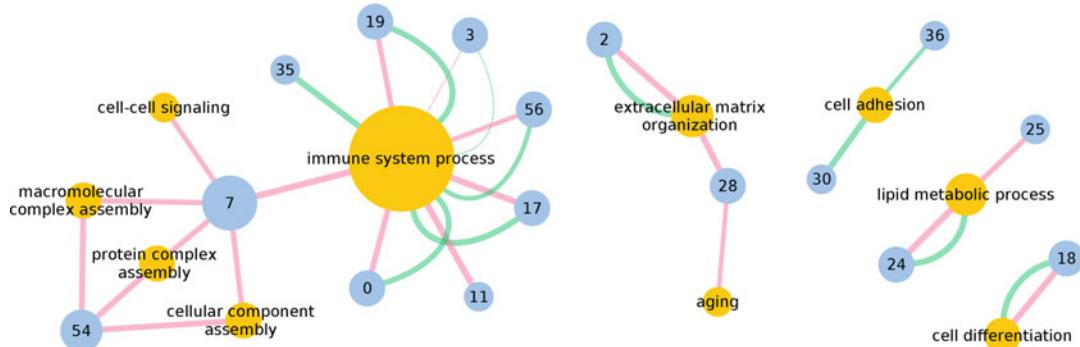


Fig. 4 Enrichment for GO Slim terms in the STAGE AAW-IMA differential module network. Blue nodes are modules, and yellow nodes GO Slim terms. Red and green edges indicate enrichment ($q < 0.05$) in the corresponding AAW and IMA module, respectively. The reassignment threshold used is 0.015

The biological assumption underpinning the differential module network model (Subheading 4.1) is that each module represents a higher-level biological process, or set of processes, that is shared between conditions, whereas the differences in gene assignments reflect differences in molecular pathways that are affected by, or interact with, this higher-level process. To test whether the optimization algorithm accurately captures this biological picture, we first performed gene ontology enrichment (task “go_annotation,” Subheading 3.7) using the GO Slim ontology. GO Slims give a broad overview of the ontology content without the detail of the specific fine-grained terms (<http://www.geneontology.org/page/go-slim-and-subset-guide>). Consistent with our biological assumption, matching modules in atherosclerotic and non-atherosclerotic tissues are often enriched for the same GO Slim categories (Fig. 4).

Next, we performed gene ontology enrichment using the complete, fine-grained ontology, and removed all enrichments that were shared between matching modules. The resulting tissue-specific module enrichments reflected biologically meaningful differences between healthy and diseased arteries (Fig. 5). For instance, clusters 3 and 7 present a strong enrichment in AAW for the regulation of natural killer (NK) cells that augment atherosclerosis by cytotoxic-dependent mechanisms [49]. In IMA, these clusters are predicted to be regulated by genetic variation in CD2, a cell adhesion molecule found on the surface of T and NK cells, whereas in AAW their predicted regulator is BCL2A1, an important cell death regulator and pro-inflammatory gene that is upregulated in coronary plaques compared to healthy controls [50]. This suggests that misregulation of cytotoxic response processes plays a role in the disease, further supported by the overrepresentation in cluster 10 of genes associated with cell death that are an important trigger of plaque rupture [51].

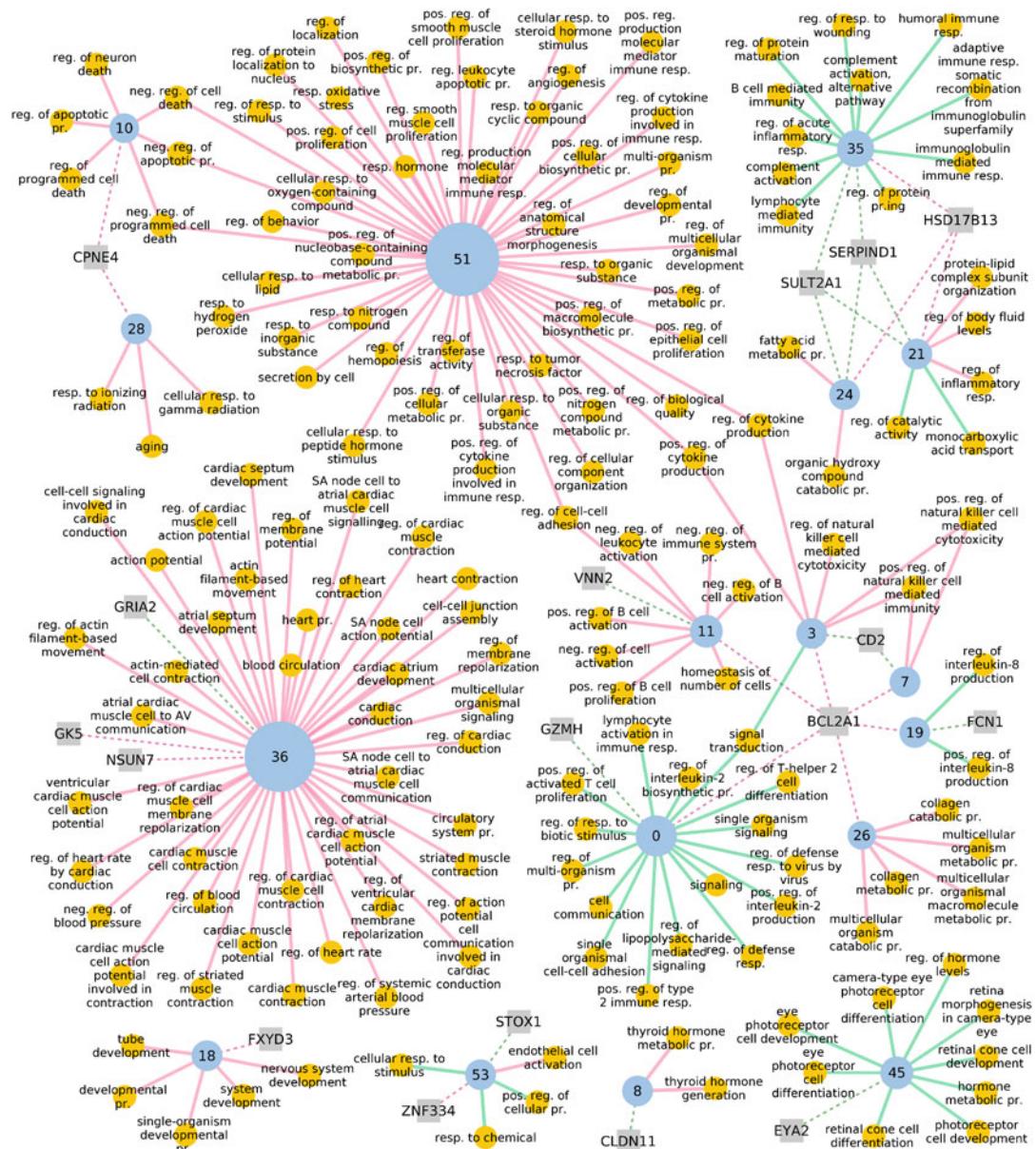


Fig. 5 Tissue-specific GO enrichment for terms related to the immune system process (GO00002376) and regulator assignment in the STAGE AAW-IMA differential module network. Reassignment of nodes was computed with a threshold of 0.015. Blue nodes are modules, yellow nodes GO terms, and gray nodes regulatory genes. Red and green edges indicate tissue-specific enrichment ($q < 0.01$) in the corresponding AAW and IMA module, respectively. Dashed red and green edges indicate regulator assignments in AAW and IMA, respectively. Only top 1% regulators are depicted. neg., negative; pos., positive; pr., process; reg., regulation; resp., response

Furthermore, variations in BCL2A1 are predicted to regulate other clusters exclusively in AAW too, with disease-relevant AAW enrichments. Cluster 11 is associated with the regulation of B lymphocytes, which may attenuate the neointimal formation of atherosclerosis [52], while cluster 26 is enriched for collagen production regulation. Uncontrolled collagen accumulation leads to arterial stenosis, while excessive collagen breakdown combined with inadequate synthesis weakens plaques thereby making them prone to rupture [53]. Last, as expected, terms related with the heart, cardiac muscle, and blood circulation are strongly enriched in AAW, in particular in cluster 36. In AAW, this cluster is regulated by GK5, which plays an important role in fatty acid metabolism and whose upregulation has previously been associated to the pathogenesis of atherosclerosis and cardiovascular disease in patients with autoimmune conditions [54]. On the opposite side, cluster 36 in IMA is regulated by GRIA2, a player in the ion transport pathway, which has been shown to be downregulated in advanced atherosclerotic lesions [55].

In summary, this application has shown that differential module network inference allows to identify sets of one-to-one mapping modules representing broad biological processes conserved between conditions, with biologically relevant differences in fine-grained gene-to-module assignments and upstream regulatory factors.

Acknowledgements

PE and TM are supported by Roslin Institute Strategic Programme funding from the BBSRC [BB/P013732/1].

References

1. Newman MEJ (2006) Modularity and community structure in networks. *Proc Natl Acad Sci U S A* 103:8577–8582
2. Hartwell LH, Hopfield JJ, Leibler S, Murray AW (1999) From molecular to modular cell biology. *Nature* 402:C47–C52
3. Qi Y, Ge H (2006) Modularity and dynamics of cellular networks. *PLoS Comput Biol* 2:e174
4. Eisen MB, Spellman PT, Brown PO, Botstein D (1998) Cluster analysis and display of genome-wide expression patterns. *Proc Natl Acad Sci U S A* 95(25):14863–14868
5. Spellman PT, Sherlock G, Zhang MQ, Iyer VR, Anders K, Eisen MB, Brown PO, Botstein D, Futcher B (1998) Comprehensive identification of cell cycle-regulated genes of the yeast *Saccharomyces cerevisiae* by microarray hybridization. *Mol Biol Cell* 9:3273–3297
6. Segal E, Shapira M, Regev A, Pe'er D, Botstein D, Koller D, Friedman N (2003) Module networks: identifying regulatory modules and their condition-specific regulators from gene expression data. *Nat Genet* 34:166–167
7. Friedman N (2004) Inferring cellular networks using probabilistic graphical models. *Science* 308:799–805
8. Koller D, Friedman N (2009) Probabilistic graphical models: principles and techniques. The MIT Press, Cambridge
9. Lee SI, Pe'er D, Dudley AM, Church GM, Koller D (2006) Identifying regulatory mecha-

- nisms using individual variation reveals key role for chromatin modification. Proc Natl Acad Sci U S A 103:14062–14067
10. Zhang W, Zhu J, Schadt EE, Liu JS (2010) A Bayesian partition method for detecting pleiotropic and epistatic eQTL modules. PLoS Comput Biol 6(1):e1000642
 11. Lee S-I, Dudley AM, Drubin D, Silver PA, Krogan NJ, Pe'er D, Koller D (2009) Learning a prior on regulatory potential from eQTL data. PLoS Genet 5(1):e1000358
 12. Bonnet E, Tatari M, Joshi A, Michoel T, Marchal K, Berx G, Van de Peer Y (2010a) Network inference from a cancer gene expression data set identifies microRNA regulated modules. PLoS One 5:e10162
 13. Bonnet E, Michoel T, Van de Peer Y (2010b) Prediction of a gene regulatory network linked to prostate cancer from gene expression, microRNA and clinical data. Bioinformatics 26:i683–i644
 14. Akavia UD, Litvin O, Kim J, Sanchez-Garcia F, Kotliar D, Causton HC, Pochanard P, Mozes E, Garraway LA, Pe'er D (2010) An integrated approach to uncover drivers of cancer. Cell 143:1005–1017
 15. Bonnet E, Calzone L, Michoel T (2015) Integrative multi-omics module network inference with Lemon-Tree. PLoS Comput Biol 11(2):e1003983
 16. Novershtern N, Regev A, Friedman N (2011) Physical module networks: an integrative approach for reconstructing transcription regulation. Bioinformatics 27(13):i177–i185
 17. Michoel T, De Smet R, Joshi A, Van de Peer Y, Marchal K (2009) Comparative analysis of module-based versus direct methods for reverse-engineering transcriptional regulatory networks. BMC Syst Biol 3:49
 18. Roy S, Lagree S, Hou Z, Thomson JA, Stewart R, Gasch AP (2013) Integrated module and gene-specific regulatory inference implicates upstream signaling networks. PLoS Comput Biol 9(10):e1003252
 19. Segal E, Sirlin CB, Ooi C, Adler AS, Gollub J, Chen X, Chan BK, Matcuk GR, Barry CT, Chang HY, Kuo MD (2007) Decoding global gene expression programs in liver cancer by noninvasive imaging. Nat Biotechnol 25:675–680
 20. Zhu H, Yang H, Owen MR (2007) Combined microarray analysis uncovers self-renewal related signaling in mouse embryonic stem cells. Syst Synth Biol 1:171–181
 21. Li J, Liu ZJ, Pan YC, Liu Q, Fu X, Cooper NG, Li YX, Qiu MS, Shi TL (2007) Regulatory module network of basic/helix-loop-helix transcription factors in mouse brain. Genome Biol 8:R244 (2007)
 22. Novershtern N, Itzhaki Z, Manor O, Friedman N, Kaminski N (2008) A functional and regulatory map of asthma. Am J Respir Cell Mol Biol 38:324–336
 23. Amit I, Garber M, Chevrier N, Leite AP, Donner Y, Eisenhaure T, Guttman M, Grenier JK, Li W, Zuk O, Schubert LA, Birditt B, Shay T, Goren A, Zhang X, Smith Z, Deering R, McDonald RC, Cabili M, Bernstein BE, Rinn JL, Meissner A, Root DE, Hacohen N, Regev A (2009) Unbiased reconstruction of a mammalian transcriptional network mediating pathogen responses. Science 326:257
 24. Vermeirssen V, Joshi A, Michoel T, Bonnet E, Casneuf T, Van de Peer Y (2009) Transcription regulatory networks in *Caenorhabditis elegans* inferred through reverse-engineering of gene expression profiles constitute biological hypotheses for metazoan development. Mol. BioSyst. 5:1817–1830.
 25. Novershtern N, Subramanian A, Lawton LN, Mak RH, Nicholas Haining W, McConkey ME, Habib N, Yosef N, Chang CY, Shay T, et al (2011) Densely interconnected transcriptional circuits control cell states in human hematopoiesis. Cell 144(2):296–309
 26. Zhu M, Deng X, Joshi T, Xu D, Stacey G, Cheng J (2012) Reconstructing differentially co-expressed gene modules and regulatory networks of soybean cells. BMC Genom 13(1):437
 27. Arhondakis S, Bita CE, Perrakis A, Manioudaki ME, Krokida A, Kaloudas D, Kalaitzis P (2016) In silico transcriptional regulatory networks involved in tomato fruit ripening. Front Plant Sci 7:1234
 28. Behdani E, Bakhtiarizadeh MR (2017) Construction of an integrated gene regulatory network link to stress-related immune system in cattle. Genetica 145(4–5):441–454
 29. Marchi FA, Martins DC, Barros-Filho MC, Kuasne H, Lopes AFB, Brentani H, Filho JCST, Guimarães GC, Faria EF, Scapulatempo-Neto C, et al (2017) Multidimensional integrative analysis uncovers driver candidates and biomarkers in penile carcinoma. Sci Rep 7:6707
 30. de la Fuente A (2010) From ‘differential expression’ to ‘differential networking’—identification of dysfunctional regulatory networks in diseases. Trends Genet 26(7):326–333

31. Ideker T, Krogan NJ (2012) Differential network biology. *Mol Syst Biol* 8(1):565
32. Gambardella G, Moretti MN, De Cegli R, Cardone L, Peron A, Di Bernardo D (2013) Differential network analysis for the identification of condition-specific pathway activity and regulation. *Bioinformatics* 29(14):1776–1785
33. Ha MJ, Baladandayuthapani V, Do K-A (2015) DINGO: differential network analysis in genomics. *Bioinformatics* 31(21):3413–3420
34. McKenzie AT, Katsyv I, Song W-M, Wang M, Zhang B (2016) DGCA: A comprehensive r package for differential gene correlation analysis. *BMC Syst Biol* 10(1):106
35. Voigt A, Nowick K, Almaas E (2017). A composite network of conserved and tissue specific gene interactions reveals possible genetic interactions in glioma. *PLOS Comput Biol* 13(9):e1005739
36. Roy S, Wapinski I, Pfiffner J, French C, Socha A, Konieczka J, Habib N, Kellis M, Thompson D, Regev A (2013) Arboretum: reconstruction and analysis of the evolutionary history of condition-specific transcriptional modules. *Genome Res* 23(6):1039–1050
37. Joshi A, De Smet R, Marchal K, Van de Peer Y, Michoel T (2009) Module networks revisited: computational assessment and prioritization of model predictions. *Bioinformatics* 25(4):490–496
38. Segal E, Pe'er D, Regev A, Koller D, Friedman N (2005) Learning module networks. *J Mach Learn Res* 6:557–588
39. Michoel T, Maere S, Bonnet E, Joshi A, Saeys Y, Van den Bulcke T, Van Leemput K, van Remortel P, Kuiper M, Marchal K, Van de Peer Y (2007) Validating module networks learning algorithms using simulated data. *BMC Bioinf* 8:S5
40. Qin ZS (2006) Clustering microarray gene expression data using weighted Chinese restaurant process. *Bioinformatics* 22:1988–1997
41. Joshi A, Van de Peer Y, Michoel T (2008) Analysis of a Gibbs sampler for model based clustering of gene expression data. *Bioinformatics* 24(2):176–183
42. Lu Y, Zhou X, Nardini C (2017) Dissection of the module network implementation “Lemon-Tree”: enhancements towards applications in metagenomics and translation in autoimmune maladies. *Mol BioSyst* 13(10):2083–2091
43. Michoel T, Nachtergael B (2012) Alignment and integration of complex networks by hypergraph-based spectral clustering. *Phys Rev E* 86:056111
44. Maere S, Heymans K, Kuiper M (2005) BiNGO: a cytoscape plugin to assess overrepresentation of gene ontology categories in biological networks. *Bioinformatics* 21:3448–3449
45. Hägg S, Skogsberg J, Lundström J, Noori P, Nilsson R, Zhong H, Maleki S, Shang MM, Brinne B, Bradshaw M, Bajic VB, Samnegard A, Silveira A, Kaplan LM, Gigante B, Leander K, de Faire U, Rosfors S, Lockowandt U, Liska J, Konrad P, Takolander R, Franco-Cereceda A, Schadt EE, Ivert T, Hamsten A, Tegnér J, Björkegren J (2009) Multi-organ expression profiling uncovers a gene module in coronary artery disease involving transendothelial migration of leukocytes and LIM domain binding 2: the Stockholm Atherosclerosis Gene Expression (STAGE) study. *PLoS Genet* 5(12):e1000754
46. Foroughi Asl H, Talukdar H, Kindt A, Jain R, Ermel R, Ruusalepp A, Nguyen K-D, Dobrin R, Reilly D, CARDIoGRAM Consortium, Schunkert H, Samani N, Braenne I, Erdmann J, Melander O, Qi J, Ivert T, Skogsberg J, Schadt EE, Michoel T, Björkegren J (2015) Expression quantitative trait loci acting across multiple tissues are enriched in inherited risk of coronary artery disease. *Circ Cardiovasc Genet* 8:305–315
47. Talukdar H, Foroughi Asl H, Jain R, Ermel R, Ruusalepp A, Franzén O, Kidd B, Readhead B, Giannarelli C, Ivert T, Dudley J, Civelek M, Lusis A, Schadt E, Skogsberg J, Michoel T, Björkegren JLM (2016) Cross-tissue regulatory gene networks in coronary artery disease. *Cell Syst* 2:196–208
48. Schadt EE (2009) Molecular networks as sensors and drivers of common human diseases. *Nature* 461:218–223
49. Selathurai A, Deswaerte V, Kanellakis P, Tipping P, Toh B-H, Bobik A, Kyaw T (2014) Natural killer (NK) cells augment atherosclerosis by cytotoxic-dependent mechanisms. *Cardiovasc Res* 102(1):128–137
50. Sikorski K, Wesoly J, Bluyssen HAR (2014) Data mining of atherosclerotic plaque transcriptomes predicts STAT1-dependent inflammatory signal integration in vascular disease. *Int J Mol Sci* 15(8):14313–14331

51. Martinet W, Schrijvers DM, De Meyer GRY (2011) Pharmacological modulation of cell death in atherosclerosis: a promising approach towards plaque stabilization? *Br J Pharmacol* 164(1):1–13
52. Gjurich BN, Taghavie-Moghadam PL, Ley K, Galkina EV (2014) L-selectin deficiency decreases aortic B1a and Breg subsets and promotes atherosclerosis. *Thromb Haemost* 112(4):803
53. Rekhter MD (1999) Collagen synthesis in atherosclerosis: too much and not enough. *Cardiovasc Res* 41(2):376–384
54. Perez-Sanchez C, Barbarroja N, Messineo S, Ruiz-Limon P, Rodriguez-Ariza A, Jimenez-Gomez Y, Khamashta MA, Collantes-Estevez E, Jose Cuadrado M^a, Angeles Aguirre M^a, et al (2015) Gene profiling reveals specific molecular pathways in the pathogenesis of atherosclerosis and cardiovascular disease in antiphospholipid syndrome, systemic lupus erythematosus and antiphospholipid syndrome with lupus. *Ann Rheum Dis* 74(7): 1441–1449
55. Fu S, Zhao H, Shi J, Abzhanov A, Crawford K, Ohno-Machado L, Zhou J, Du Y, Kuo WP, Zhang J, et al Peripheral arterial occlusive disease: global gene expression analyses suggest a major role for immune and inflammatory responses. *BMC Genomics* 9(1): 369



Chapter 14

Stability in GRN Inference

**Giuseppe Jurman, Michele Filosi, Roberto Visintainer,
Samantha Riccadonna, and Cesare Furlanello**

Abstract

Reconstructing a gene regulatory network from one or more sets of omics measurements has been a major task of computational biology in the last 20 years. Despite an overwhelming number of algorithms proposed to solve the network inference problem either in the general scenario or in an ad-hoc tailored situation, assessing the stability of reconstruction is still an uncharted territory and exploratory studies mainly tackled theoretical aspects. We introduce here empirical stability, which is induced by variability of reconstruction as a function of data subsampling. By evaluating differences between networks that are inferred using different subsets of the same data we obtain quantitative indicators of the robustness of the algorithm, of the noise level affecting the data, and, overall, of the reliability of the reconstructed graph. We show that empirical stability can be used whenever no ground truth is available to compute a direct measure of the similarity between the inferred structure and the true network. The main ingredient here is a suite of indicators, called NetSI, providing statistics of distances between graphs generated by a given algorithm fed with different data subsets, where the chosen metric is the Hamming–Ipsen–Mikhailov (HIM) distance evaluating dissimilarity of graph topologies with shared nodes. Operatively, the NetSI family is demonstrated here on synthetic and high-throughput datasets, inferring graphs at different resolution levels (topology, direction, weight), showing how the stability indicators can be effectively used for the quantitative comparison of the stability of different reconstruction algorithms.

Key words Networks, Network distance, Stability, Network inference

1 Introduction

The quest for an effective algorithm reconstructing the structure of gene regulatory networks from omics data has led, starting from mid-2000s, to a steady flow of published manuscripts introducing novel procedures based on a variety of mathematical and statistical principles (*see* Fig. 1). A number of reviews have appeared on the topic [1–6] even with quantitative comparisons [7–10]. Manuscripts have been also published discussing current limitations [11–16] and relative strengths and disadvantages [17–19], as well as challenges such as the DREAM initiative [20, 21]. The

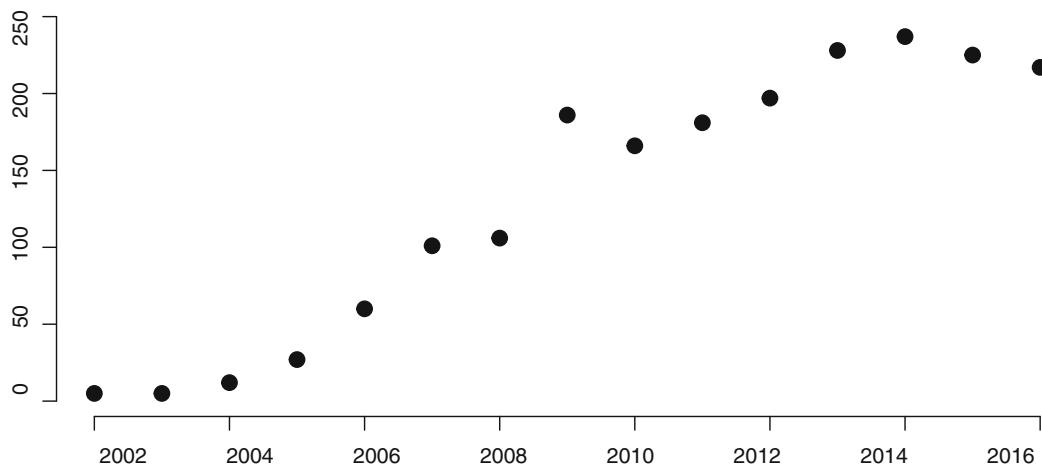


Fig. 1 Number of published papers, per year, with “inference” or “reconstruction” and “gene regulatory network” in title or abstract, as queried on the Scopus portal in November 2017

inference algorithms span a wide range of techniques, from basic solutions as correlation to quite complex methodologies: as an example, this [22–41] is an incomplete list of general-purpose reconstruction methods appeared in the literature between mid-2016 and October 2017. Moreover, inference methods can be tailored to deal with specific organisms or disease [42–46], or specific types of data, e.g., single-cell omics [47–50] or dynamic transcriptomics [51, 52]. The heterogeneity of the methods clearly contributes to make a fair comparison quite hard. An additional confounding factor is given by the different resolution level of reconstruction: while many algorithms aim at inferring the topological backbone of the gene regulatory network (GRN) seen as a simple graph $\bullet - \bullet$, some can deal also with link direction $\bullet \rightarrow \bullet$ and link weight $\bullet \Rightarrow \bullet$. Needless to say, a limited number of algorithms emerge from this zoo for robustness, flexibility, and accuracy and they are still broadly used after several years from their introduction: ARACNE [53], GENIE3 [54], WGCNA [55–57], TIGRESS [58], SIRENE [59], CLR [60] to name a few examples (*see also* <https://omictools.com/gene-regulatory-networks-category>). This given, effective GRN reconstruction is still an open problem [61], although being named as one of the most prominent goals of the post-genomic era and in systems biology [62]. The main reason of the failure is the ill-posedness of the problem, induced by the large number of interactions to discover from the (however) small number of independent and often noisy available measurements [18, 23, 63]. This underdetermination unavoidably results in a relevant number of both false negative and false positive links [64, 65], and thus in a low precision, regardless the approximation level deemed reasonable to achieve [21, 66], to the point that, under some conditions, link predictions are statistically equivalent to random guesses [67]. Furthermore, a

number of studies showed that an analogous of the no-free lunch theorem holds in GRN inference, too: although a small number of algorithms perform well in average, no single method can be deemed the best across different conditions (e.g., data type, data size, network size, noise level, experimental design). This fact supports the conjecture that "it is unlikely that there is one "right" method that fits all different biological, technical and experimental design conditions best" [4, 7]. Throughout all these studies, goodness of reconstruction was assessed by accuracy measures, i.e., a function of the matching/mismatching links between the true network and the inferred structure. Only a limited number of publications dealt with the problem of assessing the robustness of the reconstructed graph when affected by some form of perturbations, i.e., its stability [43, 68–74]. As in the case of profiling, accuracy and stability are distinct but not independent measures to evaluate an algorithm's performance [75]. However, in the case of inference, stability plays an even more crucial role, since the true network is usually unknown (or only partially known) and thus accuracy measures cannot be of help. Here in particular we deal with the issue of assessing the resilience of the inferred graph as a function of data subsampling. Size and quality of the data play a critical role in the inference process [76–79], and quantifying this effect would greatly improve the reproducibility of the whole reconstruction. If a GRN is reconstructed using all the available data or just a portion of them, the two inferred graphs will not coincide: in general, different (subsets) of data correspond to different reconstructed networks. The statistical distribution of such differences yields a quantitative way to characterize both the robustness of the inference algorithm and the quality of data, given a measure of dissimilarity between graphs on shared nodes. The NetSI [80] family of indicators provides this information in terms of the Hamming–Ipsen–Mikhailov (HIM) distance [81, 82]. HIM is a metric with values in the unit interval expressing the difference between graphs built on the same nodes, with possibly directed and weighted edges: $HIM=0$ for coinciding network, while $HIM=1$ when comparing the empty and the full network on shared vertices. Among all the available graph distances, HIM has been selected on the basis of being a good compromise between local (link-based) and global (structure-based) measures of network differences, after a quantitative comparative review [83]. Operatively, the stability indicators collect, for a given ratio of removed data and for a given amount of (bootstrap [84] or cross-validation) resampling, statistics of the mutual distances among all inferred networks and their distances to the network generated by the whole dataset. The rationale is that an algorithm displaying a smaller variability due to data perturbation is likely to be more accurate, and similarly for assessing the noise level of different datasets produced by readings of the same biological regulatory process. Further, other two

indicators describe the distribution of variability of the link weight and node degree across the generated networks, providing a ranked list of the most stable links and nodes, the least variable being the top ranked. In the present chapter, after a recap of the definition and properties of the HIM distance and the NetSI indicators, we show their application on two directed examples (a synthetic toy and an omics dataset) and on a larger but undirected case study. The synthetic example is obtained by the GeneNetWeaver (GNW) generator [85] and consists of a directed network with 33 nodes and 88 links and a dataset of 231 observations produced by the Ordinary/Stochastic Differential Equation multifactorial model (embedded in GNW). The reconstruction is performed by two well-established and one brand new algorithms, namely GeneNet [86, 87], GENIE3 [54], and bnstruct [88]. The task of the first omics example is the reconstruction using the same algorithms of a 33 gene subset \mathcal{G} of the G-protein coupled receptor signaling pathway GO:0007187, using the 233 samples measured on the 47 probes of the Affymetrix Human Genome U95 Version 2 platform corresponding to the 33 genes \mathcal{G} , originally used in [89–92]. The last case study instead shows a different situation, where the target of the inference is the weighted topology of a larger GRN, neglecting directions. Both the network and the expression data are included in the R package c3net [62] <https://cran.r-project.org/web/packages/c3net/index.html>: the graph corresponds to a subnetwork of *E. coli* [93], while the dataset has been generated by SynTRen [94]. Here the algorithms used for the inference are WGCNA, ARACNE, CLR, c3net, bc3net [95], and Topological Overlap Matrix (TOM) [57, 96]. Further technical details and applications are available in [80, 97].

2 Network Inference Stability

Let \mathcal{A} be a reconstruction algorithm using the dataset D to infer the empirical graph N_D , approximation of the true Gene Regulatory Network \mathbf{N} at a certain level of detail (e.g., the simple topological backbone, or also the direction and the strength of the interactions). Let p be the number of nodes of the graphs, and s the number of samples of the dataset D . Given a (dis)-similarity measure δ between networks (e.g., a graph distance), if the true network \mathbf{N} is known, the most natural performance measure for \mathcal{A} is given by the reconstruction accuracy $\delta(\mathbf{N}, N_D)$. On the other hand, the reconstruction stability, which is independent of the knowledge of the gold standard, can be evaluated by means of the four NetSI indicators defined in Fig. 3. The NetSI [80] family evaluates both the resilience of the algorithm \mathcal{A} when less data become available and the noise level in the data, as a function of a data subsampling procedure. In detail, chosen two integers r and n such that $n < s$ and $r < \binom{s}{n}$, r datasets D_i can

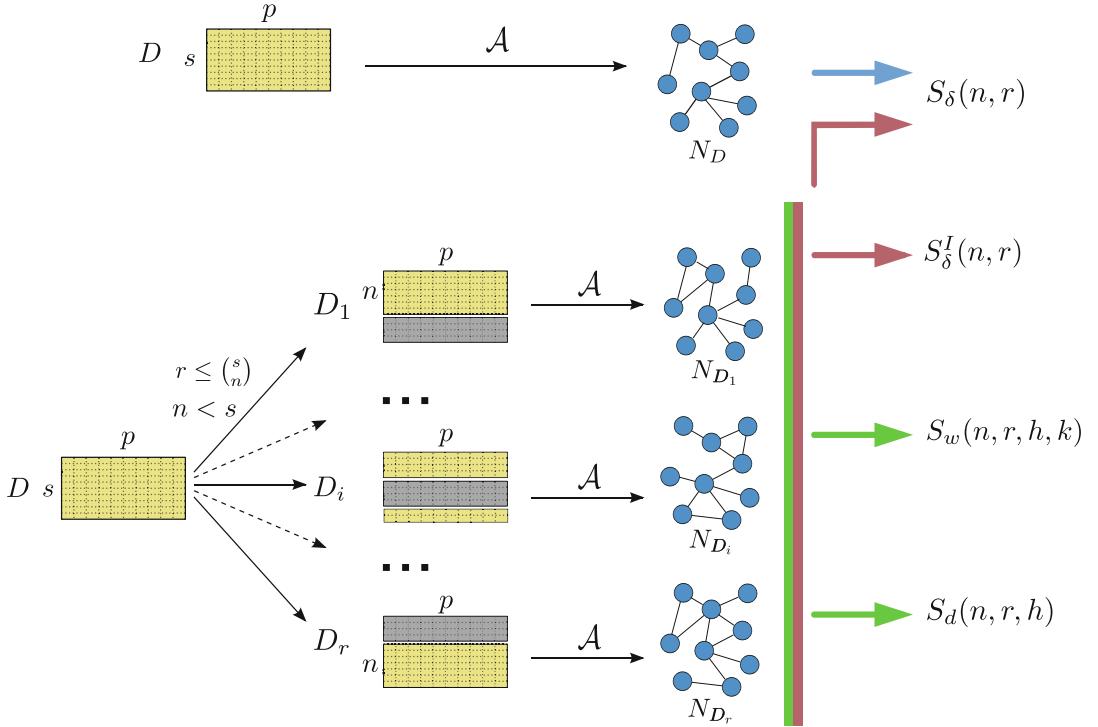


Fig. 2 Resampling schema for the evaluation of the inference stability

be generated by randomly sampling n samples from the original dataset D as shown in Fig. 2. The network N_{D_i} inferred by \mathcal{A} from D_i has adjacency matrix $(\alpha_{hk}^{D_i})$ and its nodes are denoted by $x_h^{D_i}$, for $1 \leq h, k \leq p$. The first indicator S_{δ} is a statistic (mean m and 95% bootstrap t-student confidence interval CI [84]) of the set \mathcal{H}_1 of all the distances between the network inferred on the whole dataset against the networks inferred from the resampled subsets. The second indicator S_{δ}^I is the statistic (m , CI) of the set \mathcal{H}_2 of all mutual distances between the networks inferred from the resampled subsets. These two indicators thus concern the stability of the whole reconstructed network (Fig. 3).

The remaining two indicators, the edge weight stability indicator S_w and the node degree stability indicator S_d , concern instead the stability of the single links and nodes, in terms of mutual variability of their respective weight and degree. In all cases, smaller indicator values correspond to more stable objects. In both cases, a ranked list (of nodes and edges, respectively) can be produced, where the elements are ordered by decreasing stability. If ranked lists L_1 and L_2 need to be compared, the indicator $Ca(L_1, L_2)$, based on the Canberra distance [98, 99], quantifies their differences, both for full lists on the same set of objects and for partial lists [75, 100]. For an effective visualization of the mutual distances between lists, the 2D MultiDimensional Scaling (MDS) [101] is used, a planar projection preserving distances: due

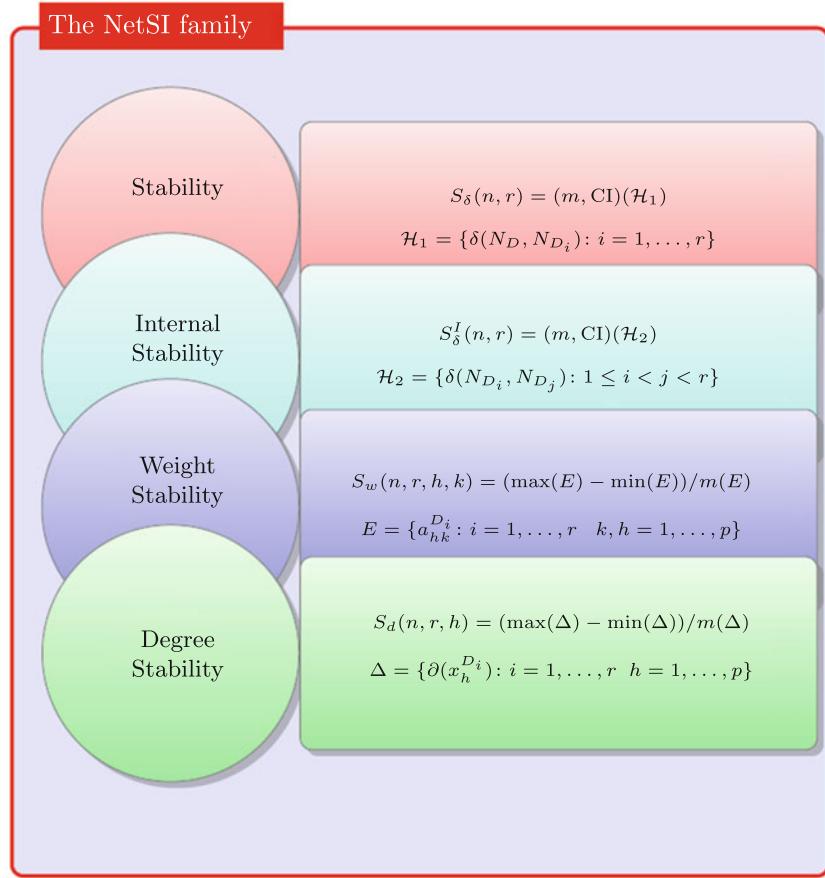


Fig. 3 Definition of the NetSI family. The first two indicators on the top concern the stability on the whole reconstructed network. The last two instead provide a measure of mutual variability of nodes and links. For weighted networks, the weights belong to the unit interval [0,1], and the node degree is defined as the sum of the weights of the edges entering or leaving the node. Depending on the inference algorithm and the used data, weights smaller than a given threshold may be discarded. Full details can be found in [80]

to the nature of the plot, the two dimensions of the plot have no direct relations with the points' features, and thus they are omitted. As the graph similarity measure δ we choose the Hamming–Ipsen–Mikhailov (HIM) distance [81, 82]. This is the product metric of the Hamming H [102–106] and the Ipsen–Mikhailov IM [107] distance:

$$\text{HIM}(\mathcal{N}_1, \mathcal{N}_2) = \frac{1}{\sqrt{2}} \|(\text{H}(\mathcal{N}_1, \mathcal{N}_2), \text{IM}(\mathcal{N}_1, \mathcal{N}_2))\|_2 ,$$

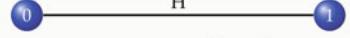
for the two simple networks \mathcal{N}_1 and \mathcal{N}_2 . Notations and mathematical details on the H, IM, and HIM metrics are reported in Fig. 4. Hamming distance is a local metric counting the mismatches between corresponding links while Ipsen–Mikhailov metric is a global distance considering the overall structure of the graph. The

NOTATIONS

- $\mathcal{N}_1, \mathcal{N}_2$ simple networks on p nodes, adjacency matrices $A^{(1)}$ and $A^{(2)}$ on $\mathcal{F} = \mathbb{F}_2 = \{0, 1\}$ or $\mathcal{F} = [0, 1] \subseteq \mathbb{R}$
- Laplacian $L = D - A$, where D is the diagonal matrix with vertex degrees as entries
- $\mathbb{I}_p = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}$, $\mathbb{1}_p = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{pmatrix}$, $0_p = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix}$
- \mathcal{E}_p empty network (adjacency matrix 0_p), \mathcal{F}_p the full graph (adjacency matrix $\mathbb{1}_p - \mathbb{I}_p$)

HAMMING DISTANCE: COUNTING UNMATCHING LINKS

$$\text{Hamming}(\mathcal{N}_1, \mathcal{N}_2) = \sum_{1 \leq i, j \leq p} 1 - \delta_K(a_{ij}^{(1)}, a_{ij}^{(2)})$$

$$H(\mathcal{N}_1, \mathcal{N}_2) = \frac{\text{Hamming}(\mathcal{N}_1, \mathcal{N}_2)}{\text{Hamming}(\mathcal{E}_p, \mathcal{F}_p)} = \frac{1}{p(p-1)} \sum_{1 \leq i, j \leq p} |A_{ij}^{(1)} - A_{ij}^{(2)}|,$$


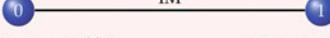
$$A^{(1)} = A^{(2)}$$

$$A^{(1)} + A^{(2)} = \mathbb{1}_p - \mathbb{I}_p = \begin{pmatrix} 0 & 1 & \cdots & 1 \\ 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 0 \end{pmatrix}$$

[Hamming, 1950; Tun et al. 2006]

IPSEN-MIKHAIEV DISTANCE: l_2 INTEGRATED DIFFERENCE OF THE LAPLACIAN SPECTRAL DENSITIES

$$\text{IM}(\mathcal{N}_1, \mathcal{N}_2) = \sqrt{\int_0^\infty [\rho_{\mathcal{N}_1}(\omega, \bar{\gamma}) - \rho_{\mathcal{N}_2}(\omega, \bar{\gamma})]^2 d\omega}$$



$$\text{spec}(L^{(1)}) = \text{spec}(L^{(2)})$$

$$\{\mathcal{N}_1, \mathcal{N}_2\} = \{\mathcal{E}_p, \mathcal{F}_p\}$$

- balls_and_springs net model



$$\ddot{x}_i + \sum_{j=1}^p A_{ij}(x_i - x_j) = 0 \quad \text{for } i = 0, \dots, p-1$$

- squared vibrational frequencies are the Laplacian eigenvalues $\omega_i = \sqrt{\lambda_i}$, for $\lambda_i \in \text{spec}_L = \text{spec}_{D-A}$

- spectral density as sum of Lorentz distributions $\rho(\omega, \gamma) = K \sum_{i=1}^{p-1} \frac{\gamma}{(\omega - \omega_i)^2 + \gamma^2}$

- K normalization constant defined by $\int_0^\infty \rho(\omega, \gamma) d\omega = 1$

- γ common width chosen as $\bar{\gamma}$, the unique solution of $\int_0^\infty [\rho_{\mathcal{E}_p}(\omega, \gamma) - \rho_{\mathcal{F}_p}(\omega, \gamma)]^2 d\omega = 1$

[Ipsen et al., 2002]

HAMMING-IPSEN-MIKHAIEV DISTANCE: NORMALIZED EUCLIDEAN PRODUCT METRIC OF H AND IM

$$\text{HIM}(\mathcal{N}_1, \mathcal{N}_2) = \frac{1}{\sqrt{2}} \sqrt{\text{H}^2(\mathcal{N}_1, \mathcal{N}_2) + \text{IM}^2(\mathcal{N}_1, \mathcal{N}_2)}$$



$$\mathcal{N}_1 = \mathcal{N}_2$$

$$\{\mathcal{N}_1, \mathcal{N}_2\} = \{\mathcal{E}_p, \mathcal{F}_p\}$$

[Jurman et al., 2015]

Fig. 4 Notations and definitions for the H, IM, and HIM metrics

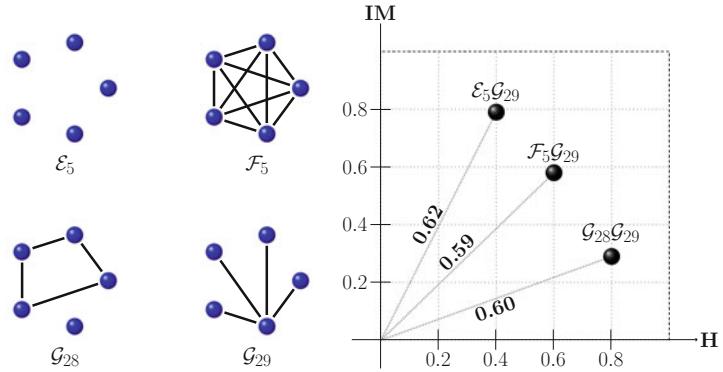


Fig. 5 Four 5-node networks, namely the empty net \mathcal{E}_5 , the full network \mathcal{F}_5 and the graphs \mathcal{G}_{28} , \mathcal{G}_{29} of the graph atlas [115]. The star network \mathcal{G}_{29} has similar HIM distances from the other three graphs, but with different H and IM components: in particular, \mathcal{G}_{29} and \mathcal{G}_{28} are very different in terms of common links, but their spectral structure is relatively similar

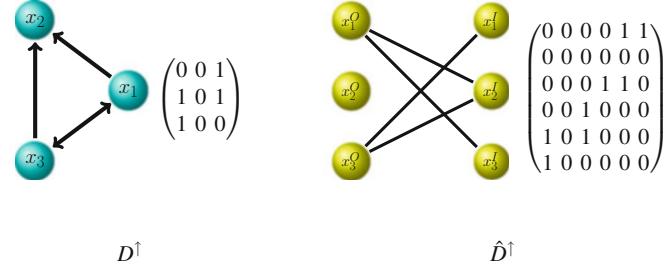


Fig. 6 A directed network D^\uparrow on three nodes and the equivalent undirected network \hat{D}^\uparrow on six nodes, together with their adjacency matrices

glocal metric HIM benefits from the features of both components, also overcoming the issues affecting each of them, resulting in an effective measure already used in different scientific areas [108–114]. In Fig. 5 we display three network comparisons within a set of four 5-nodes networks having close HIM distances, but different H and IM components.

Finally, extension to the directed case follows from the convention in [116], transforming the directed network D^\uparrow into the undirected bipartite \hat{D}^\uparrow . For each node x_i in D^\uparrow , the graph \hat{D}^\uparrow has two nodes x_i^I and x_i^O (where I and O stand for In and Out, respectively) and for each directed link $x_i \rightarrow x_j$ in D^\uparrow there is a link $x_i^O - x_j^I$ in \hat{D}^\uparrow . If the adjacency matrix for D^\uparrow is A_{D^\uparrow} , the corresponding matrix for \hat{D}^\uparrow is $A_{\hat{D}^\uparrow} = \begin{pmatrix} 0 & A_{D^\uparrow}^T \\ A_{D^\uparrow} & 0 \end{pmatrix}$, with respect to the node ordering $x_1^O, x_2^O, \dots, x_n^O, x_1^I, \dots, x_n^I$. An example of the above transformation is shown in Fig. 6. Full mathematical details about the HIM distance are provided in [81, 82]. The HIM distance and the NetSI indicators have been implemented in the open source R package nettools for the CRAN archives, as well as

on GitHub at the address <https://github.com/filosf/nettools.git>. For computing efficiency, the software can be used on multicore workstations and on high performance computing (HPC) clusters.

We present hereafter the NetSI stability analysis for three reconstruction tasks. In the first case, we start from a known network, thus the stability analysis is coupled with the assessment of the reconstruction accuracy; in the remaining two cases, no ground truth is available. In all cases, the experimental setup of the stability analysis is a set of 20 runs of bootstrap Montecarlo (MC) with $K = 1.5, 2, 3, 5$, using $1 - 1/K$ of the original data: thus, for the inference, the portion of data used at each MC run is respectively $1/3, 1/2, 2/3, 4/5$.

3 Case Studies

To indicate that the whole dataset is used in the inference, we use the position $K = \infty$.

3.1 *E. coli* Synthetic Example #1

The modular subnetwork \mathcal{S} with 33 nodes and 88 links shown in Fig. 7 is extracted from the *Escherichia coli* transcriptional regulatory network [117] using the GeneNetWeaver (GNW) simulator [85, 118]. GNW also endows \mathcal{S} with a dynamics using

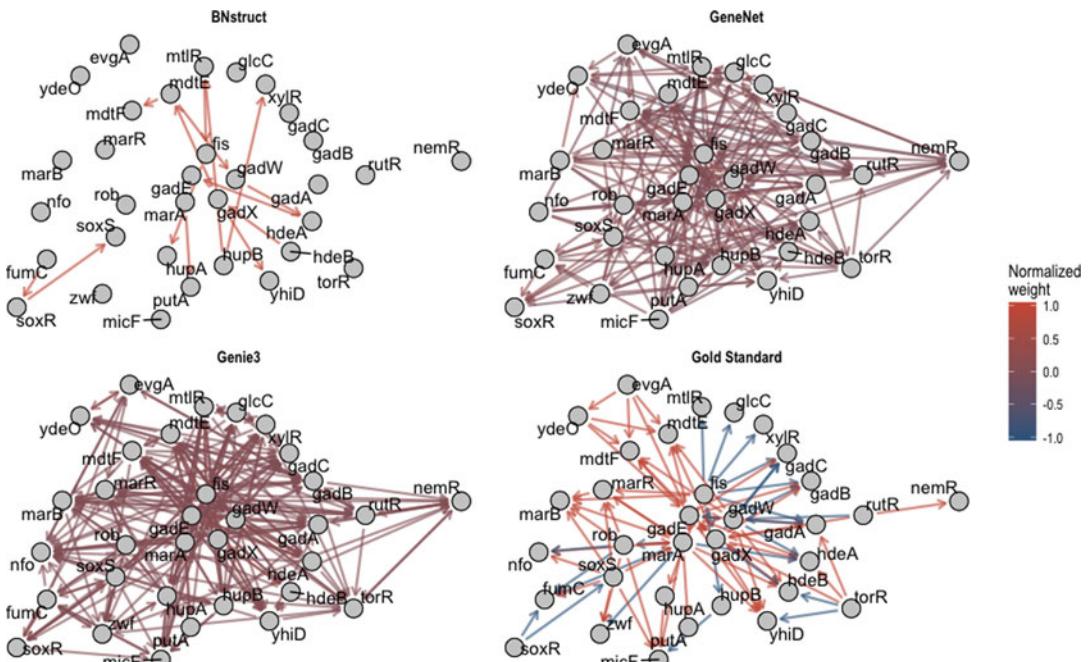


Fig. 7 The true modular subnetwork \mathcal{S} of the *E. coli* transcriptional GRN generated by GNW and same net as reconstructed using the whole dataset by GeneNet, bnstruct, and GENIE3 (d). The weight of each network has been normalized to 1 and the GENIE3 network has been represented after thresholding the weight at the mean weight

a kinetic multifactorial thermodynamical model of transcriptional regulation based on stochastic and ordinary differential equations including both mRNA and protein dynamics: the basal expression level of all the genes in \mathcal{S} is slightly modified and noise is added using a model mimicking microarray noise. A set of 7 simulated omics datasets is generated for a total of 231 measurements of the network dynamics, used as the input for the three reconstruction algorithms GeneNet [86, 87], GENIE3 [54], and bnstruct [88], used in their R/Bioconductor implementation. Since the true network \mathcal{S} is available, first we compute the average HIM distance of the reconstructed networks from the ground truth for each value of K and for each inference method: the plot is shown in Fig. 8 and includes also the HIM distance to nets inferred using all data, corresponding to $K \rightarrow \infty$. GENIE3 clearly outperforms the other two algorithms in terms of reconstruction accuracy. Moreover, the distance between the reconstructed network and the gold standard is decreasing for increasing K as expected for GENIE3 and bnstruct, while is almost constant for GeneNet. A caveat is needed here: the three algorithms are targeted to infer slightly different aspects of the node interactions of the networks. Thus an overall statement of superiority of an algorithm over another cannot be drawn from the considerations made here: these aspects can be used to describe different behaviors of the algorithms on this particular net inferred on the chosen dataset. The scatterplot in Fig. 9 is built plotting the pairs (algorithm, K) and the coordinates of each point are defined by the first two NetSI, namely S on the x -axis and S_I on the y -axis, while in Fig. 10 we show the corresponding boxplots. Interestingly, while for GENIE3 the impact of the data subsampling is quite limited, GeneNet shows a much larger variability on both indices, while bnstruct is quite stable in terms of S_I but less with respect to S . Overall, GENIE3 results the best algorithm for inferring the net \mathcal{S} , as graphically shown in Fig. 11, where the scatterplot with the 12 combinations of algorithm and resampling is represented in the accuracy (1-HIM) and stability ($1 - S$) cartesian plane. Both indicators are consistent in ranking the three algorithms. Finally, we move from the global stability of the whole inferred networks to the local stability of the links and the nodes, evaluated by the indicators S_w and S_d . In Fig. 12 we show the evolution, along different values of K , of the S_d indicators for the three inference algorithms. The trajectories of the node degree stability are quite consistent (monotonically decreasing) for GENIE3, while GeneNet and bnstruct present a larger number of irregularities, due to odd behaviors of the reconstruction on some nodes. Furthermore, for each couple (algorithm, K) the corresponding lists are built, ranked by decreasing stability: the MDS plots in Fig. 13 show the mutual Canberra distance between lists for S_d and S_w , respectively. GENIE3 and bnstruct have the best stability: the lists of nodes and links generated are quite similar (corresponding points are closer

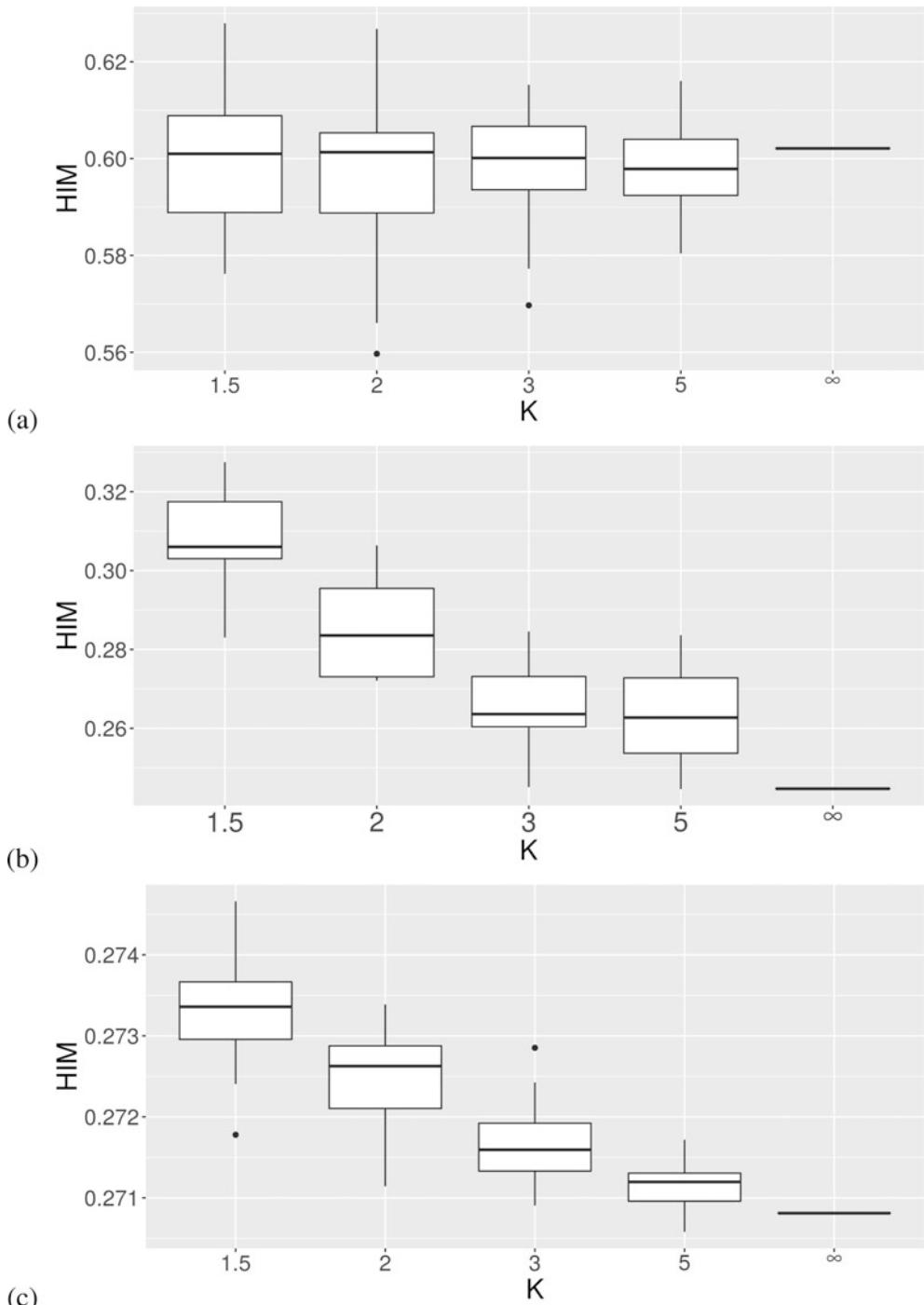


Fig. 8 *E. coli* synthetic example #1. Boxplots of HIM distance with 95% Student bootstrap confidence intervals of the reconstructed networks from the true network \mathcal{S} for the three algorithms GeneNet (a), bnstruct (b), and GENIE3 (c). $K = \infty$ means that the whole dataset has been used for the reconstruction. In this case, no resampling can be done, so there is a unique value without confidence intervals

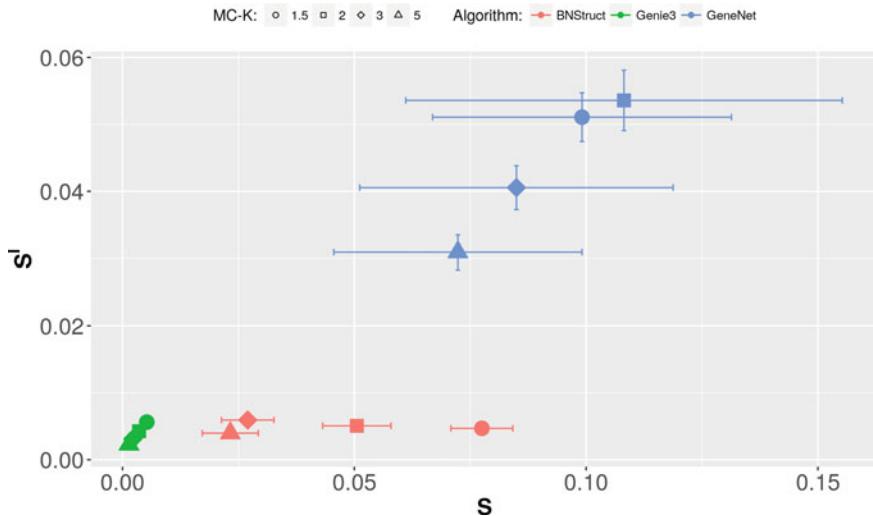


Fig. 9 *E. coli* synthetic example #1. Stability scatterplot for the three algorithms GeneNet, GENIE3, and bnstruct and the four resampling parameters $K = 1.5, 2, 3, 5$ in the cartesian coordinates S and S_i with the corresponding confidence intervals

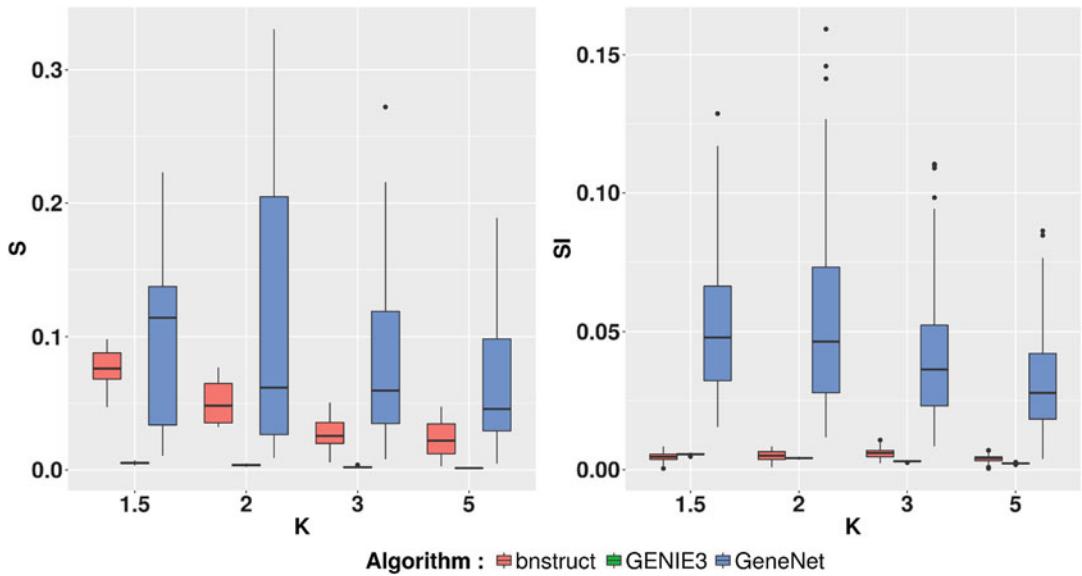


Fig. 10 *E. coli* synthetic example #1. Boxplot of the indicators S (left) and \bar{S}_i (right) for the three algorithms and the four resampling strategies

in the plot) regardless of the subsampling parameter, to which GeneNet is more sensitive, and the corresponding lists are mutually more distant.

Finally, the lists produced by different algorithms, for any value of K , are quite diverse, supporting the fact that different algorithms are targeted to infer relations of different natures: thus, links or nodes which are stably reconstructed by one algorithm are

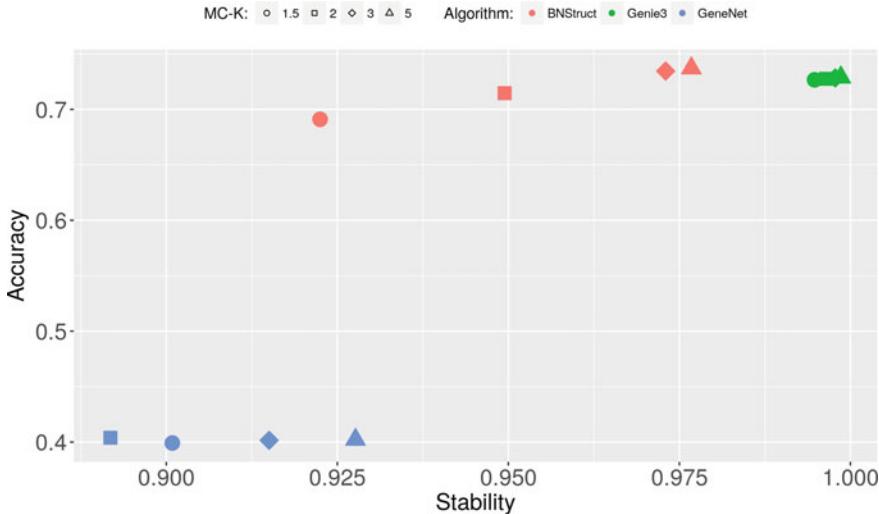


Fig. 11 *E. coli* synthetic example #1. Scatterplot for the three algorithms GeneNet, GENIE3, and bnstruct and the four resampling parameters $K = 1.5, 2, 3, 5$ in the cartesian coordinates, accuracy (1-HIM) and stability ($1 - S$)

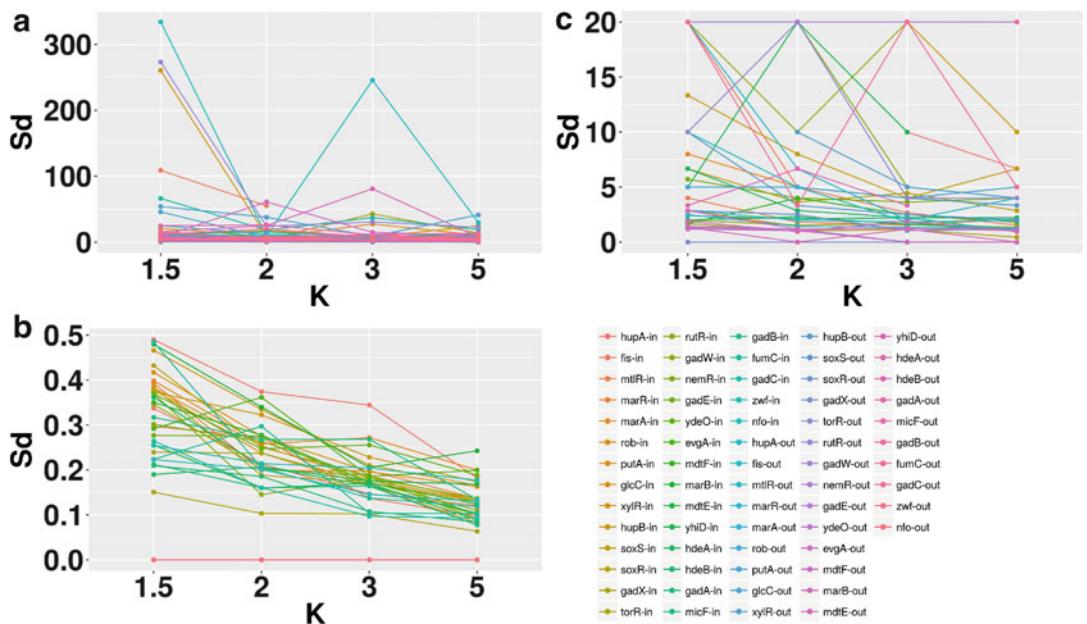


Fig. 12 *E. coli* synthetic example #1. Lineplot of the S_d indicator, for each (in/out) nodes of \mathcal{S} , for the three algorithms GeneNet (a), GENIE3 (b), and bnstruct (c)

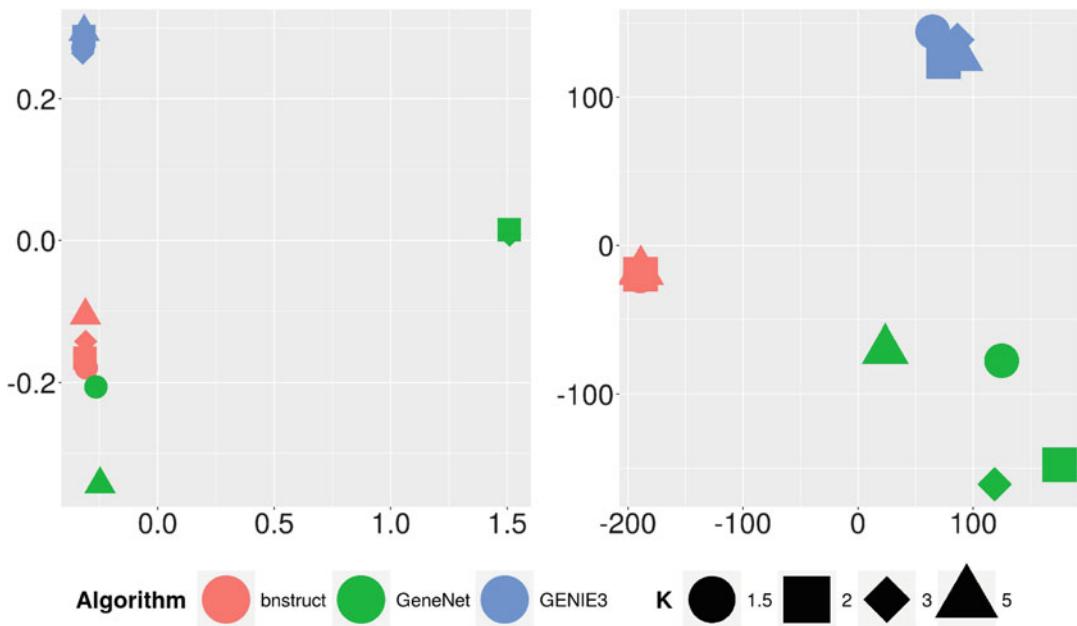


Fig. 13 *E. coli* synthetic example #1. Planar MDS of the Canberra distances between lists ranked by decreasing stability, for each combination of (algorithms, K), for S_d (left) and S_W (right)

not necessarily similarly stable for another method. In particular, if we consider the three top-10 ranked sublists of stablest nodes of each algorithm in the case $K = 5$, their intersection includes only four nodes; the situation is even worse for the links, where there is an empty intersection among the three top-10 ranked sublists of stablest edges.

3.2 The GO:0007187 Pathway

The same three inference algorithms of the previous example are here used to reconstruct a 33 gene subset \mathcal{G} of the G-protein coupled receptor signaling pathway GO:0007187, using the 233 samples measured on the 47 probes of the Affymetrix Human Genome U95 Version 2 platform corresponding to the 33 genes \mathcal{G} , originally used in [89–92]. Here the gold standard is not available, because there is only a limited knowledge of the relations among the participating genes: thus only the NetSI can be used to assess the reliability of the algorithm and the dataset. First we show in Fig. 14 the scatterplot of the different algorithms and resampling on cartesian plane $S \times S_I$. Again, GENIE3 is the method less sensible to the data subsampling, all its points are mutually close and with very small values of both indicators. On the other extreme, GeneNet points are quite distant and with large values of both S and S_I , also with large confidence intervals, as confirmed by the different view offered by the boxplots in Fig. 15. As expected, all indicators are decreasing for increasing K . Finally, we consider the local stability in Fig. 16 and in the MDS in Fig. 17. The two plots are very similar to the corresponding figures for the synthetic example, and the same conclusions can be drawn.

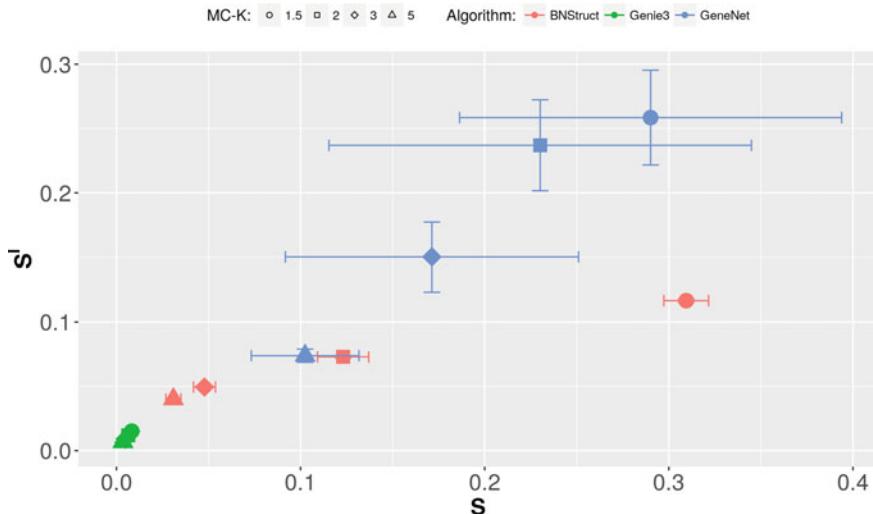


Fig. 14 GO:0007187 pathway. Stability scatterplot for the three algorithms GeneNet, GENIE3, and bnstruct and the four resampling parameters $K = 1.5, 2, 3, 5$ in the cartesian coordinates S and S_i with the corresponding confidence intervals

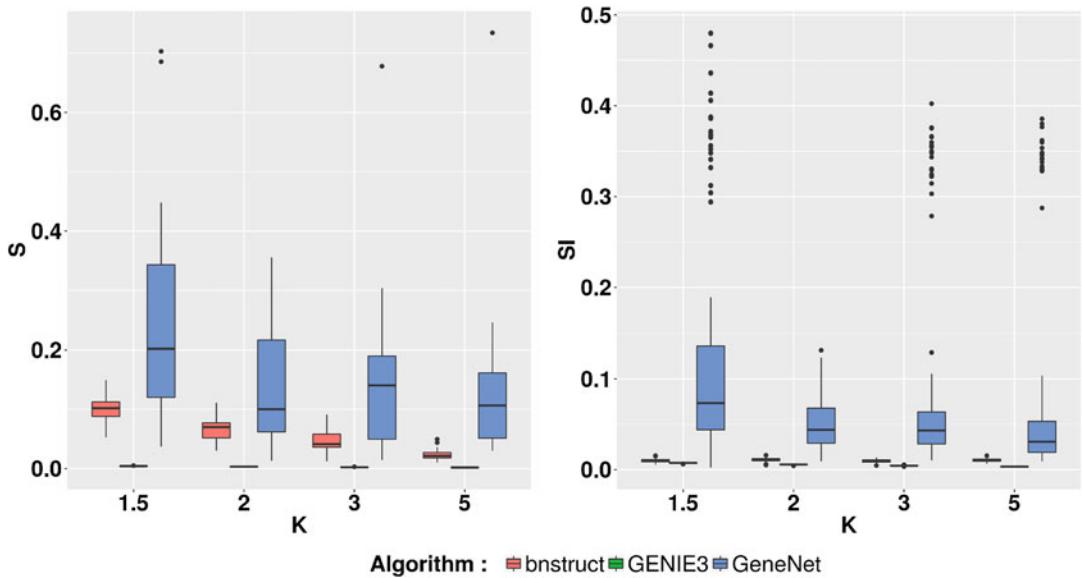


Fig. 15 GO:0007187 pathway. Boxplot of the indicators S (left) and S_i (right) for the three algorithms and the four resampling strategies

Similar considerations hold for the tiny overlap between top-10 lists generated by different algorithms, where only six nodes and two links are in common between at least two lists for $K = 5$.

3.3 *E. coli* Synthetic Example #2

A subnetwork \mathcal{E} with 400 genes and 898 undirected weighted edges is extracted in [93] from the transcriptional GRN of *Escherichia coli* and the SynTRen [94] simulator is used to endow \mathcal{E} with a non-linear dynamics based on Michaelis–Menten and Hill

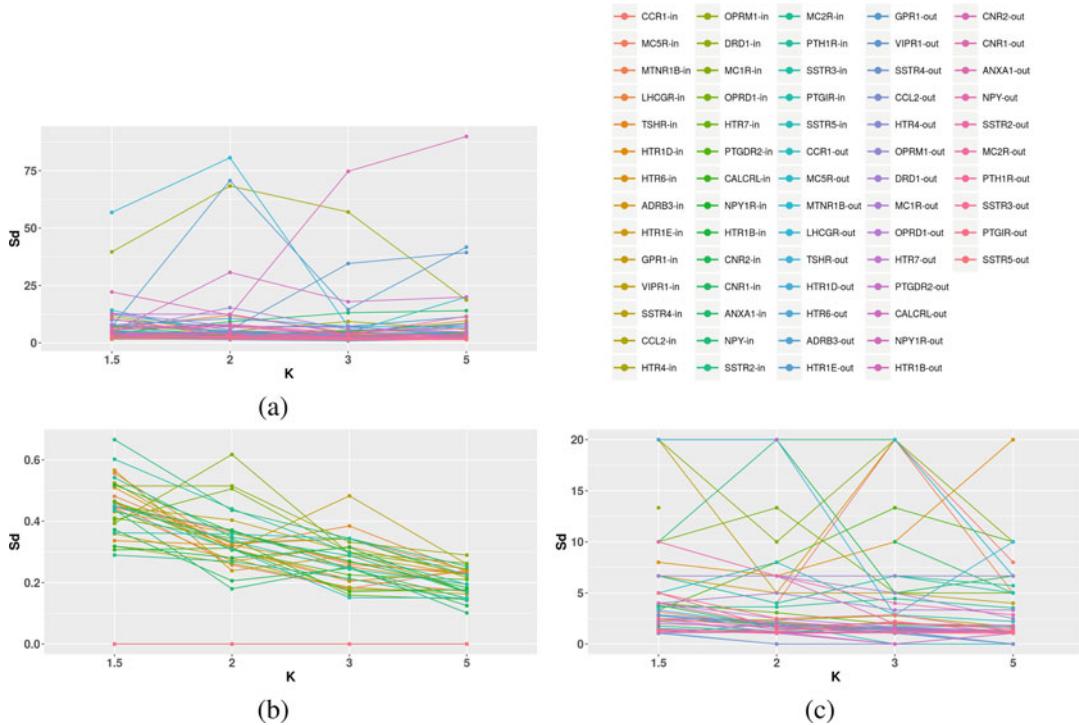


Fig. 16 GO:0007187 pathway. Lineplot of the S_d indicator, for each (in/out) nodes of \mathcal{S} , for the three algorithms GeneNet (a), GENIE3 (b), and bnstruct (c)

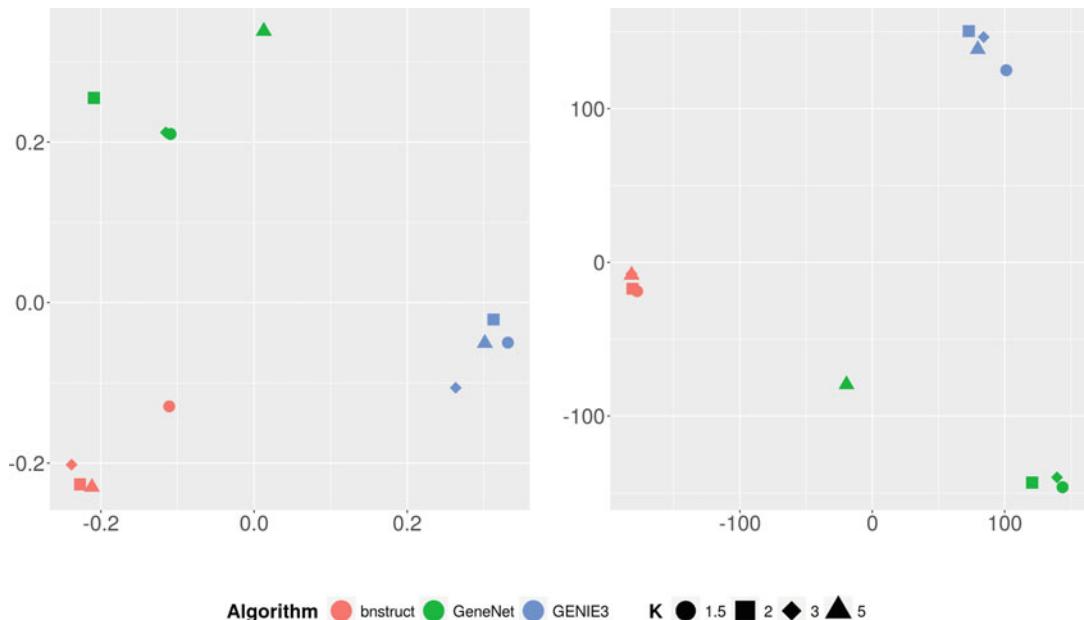


Fig. 17 GO:0007187 pathway. Planar MDS of the Canberra distances between lists ranked by decreasing stability, for each combination of (algorithms, K), for S_d (left) and S_w (right)

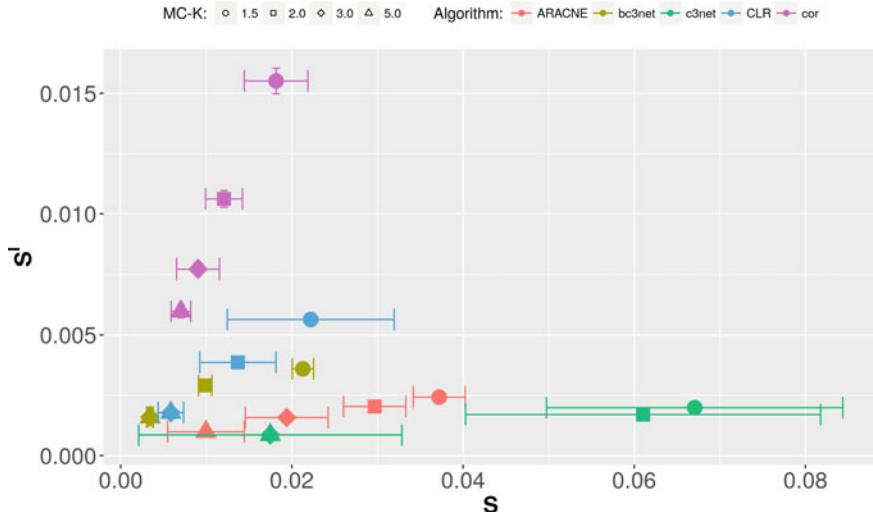


Fig. 18 *E. coli* synthetic example #2. Stability scatterplot for the six algorithms WGCNA, ARACNE, CLR, c3net, bc3net, and TOM and the four resampling parameters $K = 1.5, 2, 3, 5$ in the cartesian coordinates S and S_I with the corresponding confidence intervals

enzyme kinetic equations, producing a synthetic expression dataset with 800 steady-state samples. The R package c3net includes the full expression dataset, but only a partial version \mathcal{E}' of the true network, with only a limited set of links and without weights. The original network \mathcal{E} (as referenced in the R package) is not available anymore, and thus we consider the gold standard as not known.

In what follows, the six inference algorithms WGCNA (here denoted by cor), ARACNE, CLR, c3net, bc3net, and TOM are compared on the basis of their stability in the reconstruction of the network \mathcal{E} from the synthetic expression dataset, in the same experimental setup with 20 replicates of bootstrap Montecarlo with $K = 1.5, 2, 3, 5$. As in the previous case, no gold standard is available, thus the stability is the only quantitative criterion for evaluating performance on this task. In Fig. 18 the scatterplot shows the 24 combinations of algorithm and value of K , with the corresponding 95% Student bootstrap confidence intervals, while in Fig. 19 the same data are portrayed under a different point of view. Here ARACNE and bc3net are the stablest algorithms, with c3net displaying a very peculiar behavior: for $K = 1.5, 2, 3$ its S value (i.e., the distance with the net inferred on the whole dataset) is high and very variable, while its S_I indicator is quite small and stable. We conclude the analysis with an overview of the local stability of the six algorithms in terms of links' weights and nodes' degree: for each pair (algorithm, K) the corresponding lists of links and nodes ranked by decreasing stability are mutually compared using the Canberra list distance indicator $\text{Ca}(\cdot, \cdot)$, and the distances projected on the plane through a 2D MultiDimensional Scaling,

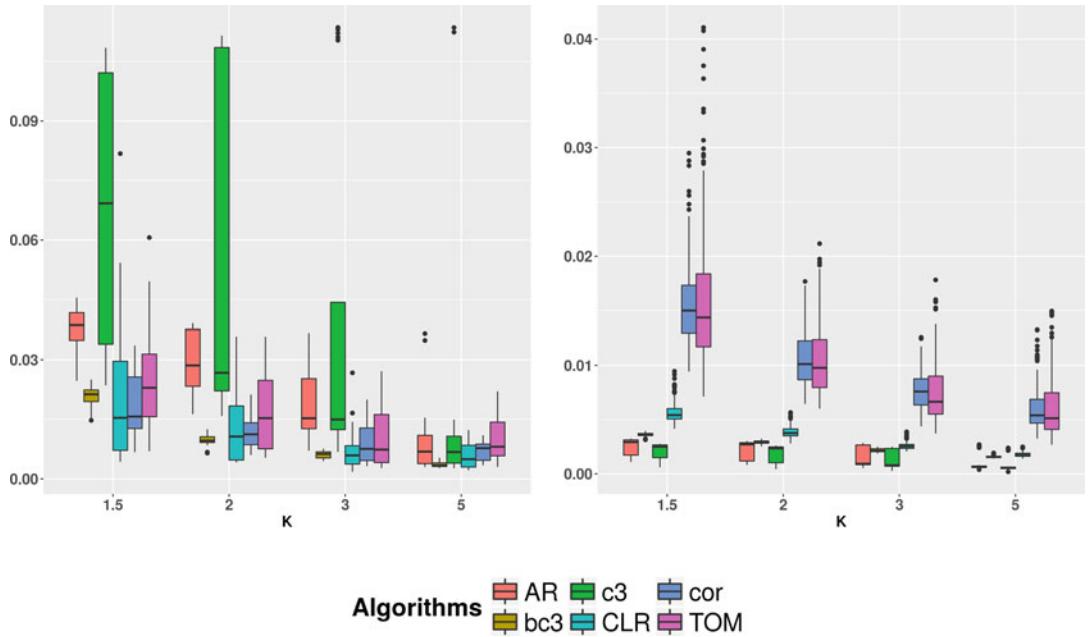


Fig. 19 *E. coli* synthetic example #2. Boxplot of the indicators S (left) and S_i (right) for the six algorithms and the four resampling strategies

reported in Fig. 20. In this case we face a slightly different situation than the previous examples: the ranked lists of nodes do not cluster according to the inference algorithm, apart (partially) for the lists produced by c3net, thus no similarity can be claimed. In particular, among the six top-10 ranked lists of stable nodes, three elements are in common among three lists and 15 nodes are in common among two lists for $K = 5$. For edges instead, the situation is more similar to the previous cases, where lists produced by same algorithms are closer, although some methods tend to cluster together, e.g., ARACNE, bc3net, and c3net in one group and TOM and WGCNA on another group. Furthermore, distances between groups are quite large, due to the fact that these lists are very long (79800 elements). Moreover, only two edges are in common between two top-10 lists for $K = 5$. Finally, here ARACNE and c3net generate lists that are less sensible to the subsampling.

3.4 Studies in Differential Network Analysis

We conclude observing that the study of stability is mostly effective within differential network analysis, adding a crucial discriminating criterion. This added value has been demonstrated in a number of examples shown in [80, 119], e.g., in oncogenomics and neurogenomics. In particular, in [80] the stability analysis is used together with classification and feature selection on a human miRNA array dataset of hepatocellular carcinoma with 480 samples labeled by the two phenotypes sex and disease status; the target

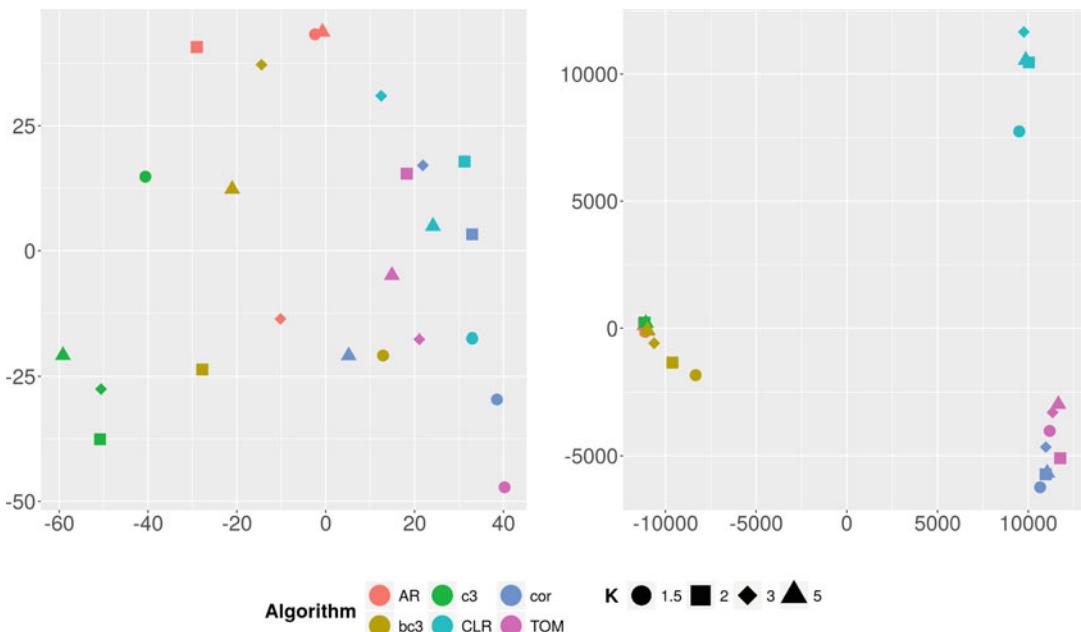


Fig. 20 *E. coli* synthetic example #2. Planar MDS of the Canberra distances between lists ranked by decreasing stability, for each combination of (algorithms, K), for S_d (left) and S_w (right)

was the inference of the network of the interactions among the 240 miRNAs separately for the four combinations of the two phenotypes. By using the NetSI, the best inferring method is detected, together with the data subset leading to the stablest reconstruction.

References

1. Kaderali L, Radde N (2008) Inferring gene regulatory networks from expression data. *Stud Comput Intell* 94:33–74
2. Oates C, Mukherjee S (2012) Network inference and biological dynamics. *Ann Appl Stat* 6(3):1209–1235
3. Noor A, Serpedin E, Nounou M, Nounou H, Mohamed N, Chouchane L (2013) An overview of the statistical methods used for inferring gene regulatory networks and protein-protein interaction networks. *Adv Bioinf* 2013: 12 pp. Article ID 953814
4. Emmert-Streib F, Dehmer M, Haibe-Kains B (2014) Gene regulatory networks and their applications: understanding biological and medical problems in terms of networks. *Front Cell Dev Biol* 2:38
5. Chai L, Loh SK, Low ST, Mohamad MS, Deris S, Zakaria Z (2014) A review on the computational approaches for gene regulatory network construction. *Comput Biol Med* 48(Supplement C):55–65
6. Banf M, Rhee SY (2017) Computational inference of gene regulatory networks: approaches, limitations and opportunities. *Biochim Biophys Acta (BBA) Gene Regul Mech* 1860(1):41–52
7. Bellot P, Olsen C, Salembier P, Oliveras-Vergés A, Meyer PE (2015) NetBenchmark: a bioconductor package for reproducible benchmarks of gene regulatory network inference. *BMC Bioinf* 16(1):312
8. de Matos Simoes R, Dehmer M, Emmert-Streib F (2014) B-cell lymphoma gene regulatory networks: biological consistency among inference methods. *Front Genet* 4:281
9. Thompson D, Regev A, Roy S (2015) Comparative analysis of gene regulatory networks:

- from network reconstruction to evolution. *Ann Rev Cell Dev Biol* 31(1):399–428
10. Wang JTL (2015) Inferring gene regulatory networks: challenges and opportunities. *J Data Mining Genom Proteom* 6:e118
 11. Phenix H, Perkins T, Kærn M (2013) Identifiability and inference of pathway motifs by epistasis analysis. *Chaos* 23:025103
 12. Ud-Dean SMM (2016) Inferability and inference of gene regulatory networks. PhD thesis, ETH Zürich
 13. Baralla A, Mentzen W, de la Fuente A (2009) Inferring gene networks: dream or nightmare? *Ann N Y Acad Sci* 1158:246–256
 14. Haibe-Kains B, Emmert-Streib F (2014) Quantitative assessment and validation of network inference methods in bioinformatics. *Front Genet* 5:221
 15. Altay G, Emmert-Streib F (2010) Revealing differences in gene network inference algorithms on the network level by ensemble methods. *Bioinformatics* 26(14):1738–1744
 16. Krishnan A, Giuliani A, Tomita M (2007) Indeterminacy of reverse engineering of gene regulatory networks: the curse of gene elasticity. *PLoS One* 2(6):e562
 17. Madhamshettiar P, Maetschke S, Davis M, Reverter A, Ragan M (2012) Gene regulatory network inference: evaluation and application to ovarian cancer allows the prioritization of drug targets. *Genome Med* 4(5):41
 18. De Smet R, Marchal K (2010) Advantages and limitations of current network inference methods. *Nat Rev Microbiol* 8(10):717–729
 19. Pataskar A, Tiwari VK (2016) Computational challenges in modeling gene regulatory events. *Transcription* 7(5):188–195
 20. Marbach D, Prill RJ, Schaffter T, Mattiussi C, Floreano D, Stolovitzky G (2010) Revealing strengths and weaknesses of methods for gene network inference. *Proc Natl Acad Sci U S A* 107(14):6286–6291
 21. Marbach D, Costello J, Kuffner R, Vega N, Prill R, Camacho D, Allison K, Kellis M, Collins J, Stolovitzky G (2012) Wisdom of crowds for robust gene network inference. *Nat Methods* 9(8):796–804
 22. Fan Y, Wang X, Peng Q (2017) Inference of gene regulatory networks using Bayesian non-parametric regression and topology information. *Comput Math Methods Med* 2017:Article ID 8307530
 23. Guo S, Jiang Q, Chen L, Guo D (2016) Gene regulatory network inference using PLS-based methods. *BMC Bioinform* 17(1):545
 24. Liang X, Young WC, Hung LH, Raftery AE, Yeung KY (2017) Integration of multiple data sources for gene network inference using genetic perturbation data. *bioRxiv* 158394
 25. Liu F, Zhang SW, Guo WF, Wei ZG, Chen L (2016) Inference of gene regulatory network based on local Bayesian networks. *PLoS Comput Biol* 12(8):e1005024
 26. Liu W, Zhu W, Liao B, Chen H, Ren S, Cai L (2017) Improving gene regulatory network structure using redundancy reduction in the MRNET algorithm. *R Soc Chem Adv* 7:23222–23233
 27. Pachkov M, Balwierz PJ, Arnold P, Gruber AJ, Zavolan M, van Nimwegen E (2017) ISMARA: completely automated inference of gene regulatory networks from high-throughput data. *PeerJ Preprints* 5:e3328v1
 28. Petralia F, Wang P, Yang J, Tu Z (2015) Integrative random forest for gene regulatory network inference. *Bioinformatics* 31(12):i197–i205
 29. Shao B, Lavesson N, Boeva V, Shahzadim RK (2016) A mixture-of-experts approach for gene regulatory network inference. *Int J Data Mining Bioinf* 14(3):258–275
 30. Tjärnberg A, Morgan DC, Studham M, Nordling TEM, Sonnhammer ELL (2017) GeneSPIDER - gene regulatory network inference benchmarking with controlled network and data properties. *Mol BioSyst* 13:1304–1312
 31. Wang J, Hu Y, Li C, Yao JC (2017) Linear convergence of CQ algorithms and applications in gene regulatory network inference. *Inverse Probl* 33(5):055017
 32. Wu J, Zhao X, Lin Z, Shao Z (2016) Large scale gene regulatory network inference with a multi-level strategy. *Mol BioSyst* 12:588–597
 33. Yu B, Xu JM, Li S, Chen C, Chen RX, Wang L, Zhang Y, Wang MH (2017) Inference of time-delayed gene regulatory networks based on dynamic Bayesian network hybrid learning method. *Oncotarget* 8(46):80373–80392
 34. Sanchez - Castillo M, Blanco D, Tienda - Luna IM, Carrion MC, Huang Y (2017) A Bayesian framework for the inference of gene regulatory networks from time and pseudo-time series data. *Bioinf Adv articles*:bt605

35. Barreto NM, Machado KS, Werhli AV (2017) Inference of regulatory networks with MCMC sampler guided by mutual information. In: Proceedings of the symposium on applied computing (SAC 2017). ACM, New York, pp 18–23
36. Carré C, Mas A, Krouk G (2017) Reverse engineering highlights potential principles of large gene regulatory network design and learning. *npj Syst Biol Appl* 3:17
37. Zarayeneh N, Oh JH, Kim D, Liu C, Gao J, Suh SC, Kang M (2016) Integrative gene regulatory network inference using multi-omics data. In: Proceedings of the IEEE international conference on bioinformatics and biomedicine (BIBM 2016). IEEE, Piscataway, pp 1336–1340
38. Banf M, Rhee SY (2017) Enhancing gene regulatory network inference through data integration with Markov random fields. *Sci Rep* 7:41174
39. Ud-Dean SMM, Heise S, Klamt S, Gunawan R (2016) TRaCE+: ensemble inference of gene regulatory networks from transcriptional expression profiles of gene knock-out experiments. *BMC Bioinf* 17(1):252
40. Altarawy D, Eid FE, Heath LS (2017) PEAK: integrating curated and noisy prior knowledge in gene regulatory network inference. *J Comput Biol* 24(9):863–873
41. Koike CY, Higa CHA (2016) Inference of gene regulatory networks using coefficient of determination, Tsallis entropy and biological prior knowledge. In: Proceedings of the IEEE international conference on bioinformatics and bioengineering (BIBE 2016). IEEE, Piscataway, pp 64–70
42. de Luis Balaguer MA, Fisher AP, Clark NM, Fernandez-Espinosa MG, Moeller BK, Weijers D, Lohmann JU, Williams C, Lorenzo O, Sozzani R (2017) Predicting gene regulatory networks by combining spatial and temporal gene expression data in *Arabidopsis* root stem cells. *Proc Natl Acad Sci* 114(36):E7632–E7640
43. Buceta J, Herranz H, Canela-Xandri O, Reigada R, Sagués F, Milán M (2007) Robustness and stability of the gene regulatory network involved in DV boundary formation in the *Drosophila* wing. *PLoS One* 2(7):e602
44. Iglesias-Martinez LF, Kolch W, Santra T (2016) BGRMI: a method for inferring gene regulatory networks from time-course gene expression data and its application in breast cancer research. *Sci Rep* 6:37140
45. Kim D, Kang M, Biswas A, Liu C (2016) Integrative approach for inference of gene regulatory networks using Lasso-based random featuring and application to psychiatric disorders. Faculty Publications 3909, Kennesaw State University
46. Ni Y, Aghamirzaie D, Elmarakeby H, Collakova E, Li S, Grene R, Heath LS (2016) A machine learning approach to predict gene regulatory networks in seed development in *Arabidopsis*. *Front Plant Sci* 7:1936
47. Aibar S, Bravo González-Blas C, Moerman T, Wouters J, Huynh-Thu VA, Imrichová H, Kalender Atak Z, Hulselmans G, Dewaele M, Rambow F, Geurts P, Aerts J, Marine JC, van den Oord J, Aerts S (2017) SCENIC: Single-Cell Regulatory Network Inference and Clustering. *bioRxiv* 144501
48. Chan TE, Stumpf MPH, Babtie AC (2017) Gene regulatory network inference from single-cell data using multivariate information measures. *Cell Syst* 5(3):251–267.e3
49. Gao NP, Ud-Dean SMM, Gandrillon O, Gunawan R (2017) SINCERITIES: inferring gene regulatory networks from timestamped single cell transcriptional expression profiles. *Bioinf Adv*. <https://doi.org/10.1093/bioinformatics/btx575>
50. Herbach U, Bonnaffoux A, Espinasse T, Gandrillon O (2017) Inferring gene regulatory networks from single-cell data: a mechanistic approach. *arXiv:1705.03407*
51. Hillenbrand P, Maier KC, Cramer P, Gerland U (2016) Inference of gene regulation functions from dynamic transcriptome data. *eLife* 5:e12188
52. Desai J, Sartor RC, Lawas LM, Jagadish KSV, Doherty CJ (2017) Improving gene regulatory network inference by incorporating rates of transcriptional change. *bioRxiv* 093807
53. Margolin A, Nemenman I, Bassi K, Wiggins C, Stolovitzky G, Dalla Favera R, Califano A (2006) ARACNE: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. *BMC Bioinf* 7(7):S7
54. Huynh-Thu VA, Irrthum A, Wehenkel L, Geurts P (2010) Inferring regulatory networks from expression data using tree-based methods. *PLoS One* 5(9):e12776

55. Horvath S (2011) Weighted network analysis: applications in genomics and systems biology. Springer, New York
56. Langfelder P, Horvath S (2008) WGCNA: an R package for weighted correlation network analysis. *BMC Bioinf* 9(1):559
57. Zhang B, Horvath S (2005) A general framework for weighted gene co-expression network analysis. *Stat Appl Genet Mol Biol* 4(1):Article 17
58. Haury AC, Mordelet F, Vera-Licona P, Vert JP (2012) TIGRESS: Trustful Inference of Gene REGulation using Stability Selection. *BMC Syst Biol* 6(1):145
59. Mordelet F, Vert JP (2008) SIRENE: supervised inference of regulatory networks. *Bioinformatics* 24(16):i76–i82
60. Faith J, Hayete B, Thaden J, Mogno I, Wierzbowski J, Cottarel G, Kasif S, Collins J, Gardner T (2007) Large-scale mapping and validation of *Escherichia coli* transcriptional regulation from a compendium of expression profiles. *PLoS Biol* 5(1):e8
61. Szederkenyi G, Banga J, Alonso A (2011) Inference of complex biological networks: distinguishability issues and optimization-based solutions. *BMC Syst Biol* 5(1):177
62. Altay G, Emmert-Streib F (2010) Inferring the conservative causal core of gene regulatory networks. *BMC Syst Biol* 4(1):132
63. Hecker M, Lambeck S, Toepfer S, van Someren E, Guthke R (2009) Gene regulatory network inference: data integration in dynamic models – a review. *Biosystems* 96(1):86–103
64. Kamburov A, Stelzl U, Herwig R (2012) Intscore: a web tool for confidence scoring of biological interactions. *Nucleic Acids Res* 40(W1):W140–W146
65. Feizi S, Marbach D, Médard M, Kellis M (2013) Network deconvolution as a general method to distinguish direct dependencies in networks. *Nat Biotechnol* 31(8):726–733
66. Meyer P, Alexopoulos L, Bonk T, Califano A, Cho C, de la Fuente A, de Graaf D, Hartemink A, Hoeng J, Ivanov N, Koepl H, Linding R, Marbach D, Norel R, Peitsch M, Rice J, Royyuru A, Schacherer F, Sprengel J, Stolle K, Vitkup D, Stolovitzky G (2011) Verification of systems biology research in the age of collaborative competition. *Nat Biotechnol* 29(9):811–815
67. Prill R, Marbach D, Saez-Rodriguez J, Sorger P, Alexopoulos L, Xue X, Clarke N, Altan-Bonnet G, Stolovitzky G (2010) Towards a rigorous assessment of systems biology models: the DREAM3 challenges. *PLoS One* 5(2):e9202
68. Larvie JE, Gorji MS, Homaifar A (2015) Inferring stable gene regulatory networks from steady-state data. In: 2015 41st annual Northeast Biomedical Engineering Conference (NEBEC), pp 1–2
69. Larvie J, Sefidmazgi MG, Homaifar A, Harrison SH, Karimoddini A, Guiseppe-Elie A (2016) Stable gene regulatory network modeling from steady-state data. *Bioengineering* 3:12
70. Luo Q, Gong Y, Jia C (2017) Stability of gene regulatory networks with Lévy noise. *Sci China Inf Sci* 60(7):072204
71. Manshaei R, Kyan M (2013) Sparse and stable reconstruction of genetic regulatory networks using time series gene expression data. In: Proceedings of the international conference on bioinformatics, computational biology and biomedical informatics, BCB'13. ACM, New York, pp 710:710–710:711
72. Michailidis G, d'Alché-Buc F (2013) Autoregressive models for gene regulatory network inference: sparsity, stability and causality issues. *Math Biosci* 246(2):326–334
73. Rajapakse JC, Mundra PA (2011) Stability of building gene regulatory networks with sparse autoregressive models. *BMC Bioinf* 12(13):S17
74. Ugander J (2008) Delay-dependent stability of genetic regulatory networks. Master's thesis, Lund University
75. Jurman G, Merler S, Barla A, Paoli S, Galea A, Furlanello C (2008) Algebraic stability indicators for ranked lists in molecular profiling. *Bioinformatics* 24(2):258–264
76. Logsdon B, Mezey J (2010) Gene expression network reconstruction by convex feature selection when incorporating genetic perturbations. *PLoS Comput Biol* 6(12):e1001014
77. Gillis J, Pavlidis P (2011) The role of indirect connections in gene networks in predicting function. *Bioinformatics* 27(13):1860–1866
78. Miller M, Feng XJ, Li G, Rabitz H (2012) Identifying biological network structure, predicting network behavior, and classifying network state with high dimensional model representation (HDMR). *PLoS One* 7(6):e37664
79. Altay G (2012) Empirically determining the sample size for large-scale gene network inference algorithms. *IET Syst Biol* 6(2):35–43

80. Filosi M, Visintainer R, Riccadonna S, Jurman G, Furlanello C (2014) Stability indicators in network reconstruction. PLoS One 9(2):e89815
81. Jurman G, Visintainer R, Filosi M, Riccadonna S, Furlanello C (2015) The HIM glocal metric and kernel for network comparison and classification. In: Proceedings IEEE data science and advance analytics (DSAA 2015), vol 36678. IEEE, Piscataway, pp 1–10
82. Jurman G, Visintainer R, Riccadonna S, Filosi M, Furlanello C (2013) The HIM glocal metric and kernel for network comparison and classification. arXiv:1201.2931 [math.CO]
83. Jurman G, Visintainer R, Furlanello C (2011) An introduction to spectral distances in networks. Front Artif Intell Appl 226:227–234
84. Davison A, Hinkley D (1997) Bootstrap methods and their applications. Cambridge University Press, Cambridge
85. Schaffter T, Marbach D, Floreano D (2011) GeneNetWeaver: in silico benchmark generation and performance profiling of network inference methods. Bioinformatics 27(16):2263–2270
86. Opgen-Rhein R, Strimmer K (2007) From correlation to causation networks: a simple approximate learning algorithm and its application to high-dimensional plant gene expression data. BMC Syst Biol 1(1):37
87. Schäfer J, Strimmer K (2005) An empirical Bayes approach to inferring large-scale gene association networks. Bioinformatics 21:754–764
88. Franzin A, Sambo F, Di Camillo B (2017) bnstruct: an R package for Bayesian Network structure learning in the presence of missing data. Bioinformatics 33(8):1250–1252
89. Basso K, Margolin AA, Stolovitzky G, Klein U, Dalla-Favera R, Califano A (2005) Reverse engineering of regulatory networks in human B cells. Nature Genet 37:382–390
90. Phan RT, Saito M, Basso K, Niu H, Dalla Favera R (2005) BCL6 interacts with the transcription factor Miz-1 to suppress the cyclin-dependent kinase inhibitor p21 and cell cycle arrest in germinal center B cells. Nat Immunol 6(10):1054–1060
91. Basso K, Saito M, Sumazin P, Margolin AA, Wang K, Lim WK, Kitagawa Y, Schneider C, Alvarez MJ, Califano A, Dalla Favera R (2010) Integrated biochemical and computational approach identifies BCL6 direct target genes controlling multiple pathways in normal germinal center B cells. Blood 115(5):975–984
92. Mani KM, Lefebvre C, Wang K, Lim WK, Basso K, Dalla Favera R, Califano A (2008) A systems biology approach to prediction of oncogenes and molecular perturbation targets in B-cell lymphomas. Mol Syst Biol 4(1):169
93. Ma HW, Kumar B, Ditges U, Gunzer F, Buer J, Zeng A (2004) An extended transcriptional regulatory network of *Escherichia coli* and analysis of its hierarchical structure and network motifs. Nucleic Acids Res 32:6643–6649
94. Van den Bulcke T, Van Leemput K, Naudts B, van Remortel P, Ma H, Verschoren A, De Moor B, Marchal K (2006) SynTReN: a generator of synthetic gene expression data for design and analysis of structure learning algorithms. BMC Bioinf 7(1):43
95. de Matos Simoes R, Emmert-Streib F (2012) Bagging statistical network inference from large-scale gene expression data. PLoS One 7(3):e33624
96. Ravasz E, Somera A, Mongru D, Oltvai Z, Barabasi AL (2002) Hierarchical organization of modularity in metabolic networks. Science 297(5586):1551–1555
97. Visintainer R (2012) Distances and stability in biological network theory. PhD thesis, DISI, University of Trento
98. Lance GN, Williams WT (1966) Computer programs for hierarchical polythetic classification (“similarity analysis”). Comput J 9:60–64
99. Lance GN, Williams WT (1967) Mixed-data classificatory programs, I. Agglomerative systems. Aust. Comput J 1:15–20
100. Jurman G, Riccadonna S, Visintainer R, Furlanello C (2012) Algebraic comparison of partial lists in bioinformatics. PLoS One 7(5):e36540
101. Cox TF, Cox MAA (2001) Multidimensional scaling. Chapman and Hall, Boca Raton
102. Hamming R (1950) Error detecting and error correcting codes. Bell Syst Tech J 29(2):147–160
103. Tun K, Dhar P, Palumbo M, Giuliani A (2006) Metabolic pathways variability and sequence/networks comparisons. BMC Bioinf 7(1):24
104. Morris M, Handcock M, Hunter D (2008) Specification of exponential-family random

- graph models: terms and computational aspects. *J Stat Softw* 24(4):1–24
105. Dougherty E (2010) Validation of gene regulatory networks: scientific and inferential. *Brief Bioinf* 12(3):245–252
 106. Iwayama K, Hirata Y, Takahashi K, Watanabe K, Aihara K, Suzuki H (2012) Characterizing global evolutions of complex systems via intermediate network representations. *Nat Sci Rep* 2:srep00423
 107. Ipsen M, Mikhailov A (2002) Evolutionary reconstruction of networks. *Phys Rev E* 66(4):046,109
 108. Furlanello T, Cristoforetti M, Furlanello C, Jurman G (2013) Sparse predictive structure of deconvolved functional brain networks. arXiv:1310.6547
 109. Mina M, Boldrini R, Citti A, Romania P, D'Alicandro V, De Ioris M, Castellano A, Furlanello C, Locatelli F, Fruci D (2015) Tumor-infiltrating T lymphocytes improve clinical outcome of therapy-resistant neuroblastoma. *OncoImmunology* 4(9):e1019981
 110. Fay D, Moore A, Brown K, Filosi M, Jurman G (2015) Graph metrics as summary statistics for Approximate Bayesian Computation with application to network model parameter estimation. *J Complex Netw* 3:52–83
 111. Gobbi A, Jurman G (2015) A null model for Pearson correlation networks. *PLoS One* 10(6):e0128115
 112. Masecchia S (2013) Statistical learning methods for high dimensional genomic data. PhD thesis, DIBRIS, University of Genoa
 113. Csermely P, Korcsmáros T, Kiss H, London G, Nussinov R (2013) Structure and dynamics of biological networks: a novel paradigm of drug discovery. A comprehensive review. *Pharmacol Therap* 138:333–408
 114. Donnat C, Holmes S (2018) Tracking network dynamics: a review of distances and similarity metrics. arXiv:1801.07351
 115. Read RC, Wilson RJ (2005) An atlas of graphs. Clarendon Press, Gloucestershire
 116. Liu YY, Slotine JJ, Barabási AL (2011) Controllability of complex networks. *Nature* 473(7346):167–173
 117. Shen-Orr S, Milo R, Mangan S, Alon U (2002) Network motifs in the transcriptional regulation network of *Escherichia coli*. *Nat Genet* 31:64–68
 118. Marbach D, Schaffter T, Mattiussi C, Floreano D (2009) Generating realistic in silico gene networks for performance assessment of reverse engineering methods. *J ComputBiol* 16(2):229–239
 119. Jurman G, Filosi M, Riccadonna S, Visintainer R, Furlanello C (2016) Differential network analysis and graph classification: a glocal approach. In: Rogato A, Zazzu V, Guerracino M (eds) Dynamics of mathematical methods in biology – bringing Math to Life. Springer, New York, p 268



Chapter 15

Gene Regulatory Networks: A Primer in Biological Processes and Statistical Modelling

Olivia Angelin-Bonnet, Patrick J. Biggs, and Matthieu Vignes

Abstract

Modelling gene regulatory networks requires not only a thorough understanding of the biological system depicted, but also the ability to accurately represent this system from a mathematical perspective. Throughout this chapter, we aim to familiarize the reader with the biological processes and molecular factors at play in the process of gene expression regulation. We first describe the different interactions controlling each step of the expression process, from transcription to mRNA and protein decay. In the second section, we provide statistical tools to accurately represent this biological complexity in the form of mathematical models. Among other considerations, we discuss the topological properties of biological networks, the application of deterministic and stochastic frameworks, and the quantitative modelling of regulation. We particularly focus on the use of such models for the simulation of expression data that can serve as a benchmark for the testing of network inference algorithms.

Key words Gene expression regulation, Regulatory network modelling, Systems biology data simulation, Post-transcriptional regulation, Post-translational regulation, Deterministic and stochastic models, Molecular regulatory interactions

1 Introduction

The different regulatory processes occurring within cells are often depicted as a network of interacting entities. These entities can be mapped onto different layers that represent the different biological molecules involved in expression regulation, for example, transcripts and proteins (Fig. 1a). High-throughput studies provide us with a measurement of the variable levels of a given layer. For example, microarrays or RNA sequencing technologies measure mRNA abundance, and are commonly referred to as gene expression data. We refer the interested reader to [1] for such modern data handling practices, to [2] for associated statistical designs, and to [3] for a data processing and primary analysis workflow.

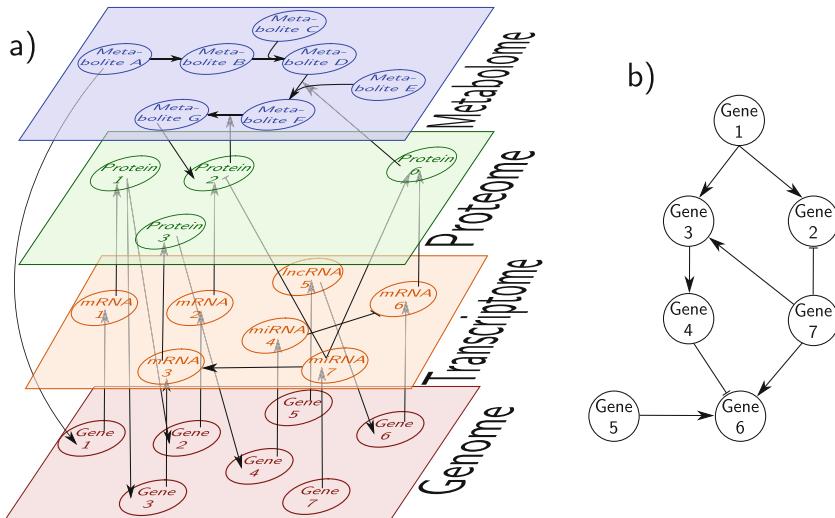


Fig. 1 Biological versus statistical representation of a GRN. **(a)** Biological regulatory systems are complex: the different intermediary products of genes—transcripts and proteins—as well as metabolites interact in a multi-layer network. Such networks are the best representation we can give of complex biological systems. **(b)** Statistical perspective: genes can be considered as nodes in a directed graph, where the edges represent regulatory interactions. Each parent variable node directly influences its children variables, therefore representing the regulation mechanism of a gene product on the transcription of another gene

From a biological perspective, entities from different layers are found to interact. Indeed, in addition to the well-known control of transcription by proteins termed transcription factors (TFs), other steps of the gene expression process are targeted by regulatory molecules beyond proteins, e.g., small molecules such as metabolites and noncoding RNAs. On top of this dynamic regulation, the information encoded in the DNA itself exerts to some extent control over the expression profile of genes. Here the term “gene” refers to a DNA sequence coding for a protein or other untranslated RNA. However, it is usually impossible to measure in the same experiment data about all these molecular layers. We are therefore most of the time bound to making the most of one given data type from which we seek to extract patterns giving insight into the regulatory interactions at play. Thus gene regulatory networks (GRNs) successfully gather the detected relationships between transcripts, even if these relationships are mediated by other molecules such as proteins. GRNs represent these interactions in a graph where nodes correspond to genes (and gene products) and edges represent the regulatory relationships among them (Fig. 1b).

The modelling of such regulatory systems is an important aspect of the reverse engineering problem. Accounting for existing biological interactions can be key to a more accurate analysis of experimental data, e.g., in the analysis of differential gene

expression [4]. In addition, such models can be used to simulate expression data in order to assess the performances of a given network inference method, just like data can be simulated to assess gene expression differential analysis method performance [5]. Indeed, a detailed analysis of the strengths and weaknesses of a given method can guide the choice of a practitioner to choose among the possible different reverse engineering approaches and pave the way for needed method development. A possibility is to use as a benchmark a previously studied experimental dataset, but this approach is limited by our incomplete knowledge about true—if this truth is ever an achievable objective—underlying pathways. On the contrary, the use of simulated expression data from *in silico* networks renders possible the objective comparison of the results of network inference to the true underlying interaction graph. More precisely, synthetic data allows the assessment of the impact of sample size, noise, or topological properties of the underlying network on the methods performance. To make valid conclusions, one expects synthetic data to have features as close as possible to real data. Modelling such complex systems seems like an insurmountable task. However, by carefully designing each constituting element of the model it is possible to link the statistical representation of a regulatory system to the underlying biological mechanisms in a meaningful way. This is the very topic of this work.

This chapter aims at bringing together the biological and statistical representation of GRNs. In Subheading 2, we provide an overview of the different regulatory mechanisms that shape the gene expression profiles. We focus on the different regulatory molecules that target each step of the expression process. In Subheading 3, we introduce the reader to the basic concepts necessary to the construction of a GRN model, from the topological properties shaping biological networks to the mathematical frameworks used for the dynamic simulation of expression data and the representation of regulation from a quantitative point of view. Together, this chapter provides a first guide to GRN modelling anchored in the biological reality of gene expression regulation.

2 Biological Processes: From Gene to Protein

Proteins are the main actors in living organisms. They achieve a myriad of functions. Yet their structure, their production mechanisms, and their regulation to allow the cell or organism to adequately adapt to the environment are dictated by the information contained in the genetic material of the organism. The expression of a gene, a “coding sequence” into an active protein is a complex process involving numerous biological molecules transformed via varied reactions and interactions. The information encoded in the coding sequence of the DNA is transcribed into a messenger RNA (mRNA), which is processed and translated

into a protein, according to the central dogma of biology. Once synthesized, a protein may require additional “post-translational” modifications to acquire a functional form. In this section, we aim at providing an overview of the different regulatory interactions targeting each of these steps. This knowledge certainly helps data analysts designing more ad hoc models to extract knowledge from modern high-throughput measurements. While it is out of the scope of this chapter to provide a detailed and comprehensive description of the specific biological mechanisms, we provide references to more biology-centered reviews of the subject. An overview of the different molecular actors of this regulation can be found in Fig. 2.

2.1 Regulation of Transcription

The regulation of transcription is believed to be a key determinant of gene expression profile [6, 7]. It mainly leverages the action of TFs which act as activators or repressors for the transcription of target genes. Regulators act by binding to proximal or distal sites on the promoter of the target genes. They impact transcription by facilitating or restraining the recruitment of the transcriptional machinery to the target gene via protein–protein interactions with its constituents. While TF binding only involves proximal promoters in bacteria, additional remote regulatory elements such as enhancers, insulators, or locus control regions play an important role in the regulation of eukaryotic genes [7, 8]. A given TF can affect the expression of one or more target genes, and its impact on gene expression (i.e., activation, repression, or modulation) can change in response to a specific environmental or molecular stimulus. Typically, a TF will only regulate a few targets, but some global TFs can control transcription of large sets of genes [9]. Interestingly, while TFs play a crucial role in the control of gene expression, they are often found in low concentration, possibly only a few molecules per cell [7].

Conversely, the transcription of a specific gene can be controlled by several TFs. This important feature, termed “combinatorial regulation,” provides the cell with an increased complexity in transcriptional regulation. Each gene can potentially process several inputs which dictate its resulting expression profile [9]. The different regulator molecules can act independently, if each of them affects a different aspect of the transcriptional machinery. Alternatively, TFs are often found to form complexes, either homodimers or hetero-dimers, thereby exerting cooperative regulation on the target [9, 10]. Importantly, such cooperation implies that the regulation only occurs when all the components of the regulatory complex are present. Yet another mechanism of combinatorial regulation is a synergistic interaction, where the global effect of the different TFs is greater than the sum of their individual effects [8, 11]. Finally, different TFs can compete for the same binding site on the target promoter. The respective affinity of the

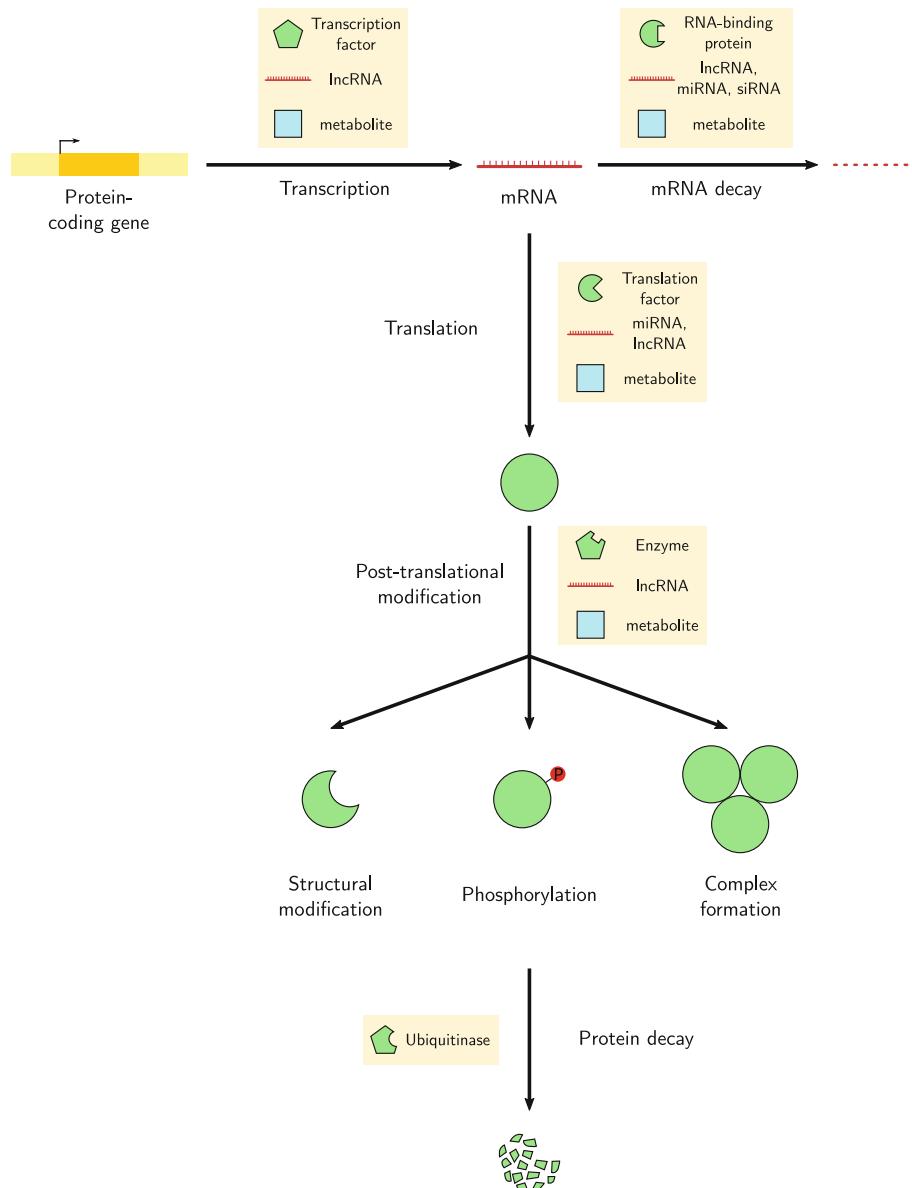


Fig. 2 The different steps of the expression process of a protein-coding gene, and its possible regulatory molecules. The colors represent the different molecule types: yellow: DNA, red: RNA, green: protein, blue: metabolite. A gene is first transcribed into an mRNA, with the possible involvement of transcription factors, long noncoding RNAs (lncRNAs) or metabolites. The mRNA is then processed and translated into a protein; again this process can be affected by translation factors, microRNAs (miRNAs), lncRNAs, or small molecules. The degradation of transcripts is influenced by noncoding RNAs, RNA-binding proteins or metabolites. Once synthesized, a protein can undergo post-translational modifications, mediated by other proteins, lncRNAs, or other small metabolites. Possible modifications include conformational change, modification of specific residues such as phosphorylation, or the formation of protein complexes. Proteins are tagged to degradation by specific enzymes, termed ubiquitinases

different molecules for the binding sequence determines which of them preferentially occupies the promoter. These affinities can be altered by environmental cues or changes in the promoter context (occupancy of neighboring sites, etc.).

In addition to protein regulators, transcription can be controlled by noncoding RNAs, whose role will be discussed later in this chapter. Furthermore, recent evidence tends to suggest that small RNAs and in particular microRNAs also play a role in transcription silencing, in addition to their impact on post-transcriptional functions detailed in the following sections [12, 13]. Lastly, other features can affect the transcription of genes. Specifically, the methylation state of DNA, and in particular of gene promoters, has been linked to gene silencing [14, 15]. A widespread example is the inactivation of tumor-suppressor genes by hypermethylation as a hallmark of cancer [16]. DNA methylation is controlled by an array of specialized enzymes. In eukaryotic cells, the chromatin structure, that is the packaging of the DNA, affects all steps of the transcription process. This structure is dynamic and is regulated by ATP-dependent chromatin-remodelling complexes which control DNA–histones interactions, and by histone-modification factors [17]. Histone post-translational modifications (such as methylation, acetylation, phosphorylation, etc.) greatly affect the chromatin structure, notably through the recruitment of chromatin-remodelling complexes, or by directly influencing their interactions with DNA. These histone marks have been found to correlate with transcription efficiency and in some case control the access of TFs to promoters [17].

2.2 Regulation of Translation

Following gene transcription and transcript processing, mRNAs are translated into proteins by ribosomes and associated molecules. This process is also targeted for regulation, both global and specific. The initiation of translation, that is binding of the translational apparatus to mRNAs and recognition of the translation starting site, is thought to be the step where most of the regulation occurs. As the specific mechanisms of translation initiation differ between bacteria and eukaryotes [7, 18], regulation processes are specific to each, but similarities can be observed. Regulation of translation offers a faster modulation of the concentration of proteins compared to transcription regulation, as the former silences already existing mRNAs, while with the latter these mRNAs are still transcribed until their decay [19].

As an example, during the response to a particular stress such as nutrient deprivation or temperature shock, cells often undergo a global decrease of their translational activity [19–21]. This global programming switch occurs through the control of the availability or the activity of the translational apparatus, notably through the phosphorylation state of eukaryotic initiation factors in eukaryotes. Such massive translation reduction allows a decrease of the energy

demand and a reallocation of cellular resources to stress response. Specific mRNAs encoding stress-response proteins can escape this regulation via distinct mechanisms.

Alternatively to global programming, translation of mRNAs can be specifically regulated for a small set of genes via the involvement of RNA-binding proteins (RBPs) or microRNAs (miRNAs) [20, 22]. These regulatory molecules recognize and bind to specific sequences in the target transcript, mainly situated in the untranslated regions of the mRNA. RBPs mainly act through interactions with the translational apparatus, leading to the inhibition of translation [20]. miRNAs act through the RNA-induced silencing complex (RISC) [23, 24]. The level of complementarity between a miRNA and its binding sequence on the target transcript specifies the triggered mechanism of regulation [7]: the extensive base-pairing between the miRNA and its target triggers the degradation of the latter, while partial base-pairing induces translation inhibition [25]. Interestingly, it has been shown that in the case of miRNA-mediated translational repression, the promoter of the target gene determines the precise mechanism of action of the miRNA [26]. However the role of small RNAs are still not perfectly clear and additional processes could be discovered by further experimental studies [27].

It is interesting to note that the direct impact of small RNAs on the translation of a gene can also indirectly affect other processes such as transcription of nontarget genes. For example, [28] used miRNAs intervention experiments to detect their direct impact on TFs levels, but also reported the indirect effect of these miRNAs on the expression of these TFs' targets. The conservation of miRNA–mRNA sequence match, particularly in the 3' untranslated regions of genes, enables the identification of the miRNA-target potential pairings [29]. Conversely, evidence suggests that miRNA synthesis can also be controlled by other RNAs [30]. We can here again raise the concept of regulation network to start organizing this knowledge. Prior interactions can be predicted to create such networks [28, 31]. Algorithms for RNA–RNA interaction predictions (e.g., [32]) are compared in [33]. Then molecular techniques can confirm the putative relationships [34].

The primary sequence of the transcripts also heavily influences their translation [18]. In particular, the formation of secondary structures within the transcript (e.g., hairpin, stem-loop, etc., which can be facilitated by the properties of the primary sequence such as GC content) and specifically in regions involved in translation initiation can impair the translation process. Specific structural features such as upstream open reading frames or internal ribosome entry sites can also impact translation. The detailed features of such mechanisms are beyond the scope of this chapter, and we refer the reader to [18]. However, being aware of the existence of these mechanisms can allow the modeller to include them or at least

discuss their effect on the outcome of an analysis. In this vein, [35] postulate that protein–protein interactions are linked to the regulation of the corresponding genes by miRNAs.

Lastly, an interesting mechanism of translation control is the regulation via “riboswitches” [36–38]. A riboswitch is a regulatory sequence within mRNAs which responds to specific cues, namely temperature or the presence of particular metabolites. Thermosensors are a class of riboswitches that respond to temperature by changing their conformation, therefore modifying the translation rate of the transcript. Alternatively, riboswitches can detect and link to specific metabolites. This provokes a modulated translational activity of the transcript via the modification of the mRNA conformation.

2.3 Regulation of mRNA Decay

In addition to the elimination of defective mRNAs arising, for example, from transcription or splicing errors, fully functional mRNAs are subject to spontaneous or targeted degradation. Regulation of mRNA decay plays an important role in the resulting transcript level. Specific degradation of transcripts can be mediated by RBPs, or by small RNAs, namely miRNAs and small interfering RNAs (siRNAs) [21]. Interestingly, mRNAs encoding functionally related proteins were shown to exhibit correlated half-lives. This phenomenon suggests a common regulation of mRNAs involved in similar biological processes [39, 40].

RBPs recognize and bind to specific sequences in mRNAs. It allows them to trigger the recruitment of decay factors, ultimately leading to target degradation. Alternatively, some RBPs have been found to stabilize their targets, protecting them from degradation [41]. Just as factors regulating other aspects of gene expression, RBPs can interact to exert combinatorial control over mRNA decay rates.

Alternatively, miRNAs and siRNAs can promote the decay of target mRNAs, via interactions with RISC and possibly with other RBPs. Such a phenomenon is coined RNA interference or RNAi [24, 42]. As mentioned previously, such degradation is promoted by the perfect pairing of the small RNAs with the target transcript. Several mechanisms can be involved to trigger target degradation. Possibly, interactions with small RNAs and RISCs promote the endonucleolytic cleavage of the transcript. Another explanatory mechanism is that the target can be directed to P-bodies, which are small cytoplasmic granules containing RNA degradation machinery [24]. Targeted mRNAs are locked in these P-bodies and consequently degraded before they can be further processed, e.g., translated.

2.4 Regulation of Protein Activity

After translation, proteins are sometimes subjected to additional modifications to acquire their fully functional state. These changes can be irreversible, i.e., proteolytic cleavage of the peptidic precursor to obtain a functional protein [7, 43]. Alternatively, the

cell can modulate the activity of its protein pool via a number of reversible post-translational modifications. A common mechanism is the modification by specialized enzymes of some amino acids on the protein, such as phosphorylation, oxidation, or acetylation [44]. In particular, phosphorylation is a common mechanism for the activation of enzymes, TFs, or other proteins. It is used in signalling pathways to relay extracellular messages to the nucleus, via a cascade of phosphorylation which activate kinase proteins [45, 46]. The endpoints of such pathways are generally TFs, whose phosphorylation lead to their activation and relocalization into the nucleus where they can modulate the expression of appropriate response genes.

Taking again the example of signal transduction in the cell, the cascade of phosphorylation is initiated by the activation of membrane receptors that detect a particular signal in the environment—generally a vitamin, a hormone, or another metabolite. This specific ligand binds to the receptor peptide, and triggers conformational changes to lead to the activation of the receptor. Such activation prompted by the binding of a small molecule is also frequently found in metabolic pathways, as a mean to regulate the production of a specific compound [43]. Metabolites can bind to the enzymes responsible for their synthesis in a feedback loop that auto-regulates enzyme activity according to the abundance of this specific product. Conformational changes resulting from ligand binding can mask or reveal the catalytic site of the enzyme, thereby controlling its ability to bind with its substrates.

Lastly, peptidic chains sometimes need to assemble into multimers, to form a functionally active molecular complex [43]. Such protein complexes can be composed of several copies of the same protein or of different proteins. In the latter case, the abundance of the complex, and hence its activity, is limited by the least abundant species. It is an interesting mechanism of regulation of the complex activity. Information about interactions among subunits can be found in protein–protein interaction databases (*see, for example, [47]*).

2.5 Regulation of Protein Decay

Cells possess several pathways for the degradation of proteins. A first mechanism is concerned with the degradation within lysosomes, which is a non-specific process, notably solicited in response to nutrient starvation as a rapid source of amino acids [48]. In addition, proteins can also be specifically tagged to degradation, via conjugation of a ubiquitin chain to the target peptide [7, 49, 50]. Tagged proteins are recognized by cellular machineries termed 26S proteasomes and subsequently degraded. This ubiquitin-proteasome pathway provides the cell with a way to rapidly control a regulatory process by degrading its effectors. It is notably involved in the regulation of transcription via degradation of specific TFs [50].

The addition of ubiquitin on target proteins is mediated by the E1, E2, and E3 enzymes. The different isoforms of the E2 and especially E3 family confer a great specificity to this process, as each isoform can recognize different substrates. Additionally, some structural properties of proteins can impact their affinity as substrate for the ubiquitin-proteasome pathway. For example, a member of the E3 family recognizes particular amino acids at the N-terminal position of proteins, in what is called the N-end rule pathway [50]. The nature or accessibility of specific residues can also impact the ability of ubiquitination enzymes to recognize and tag target proteins.

It is interesting to note that the ubiquitin-proteasome pathway is able to degrade only a subunit of a given protein, for example, to produce a functionally active product or on the contrary inactivate the protein. This is the case for the NF- κ B TF, which is bound by its inhibitor, I κ B [49]. In response to a specific signal, the complex is ubiquitinated, and the proteasome cleaves the I κ B, thereby freeing the TF, which, in turn, is relocated into the nucleus to trigger the required cellular response.

Quantitative measurements have highlighted the coupling between synthesis and decay rates of proteins. As for transcripts, these parameters seem to be correlated among proteins intervening in common complexes or functions. It appears that proteins involved in housekeeping functions are relatively stable, with a high production rate, leading to high concentrations in cells. On the contrary, regulatory proteins tend to be less synthesized and more rapidly degraded. This is consistent with the observation that they are often found in a low concentration in the cell [51, 52].

2.6 The Role of Genetic Variation

In addition to diverse cellular molecules which perform a wide range of regulatory activities, the DNA sequence itself plays a role in regulating gene expression. Regulatory sequences present in the promoter region of genes or in the transcribed or translated sequences dictate the set of molecules and complexes that control the expression of these genes. These sequences target transcripts or corresponding proteins for particular regulatory mechanisms. Their affinity for regulators control the strength of this regulation. The impact of genetic variation on gene expression has been studied, notably via expression quantitative trait loci (eQTL) studies. eQTLs are genomic regions within which genetic variability is associated with variation in the abundance of a particular transcript [53]. More generally genetical genomics studies (also termed cellular genomics) [54, 55] analyze how polymorphisms lead to variation in molecular traits, such as mRNA, protein, or metabolite profiles.

Using additional genomics data such as DNA methylation state or chromatin accessibility, researchers are now focusing on identifying the specific mechanisms which relate genetic variants

to response molecular traits. At the transcript level, evidence tends to show that eQTLs lead to transcript abundance variability mainly via their impact on TF binding [55–57]. Polymorphisms at these loci also affect other aspects of transcription, but it is yet to be determined if it is a direct consequence of genetic variation or merely an indirect effect of variation in TF binding efficiency [6]. Some polymorphisms have also been shown to affect mRNA degradation, notably through modification of miRNA-binding sites, or other post-translational mechanisms [6, 55]. In a groundbreaking effort, [58] discovered instructions encoded in the sequence itself to regulate gene activity. The nucleotide composition can be directly read to accurately decipher biological mechanisms.

2.7 An Example: Long Noncoding RNAs

After this review of the possible interactions regulating the different aspects of the gene expression process, we now turn our attention to a specific class of regulators whose role in the different biological processes mentioned earlier is just starting to be appreciated. Indeed, the functional importance of long noncoding RNAs (lncRNAs) was only hinted at when experimental studies of genome-wide transcription in cells revealed that a large fraction of the genome is transcribed, even if only a small amount actually encodes proteins [59, 60]. This discovery shook the traditional central dogma of biology stating that RNAs' primary role is to serve as messengers to produce functionally active proteins. On the contrary, as highlighted above, noncoding RNAs are now known to play important regulatory roles. While we now have a fair understanding of small noncoding RNAs (e.g., miRNAs, siRNAs, etc.) and the associated biological processes, lncRNAs (exceeding 200 base-pairs, as an arbitrary defining threshold) remain for most of them *terra incognita*. In particular, the extent of their functional role is yet to be determined, and there is still debate about whether the RNA molecule itself has a functional role or if only the physical changes triggered by its transcription (e.g., chromatin opening, helix unwinding, etc.) impact the transcription of neighbor genes while the produced transcript is useless [7, 61]. This is notably due to the fact that their primary sequence is less conserved than those of protein-coding genes [60, 62]. Nonetheless, experimental studies put us on the track of lncRNA involvement in a great variety of biological processes, from regulation of gene expression and chromatin state to genomic imprinting, in particular X chromosome silencing [59, 60, 63]. In addition, characteristic features of RNA make them well-suited for regulatory functions: their fast kinetics with no need for translation and their rapid degradation is particularly convenient for a fast and transient response to external stimuli. Moreover, their ability to bind DNA and RNA allows them to interact with both genes and transcripts [61, 64]. In this section, we briefly present the diverse roles played by lncRNAs in

the regulation of gene expression. For a more thorough review about the biological roles of lncRNAs, we refer the reader to the review by Geisler et al. [64].

One of the primary focuses of early studies about lncRNAs was their involvement in chromatin modelling [59], ultimately resulting in the modulation of gene expression. lncRNAs can act as scaffold which bring together different chromatin-remodelling proteins and to assemble them into a functional complex [59, 61]. Alternatively, they can guide such proteins to a target location, triggering changes in chromatin structure [62]. An interesting mechanism of action follows the transcription trail of the non-coding RNA which influences the chromatin state. It consequently impacts the transcription of neighboring genes [64].

lncRNAs can also enhance or repress the initiation of transcription via interactions with the basal transcriptional machinery. For example, as a response to heat shock, the interaction of a specific lncRNA with RNA polymerase II triggers the inhibition of target genes [62, 64]. Additionally, lncRNAs can target TFs and modulate their activity, by directly prompting conformational changes, by recruiting TFs onto the target promoter, or by withholding the TFs away from their targets [7].

lncRNAs are also involved in other steps of the gene expression process. In particular, they can influence mRNA processing, in particular mRNA splicing and editing [62, 64]. Some lncRNAs can impact translation efficiency of their target transcripts, through mechanisms which are still not totally clear [64]. They also putatively control RNA stability, either by recruiting specialized degradation machinery to the transcript, or by competing with miRNAs for binding sites. In the latter case, lncRNAs play a protecting role for mRNAs in preventing or delaying their degradation. For example, they can lure miRNAs to competitively bind to the same targets [61, 64]. Finally, lncRNAs can assist in protein binding to modulate their activity. Target proteins can be TFs, chromatin remodellers, or other regulatory molecules.

While a few well-studied lncRNAs provide evidence for a functional role of these transcripts, a lot remains unknown about them. In particular, it is important to keep in mind that the functional roles described above apply to a few number of characterized lncRNAs, and it is possible that a fraction of these transcribed noncoding genes are the result of transcriptional noise or experimental artifacts [63]. The modeller has the choice to include such information for a few annotated lncRNAs only, or to include the different putative roles, e.g., in a Bayesian framework.

As demonstrated throughout this section, the expression of genes is subject to a tight regulation from which arises great biological complexity. We now embrace the point of view of the statistical modelling of such biological systems. In particular, we

discuss the different aspects of the construction of a model that must be carefully thought out in order to faithfully describe the biological processes under study.

3 Modelling Gene Expression

Statistical models of GRNs aim at reproducing biological systems from a mathematical perspective to permit, *inter alia*, the simulation of their dynamical behavior. A number of models for the simulation of expression data have been proposed in the last decades; an overview of the principal algorithms and their key features is presented in Table 1. However, the design of such simulation tools is far from trivial. First and foremost, the model must be a faithful representation of the biological system, from the general topology of the underlying regulatory network to the quantitative regulation exerted on the genes and gene products. In addition, different mathematical frameworks can be used for the dynamic simulation of expression profiles, each of them carrying its own set of assumptions and limitations.

With all these considerations in mind, the next section provides a thorough reflection on the different features to examine when building a simulation network as well as a contrast of existing methods to simulate data from an *in silico* network. The general pipeline for the construction of a simulation algorithm can be found in Fig. 3.

3.1 Topological Properties of Regulatory Networks

A crucial step in simulating expression data from regulatory networks is the selection of a network topology that defines the interactions among the molecules. In a graphical representation of a GRN, nodes typically represent genes and their products, while edges correspond to regulatory interactions between molecules. Edges carry the direction of the regulation, that is, which nodes are regulators and which nodes are target molecules. The choice of the topology of a GRN is by no means an easy task. A first and simple approach for network modelling is to represent regulatory networks as random networks [75, 76] (also termed Erdős-Rényi graphs) in which each pair of nodes has the same probability of being connected. This model was and is still used for expression data simulation. However, topological analysis of pathways recovered from model organisms highlighted the existence of specific structural properties among biological networks, owing to the evolutionary constraints that shaped them. Algorithms for the generation of synthetic networks with similar properties were developed to construct more realistic models of biological systems. Interestingly, a number of these properties are shared with nonbiological systems such as the Internet or social networks [77]:

Table 1
Overview of existing methods of expression data simulation and their characteristics

Simulation method	Network topology	Mathematical formalism	Simulated molecules	Simulated reactions
Mendes et al. [65]	<ul style="list-style-type: none"> • Random • Small-world • Scale-free • Regular grid 	ODEs	mRNAs	<ul style="list-style-type: none"> • Transcription (regulated, Hill function) • mRNA decay (1st order process)
Van den Bulcke et al. [66]—SynTReN	Sampling from source network	ODEs (steady-state)	mRNAs	<ul style="list-style-type: none"> • Transcription (regulated, Hill and Michaelis–Menten functions) • mRNA decay (1st order process)
Ribeiro et al. [67]—SGNSim	User-defined	Time-delay stochastic model	Gene promoters, mRNAs, proteins, RNA polymerase, ribosomes	<ul style="list-style-type: none"> • Transcription (different transcription rate for each promoter state) • Translation (1st order process) • mRNA and protein decay (1st order processes)

Roy et al. [68]—RENCO	<ul style="list-style-type: none"> • Scale-free (protein–protein interaction) • Exponential degree distribution (transcription network) 	ODEs	mRNAs, proteins	<ul style="list-style-type: none"> • Transcription (regulated, Hill function) • Translation (1st order process) • mRNA and protein decay (1st order processes)
Di Camillo et al. [69]—NETSim	Hierarchical modular topology model	ODEs	mRNAs	<ul style="list-style-type: none"> • Transcription (regulated, fuzzy logic) • mRNA decay (1st order process)
Haynes et al. [70]—GRENDEL	Distinct in- and out-degree distribution	ODEs	mRNAs, proteins, environment	<ul style="list-style-type: none"> • Transcription (regulated, Hill function) • Translation (1st order process) • mRNA and protein decay (1st order processes)
Hache et al. [71]—GeNGe	<ul style="list-style-type: none"> • Random • Scale-free • Regulatory motifs • User-defined 	ODEs	mRNAs, proteins, RNA polymerase, ribosomes	<ul style="list-style-type: none"> • Transcription (regulated, Hill function) • Translation (1st order process) • mRNA and protein decay (1st order processes or Michaelis–Menten decays)

(continued)

Table 1
(continued)

Simulation method	Network topology	Mathematical formalism	Simulated molecules	Simulated reactions
Schaffter et al. [72]—GeneNetWeaver	Module extraction from source network	Chemical Langevin Equation	mRNAs, proteins	<ul style="list-style-type: none"> • Transcription (regulated, Hill function) • Translation (1st order process) • mRNA and protein decay (1st order processes)
Pinna et al. [73]—SysGenSIM	<ul style="list-style-type: none"> • Random • Scale-free • Random modular • Modular with exponential in-degree and power law out-degree 	ODEs	mRNAs, cis- and trans-eQTLs	<ul style="list-style-type: none"> • Transcription (regulated, Hill function) • mRNA decay (1st order process)
Tripathi et al. [74]—sgnesR	User-defined	Time-delay stochastic model	Gene promoters, mRNAs, proteins	<ul style="list-style-type: none"> • Transcription (different transcription rate for each promoter state) • Translation (1st order process) • mRNA and protein decay (1st order processes)

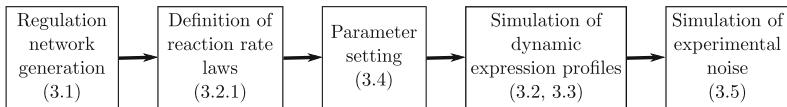


Fig. 3 The different steps of an algorithm for expression data simulation. Each of these steps will be detailed in the referred sections

- **Small-world property:** Networks are characterized as small-world if their average path length (*see Note 1*) between any two nodes is small. It has been shown that most biological networks exhibit such a property (*see*, for example, [78–80]). This implies that components of biological networks are easily reachable from any other node, which allows a rapid response to stimuli or perturbations [80]. Synthetic random small-world networks are also referred to as Watts–Strogatz networks [81].
- **Scale-free property:** When studying the in- and out-degree distribution of (directed) biological networks, i.e., the number of incoming and outgoing edges, respectively, it has been noted that this distribution can often be modelled with a power-law distribution [82, 83]. More specifically, the probability of a node to exhibit k edges is $P(k) \propto k^{-\lambda}$. An implication is that the majority of nodes interact only with a few partners, while a small number of nodes, called hubs, are highly connected. Metabolic pathways were shown to have this property [78, 79], and so were GRNs [84]. For both types of networks, the scale parameter λ usually ranges between 2 and 3 [77, 85]. However recent findings suggest that for some organisms the in-degree distribution of transcriptional networks is not scale-free, as detailed below. An algorithm for generating random scale-free networks has been proposed by Albert-Barabási [83]. Bollobás [86] presented a directed version of scale-free networks, where both the in- and out-degree distributions are power laws, with possibly different λ coefficients.
- **Exponential distribution of the in-degree distribution** (for transcriptional networks): alternatively to the scale-free property, studies [9, 87] suggested that the in-degree distribution of GRNs for some organisms is better fitted by an exponential distribution, i.e., $P(k) \propto \frac{1}{\lambda} e^{-\frac{k}{\lambda}}$. This implies that genes are regulated only by a few (generally up to three) TFs [77], a more plausible configuration in biological networks.
- **Modularity:** real networks have a tendency to form groups of highly interconnected nodes, referred to as modules. This modular organization is characterized by a high average clustering coefficient [79, 81]. The clustering coefficient C of a node is a measure of the degree of connectivity among the direct neighborhood of this gene. This property is important for biological systems, as it implies that biological networks

are organized into relatively independent modules that each perform a distinct biological function. While inside a module the components are tightly linked, modules are only weakly connected with each other. This last property ensures to some degree robustness to the network, as disruption in one module is less likely to severely impair the rest of the network [77]. Methods to identify modules within pathways [88] could clearly inform gene network inference and this information should not be ignored when exploitable.

- **Hierarchical organization:** contrary to random or scale-free networks for which the average clustering coefficient decreases with the number of nodes in the network, biological networks are characterized by a system-independent average clustering coefficient [85]. Moreover, the clustering coefficient of a node is a function of its degree [85], since: $C(k) \propto k^{-1}$. This last property is a characteristic of a hierarchical organization of the network. This important mathematical concept allows us to reconcile the scale-free property and modular nature of biological systems. Indeed, it stresses that nodes of low connectivity tend to be found in clusters, while hub nodes constitute the junction between modules. It is to note that hub nodes will less likely be connected to each other.
- **Over-representation of network motifs:** another important feature of biological networks is the abundance of small regulatory motifs [89, 90] that are recurring and non-random building blocks of the global topology [91]. They confer specific advantages to the system by encoding well-defined local dynamic behaviors in response to perturbations, for example, buffering intrinsic stochasticity or on the contrary amplifying an external signal to trigger a cellular response [92]. One well-known example is the negative feedback loop, in which the product of a gene regulates its own transcription [93]. This auto-regulation feature allows the control of the natural fluctuation in the concentration of the gene product, as its synthesis is directly coupled to its abundance [92]. Another famous example is the feed-forward loop, which can simultaneously process two different stimuli and whose output depends on the nature (activation or repression) of the regulatory interactions composing the motif [94]. A detailed quantitative analysis of such motifs and the advantages they provide to the system can be found in the book [95].

As pointed out by Przulj et al. [96], such studies are based on our current and incomplete knowledge of biological networks. Despite this limitation, algorithms for the generation of graphs mimicking these structural properties have been proposed, for a more accurate representation of biological networks. In addition to the three most commonly used Erdős–Rényi, Albert–Barabási, and Watts–Strogatz networks, Haynes et al. [70] implemented

a method for simulating topologies with scale-free out-degree distribution and any desired in-degree distribution. Di Camillo et al. [69] proposed a hierarchical modular topology model that generates networks displaying scale-free degree distribution, high clustering coefficient independent of the network size, and low average path length. However, to offer flexibility in the simulation, most simulators offer as an option for the user to choose among the different network topologies cited above. It becomes therefore possible to assess the impact of the underlying topological properties on the performances of a given network inference algorithm.

The main drawback of *in silico* networks is that none of the aforementioned network simulation methods are able to simultaneously reproduce all characteristic features of real networks [66]. Another approach for graph generation has hence been proposed. It relies on the use of real biological networks determined experimentally. They are used as seeds from which sub-networks are sampled. Van den Bulcke et al. [66] proposed two sampling approaches: the cluster addition method and the neighbor addition method. Building up on this idea, Marbach et al. [97] further refined the approach by forcing the preferential inclusion of modules in the sampled sub-networks. This module extraction method ensures a fair representation of network motifs in the generated topology, as observed in biological networks. Such an approach of sampling from real networks ensures a more faithful picture of biological pathways. Again, this is contrasted by [66], as the real network sampling strategy relies on our current knowledge of regulatory networks, which is still incomplete, and probably biased towards well-studied pathways. Lastly, Haynes et al. [70] pointed out that networks generated from the same source network may not be “statistically independent” as they may overlap and thus provide redundant information. It is particularly true when sampling a large number of sub-networks from a single source, as most of them will share common nodes and interactions.

3.2 Mathematical Frameworks and Regulation Functions

Once the network topology is set, the next step for data simulation is to decide on the mathematical framework to be used to compute the profiles of gene expression, which will impact the choice of regulation rules for the system. It is important to carefully consider the different options, as each formalism carries a number of underlying assumptions about the represented system. Moreover, different levels of precision about the system can be integrated. Choices depend on many factors to achieve a balance between the level of required details to make the simulations more realistic, and the computational efficiency desired. While it is not our goal to offer an extensive comparison of all the possible formalisms, we emphasize here the difference between the two mainstream formalisms in existing simulators of expression data: the continuous-and-deterministic and discrete-and-stochastic

frameworks. We present the basic concepts of these approaches, and highlight the different hypotheses about the biological system underlying each model. For a more detailed and mathematically centered review of these and other formalisms we refer the reader to [98] and [99].

3.2.1 The Continuous and Deterministic Approach

The continuous and deterministic approach is particularly suited to simulate data that resemble those resulting from a transcriptomics (or other 'omics) experiment. The output is a series of continuous variables, as opposed to cruder logical models that predict the activation state of each gene as a binary outcome. In such deterministic models, biological molecules are represented as time-dependent continuous variables. Typically $x_i(t)$ represents the concentration of entity (or species) i at time t . Variation in the concentration of a species over time is assumed to occur in a continuous and deterministic way. Such changes are modelled as differential equations, in the form of:

$$\frac{dx_i}{dt} = f_i(\mathbf{X}), \quad (1)$$

where \mathbf{X} refers to the state of the system (that is the concentration of all the species present in the system), and f_i represents the change in the concentration of species i as a function, often non-linear, of the global state of the system. More specifically, in the case of a chemical species and associated reactions, $f_i(\mathbf{X})$ can be written as:

$$f_i(\mathbf{X}) = v_i(\mathbf{X}) - d_i(\mathbf{X}), \quad (2)$$

where the vector $v_i(\mathbf{X})$ represents the synthesis rate for species i , while $d_i(\mathbf{X})$ models its decay rate (due to degradation, dilution, use as a reactant, etc.). Both rates are themselves expressed as a function of the system state. The ensemble of reactions occurring in the network provides a set of coupled differential equations that describe the evolution of the state of the system through time. Except for simple networks, with only a few molecules and/or linear interactions, an analytical solution is often intractable. It is however possible to integrate the model in order to compute a numerical solution. A plethora of differential equation system solvers are available in different programming languages (for example, the `deSolve` package for R, the `dsolve` function of the Symbolic Math toolbox for Matlab or Berkeley Madonna).

Regulatory molecules for species i appear on the right-hand side of Eqs. (1) and (2). They impact the abundance of their target species i via regulation of the different expression steps, as we discussed them in Subheading 2. The regulation of a particular reaction is hence modelled via a reaction rate law, which dictates how the rate of the regulated reaction (e.g., v_i

or d_i) evolves with the concentration of the different regulatory molecules. The vast majority of proposed simulation algorithms focus on the representation of transcription regulation, but similar consideration can be applied to any type of regulation (translation, degradation, etc.). Two important features must be considered in order to fully characterize a reaction rate law: (1) the quantitative relation between a regulator abundance and the resulting reaction rate, and (2) the combination scheme of the individual effects of different regulators on a common target. In the following we will discuss these two aspects, using the example of the regulation of gene transcription.

We first consider gene i whose transcription is regulated by a single molecule j . Several choices are possible with regard to the resulting effect of the regulator abundance on the transcription rate. A simple approach is to consider that the regulation effect increases linearly with the regulator concentration [100, 101] (Fig. 4 a), as follows:

$$f_i(x_j) = \beta \cdot x_j \quad (3)$$

where f_i represents the transcription rate law of gene i , which depends on the concentration of the regulatory molecule x_j . In addition to the fact that this representation ignores any saturation effect arising from the limited amount of cellular resources and the maximum possible number of simultaneous transcription events, such a linear relationship can produce concentration values out of the plausible range of abundance encountered in vivo, possibly leading to infinitely large populations, which is biologically

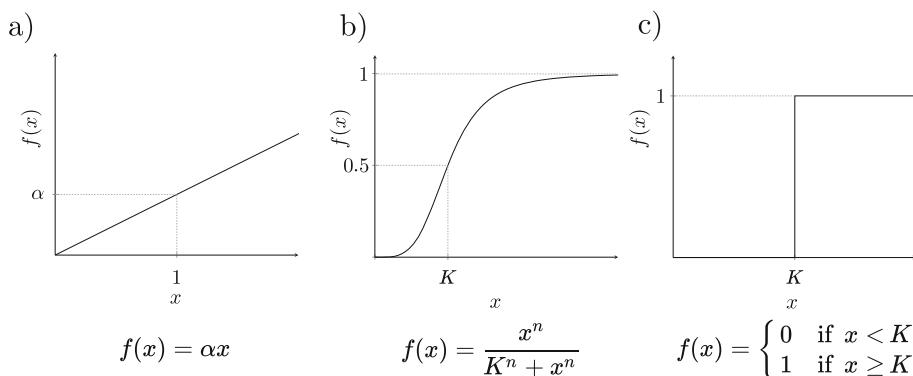


Fig. 4 Possible transcription rate law functions. **(a)** The rate law accounts for a linear dependence between the concentration of the regulator (x) and the transcription rate of the target gene ($f(x)$). **(b)** The Hill function accounts for the saturation of the regulation: when x is high, the variation of the transcription rate tends to 0. The parameter K corresponds to the concentration at which the regulatory molecules induce a transcription rate equal to half its maximum value. **(c)** With a step function, the target gene is only transcribed when the concentration of the regulator exceeds a certain threshold, here K

irrelevant. It is hence important to construct biologically credible reaction rate laws that result in realistic regulation strength and concentration values.

Alternatively, a Hill function (Fig. 4b) can be used to model the impact of an activator on the gene transcription:

$$f_i(x_j) = \frac{x_j^{n_{ij}}}{x_j^{n_{ij}} + K_{ij}^{n_{ij}}}, \quad (4)$$

or, for a repressor:

$$f_i(x_j) = \frac{K_{ij}^{n_{ij}}}{x_j^{n_{ij}} + K_{ij}^{n_{ij}}}, \quad (5)$$

where K_{ij} represents the concentration of regulator j required to obtain a half-maximum effect on the transcription rate of gene i , and n_{ij} controls the steepness of the regulation. It must be noted that K_{ij} must be nonnegative as it accounts for a concentration, and $n_{ij} \geq 1$. Indeed, when $n_{ij} = 0$ the resulting reaction rate law is constant. This sigmoid function accounts for the saturation of the regulatory effect: after the regulator concentration has reached a certain level, any further increase in this concentration will only result in a minimal change in the transcription rate. Additionally, tuning the parameter n_{ij} enables to represent a variety of regulation behaviors, from a quasi-linear (n_{ij} small) to a step-like (n_{ij} high) function. Furthermore the use of a Hill function law can be justified by a thermodynamic model of the binding of TFs on the target promoter [102, 103]. Considering that the mean transcription rate of a gene is proportional to the saturation of its promoter by TFs, the effect of the regulator can be further refined as:

$$f_i(x_j) = \alpha_0 \left[1 + (FC_{ij} - 1) \frac{x_j^{n_{ij}}}{x_j^{n_{ij}} + K_{ij}^{n_{ij}}} \right], \quad (6)$$

where α_0 represents the basal transcription rate of the target gene in the absence of the regulator, and FC_{ij} the maximum fold-change (see Note 2) of gene expression induced by the regulator. Details of this computation can be found in the Supporting Information of [104]. It is straightforward to deduce the transcription rate law for a gene whose expression is controlled by an inhibitor, as the resulting maximum fold-change is $FC_i = 0$:

$$f_i(x_j) = \alpha_0 \left[1 - \frac{x_j^{n_{ij}}}{x_j^{n_{ij}} + K_{ij}^{n_{ij}}} \right] \quad (7)$$

Such representation is massively used in simulator algorithms, although with some variations [65, 66, 68, 70, 72, 73, 105]. For example, the transcription rate law represented in Eq. (6) can be adapted to account for a gene that is not expressed in the absence of its activator.

Taking the approximation that $n_{ij} \rightarrow \infty$, it is possible to simplify this Hill function, and model the transcription rate law as an on–off switch, where the maximum effect of the regulator on the transcription rate occurs as soon as the regulator molecule level exceeds a certain threshold. Below this threshold, no regulation is observed. Such representation is described by a step function (Fig. 4c):

$$f_i(x_j) = \begin{cases} \alpha_0 & x_j < K_{ij}, \\ \alpha_1 & x_j \geq K_{ij}. \end{cases}$$

α_0 can be set to 0, to model the case of a gene that is not expressed in the absence of its regulator. This simplification provides the basis for piecewise differential equations [98]. It allows us to model a non-linear interaction even if the kinetics of the regulation are not known in detail.

Once the quantitative effect of a regulator has been chosen, one must consider the overall effect of several regulators targeting a common gene. Indeed, different combinatorial regulations can be modelled. A simple example is to assume that different regulators impact the expression of the target gene independently of each other. This approach has been used by Mendes et al. [65]. For an independent combinatorial effect model, the resulting regulation effect of all regulators is equal to the product (*see Note 3*) of the individual effects of each regulator on the transcription rate.

Alternatively, the different TFs can assemble into a complex that will bind to the target promoter to regulate transcription. In this case, the resulting regulation effect will be limited by the least abundant regulator species. An example can be found in [68], in which different TFs can assemble into cliques which in turn can form TF complexes regulating the target gene. The resulting transcription rate law is therefore equal to 0 as soon as the concentration of one of the TFs reaches 0, as it is then not possible to form a functional complex.

An interesting approach has been proposed by Di Camillo et al. [69]. It uses fuzzy logic to represent the possible combinatorial interactions between different regulators. The advantage of such an approach is that it combines the Boolean logic functions (AND, OR, NOT, etc.) well suited to describe combinatorial behavior with continuous regulation, as the output of fuzzy logic functions is a continuous value. Given a continuous input, that is the concentration of each regulator, the fuzzy logic applies a set of functions

such as min, max, or \sum (sum) to output the level of regulation commonly achieved by the different regulators. Di Camillo et al. hence represent “cooperation” (for which the regulation is only achieved in the presence of all the required regulators) as a min function applied to the set of regulator concentrations. Similarly, synergistic behavior, direct inhibition, or competition are modelled with fuzzy logic functions.

Deterministic models are traditionally used for the simulation of expression data [65, 66, 68–71, 104] (see Table 1). However, despite its broad use, the deterministic formalism presents several limitations, which relates to the underlying hypotheses about the biological system depicted. In particular, the assumption of continuous change in species concentration is only valid for a macroscopic description of biological systems [98], i.e., when the number of molecules in the cell is large enough so that species concentrations can be considered to vary continuously when a discrete number of molecules are actually added/withdrawn from the system. When the abundance of a species reaches low values (defined as a thousand or less by [106]), this assumption does not hold anymore, and it is more correct to represent this abundance by a discrete molecule count. Moreover, the deterministic assumption can be questioned, particularly for small systems, given the fluctuation in the timing of biochemical reaction events [98]. Indeed two identical genes with the same transcription rate will not produce exactly the same number of transcripts during the same time period, due to the apparent stochasticity of biological events. While this fluctuation can be averaged out for highly abundant species, it is more difficult to ignore for species with only a few molecules per cell. As numerous studies have underlined the importance of stochasticity in biochemical systems [107–110], it can be preferable to explicitly model stochasticity in the simulation.

3.2.2 The Discrete and Stochastic Approach

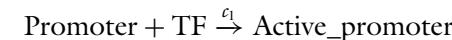
To overcome the limitation of continuous and deterministic models, in particular for modelling small systems, a discrete and stochastic representation of biological systems has been proposed. It must be noted that even if the continuous and deterministic approach and the discrete and stochastic framework are commonly referred to as respectively deterministic and stochastic models, there exist representation of biological systems that are either discrete and deterministic (e.g., Boolean networks) or continuous and stochastic (e.g., Chemical Langevin Equation, discussed later). One must hence keep in mind that continuous (resp. discrete) does not necessarily imply deterministic (resp. stochastic) as the first terms refer to the representation of species abundance while the latter corresponds to the variation of the system state.

In the discrete and stochastic framework, the state of the system corresponds to discrete values accounting for the number of molecules of each species present in the system. While the

vast majority of deterministic approaches are species-centered, i.e., one differential equation represents the evolution of one species abundance through time, stochastic models often rely directly on the biochemical reaction formalism. These reactions can be schematically represented in the form:



Or, in the context of gene expression:



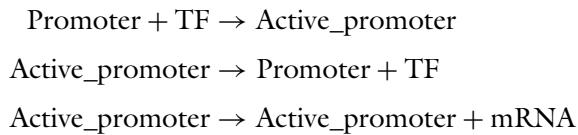
Each reaction is characterized by:

- A stoichiometry vector v_j which represents the change in abundance of the different species resulting from one firing (i.e., one occurrence) of the reaction. Negative and positive indices correspond respectively to reactants and products of the reaction.
- A propensity function $a_j(\mathbf{X})$, with $a_j(\mathbf{X})\tau$ representing the probability that the reaction will occur in the next time step $[t, t + \tau]$ given the system state at this time t . The rate a_j depends on the current state of the system. If c_j is the constant probability that one molecule of each of the r reactant species S_i , $1 \leq i \leq r$ collide and undergo the reaction in the next time unit, then $a_j = c_j \cdot \prod_r x_r$. Generally the number of reactants per reaction is limited to one or two, as a reaction involving more substrates can be decomposed into a set of elementary reactions [111]. Taking the example reactions above, the propensity function of the binding reaction of one TF molecule on the promoter will be: $c_1 \cdot x_{\text{Promoter}} x_{\text{TF}}$. It is therefore possible to link deterministic and stochastic rate constants, as shown in [111].

The system state change is then computed in terms of probability by the Chemical Master Equation (CME), which computes the evolution of the probability that the system is in state \mathbf{X} through time. For details about its computation, we refer the reader to the review by El Samad et al. [112]. An analytical solution of the CME provides the probability density function of the system state $\mathbf{X}(t)$. However, as for deterministic models, the computation of an analytical solution is impossible except for quite simple systems. Therefore, one way to study the behavior of the system is to construct numerical realizations of the CME. One of the most used method is Gillespie's Stochastic Simulation Algorithm

(SSA) [113]. SSA increments the system state at discrete time points, by randomly selecting the next reaction to fire, according to the propensity function of every possible reaction, as well as simulating the event (reaction occurring) time [112, 113]. Several exact (i.e., simulating every single reaction) alternatives to the original algorithm (the so-called direct method) have been proposed, such as the next-reaction method [114], the sorting direct method [115], and others (*see* [111, 116] for a review). However, exact algorithms are limited by their computational cost which renders the simulation of large systems intractable. Several approximation methods have been proposed, and have been thoroughly discussed [109, 112, 117, 118]. Approximation simulations such as the very popular tau-leaping method considerably reduce the simulation time, but at the expense of a hardly estimable loss of accuracy [109].

Stochastic discrete simulations offer a different perspective on the modelling of regulation compared to the deterministic continuous approach, as they explicitly model the binding of regulator molecules on to target promoter. Taking the example of TFs regulating the expression of a given gene, a stochastic model can represent the binding of regulatory molecules on the promoter of the target gene as follows:



In this model, a TF must be bound to the promoter for a transcription event to occur. Using this representation, it is easy to represent the different combinations of TFs bound to the promoter, and the transcription rate associated with each state. The possible combinatorial regulation effects can also be explicitly stated. For example, reactions can be added to encode the formation of a regulatory complex from the different TFs and to encode the binding of the complex to the target promoter.

An advantage of a stochastic model as compared to its deterministic counterpart is its ability to fit more precisely to the natural variation inherent to biological systems. This biological fluctuation can be crucial for understanding certain systems, as illustrated by El Samad et al. [112] that present several examples where a deterministic model fails to correctly predict the system behavior. As for its deterministic counterpart, the stochastic framework implies a number of hypotheses on the represented system. In particular, it relies on the essential assumption that the system is well-stirred, that is, the molecules are homogeneously spread in the volume. Moreover the simulation of temporal trajectories is computationally heavier as each reaction (in the case of exact

simulation algorithms) or group of reactions (for approximate methods) is simulated. This is especially true for a system with a high number of molecules or reactions with large value propensity functions, as both factors imply high firing rates.

3.2.3 Bridging the Gap

Starting from a stochastic model and in particular the CME representation of a given system, it is possible by means of simplifying assumptions to obtain the corresponding continuous deterministic model. As stated in the tau-leaping approximation of the SSA, under the assumption that the time step τ in the simulation is small enough so that the propensity functions of the different reactions stay approximately constant during the interval $[t, t + \tau]$, the number of reactions occurring during that time step can be modelled as a Poisson process [99, 112]. Consequently, it is possible to sample the number of occurrences of each reaction with propensity function a_j from a Poisson law with parameter $a_j \cdot \tau$. Moreover, if the time step τ is at the same time large enough so that each reaction fires more than once during the interval $[t, t + \tau]$ (usually feasible in systems where the concentration of species is large enough [112]), the system can be represented by a set of stochastic differential equations, termed the Chemical Langevin Equation (CLE) [112, 119]. In the CLE, the population of each species evolves in a continuous but stochastic manner, with the stochastic variation due to each reaction being proportional to the propensity of the reaction. As a consequence, the number of occurrences of a reaction with a high rate will have a higher variance than that of a reaction with a small reaction rate.

By further assuming that the abundance of each species is high enough, the stochasticity can be neglected, and the system is reduced to the Reaction Rate Equation (RRE) [99, 112] that is a set of differential equations:

$$\frac{d\mathbf{X}(t)}{dt} = \sum_{j=1}^M v_j \cdot a_j(\mathbf{X}(t)) \quad (8)$$

In this equation, the change per time unit in the concentration of a given species amounts to the sum over all reactions of the change in the species abundance triggered by one firing of the reaction (i.e., v_j), weighted by the rate of the reaction (i.e., the probability that the reaction will fire in one time unit, a_j). As highlighted by Higham [99], it is important to keep in mind that the solution of a deterministic model or RRE obtained by simplifying a stochastic model is not equivalent to an average of many numerical realizations of the corresponding stochastic model. It is rather a limit towards which these realizations tend when the different simplifying assumptions are fulfilled.

This connection between the stochastic and deterministic frameworks has been leveraged in the case of multi-scale systems, which are systems in which both slow and fast reactions and/or both low- and high-abundance species are present [112]. This situation can be encountered, for example, when simulating gene expression and metabolic reactions in a single model. On one hand, gene expression is a slow process involving genes that are present in only one or two copies per cell, and TFs whose abundance can be as low as a dozen of molecules only. On the other hand, metabolic reactions are fast enzyme-catalyzed processes and involve highly abundant metabolic species. The issue with such systems is that some but not all reactions could be represented by a deterministic process, while the rest requires a stochastic modelling. In such cases, the SSA performs poorly because of fast reactions and highly abundant species which monopolize most of the computational time. On the opposite, deterministic models provide a poor approximation of slow reactions and low abundant species. Several hybrid methods have hence been proposed. Their strategy is to split reactions and/or species in two sets of slow and fast reactions/species, and to use the most appropriate representation to model each set (*see* [116] for an overview of existing methods). This principle notably underlies the slow-scale SSA algorithm [120]. The interested reader is referred to [112, 116] for more details.

3.3 Model Simplifications

In addition to the mathematical framework they employ, existing simulators of expression data differ by a number of assumptions they make. Indeed, while it is crucial to accurately represent biological processes, our incomplete knowledge about the detailed mechanisms dictates the use of assumptions and simplifications in the models. These simplifications also arise from the desired level of complexity and the need for computational efficiency.

An important aspect to consider when designing a model is the type of molecules one wishes to represent. Early models were restricted to the simulation of the transcript levels only (*see* Table 1), and used the concentration of mRNAs as a proxy for the activity of their protein product. Such an assumption was justified by the inability to experimentally measure protein concentration [69]. However, as shown earlier in this chapter, a number of regulations occur post-transcriptionally. This certainly impacts protein abundance and/or activity without being reflected at the level of corresponding transcripts, except when the expression of a coding gene is linked to the activity of its corresponding protein via a feedback circuit. In particular, many studies revealed a generally weak not to say poor correlation between transcript and protein profiles [21, 52]. Such results suggest the need for more realistic models in which proteins are also included as the direct actors of transcription regulation. Some models already include the protein

level (*see* Table 1). The inclusion of other regulatory molecules, and in particular the noncoding yet very likely regulatory [121–123] fraction of the transcriptome could also be an interesting development. In addition, post-transcriptional regulations are traditionally overseen in expression simulation methods. Accounting for them would result in an increased complexity of the underlying mathematical models, but would pave the way for enhanced realism of simulated data.

Biological processes are not instantaneous. Time delays exists between, for example, transcription initiation and the release of a fully functional mRNA ready to be translated. Such delays have been mostly ignored, to the exception of SGNsim [67] (implemented in the R package `sgnesR` [74]). These stochastic models use a version of the SSA that is suited for the occurrence of delay in biochemical reactions. This can account for the time required for the transcription of a gene as well as the diffusion of molecules across cellular compartments. Additionally, spatial inhomogeneities can be considered. For example, one might want to include in the model the different cellular compartments, to account for the fact that in eukaryotic cells the synthesis of mRNA occurs in the nucleus, while their translation happens in the cytoplasm. This can be done in a deterministic framework by using partial differential equations [98].

3.4 Assigning Values for Model Parameters

When simulating in silico expression data, it is important to carefully choose the values of the different parameters in the model to obtain more plausible data. The initial abundance of each molecular species and the different reaction rates determine the resulting level of expression for each gene. It is crucial to use reasonable values in the range of those estimated from experimental datasets. The same attention must be paid to the kinetic parameters that define the strength and amplitude of regulation. This includes, for example, the Hill coefficients for a deterministic model, or the binding and unbinding rates of the different TFs on the promoter for a stochastic model. This choice is impeded by our limited knowledge about the precise kinetics of gene expression regulation [66]. However, a number of experimental results provide insights into the dynamics of the different molecular reactions, at least for some model organisms [51, 52]. The global distribution of the lifetime of transcripts and proteins, for example, is starting to be well-characterized across the different domains of life. The order of magnitude of transcription and translation rates are also available. In [124], Milo and Phillips gather relevant quantitative pieces of information about different biological processes, from cell component typical size estimates to the rates of transcription, translation, or metabolic reactions. The associated database, BioNumbers [125], allows to search the literature for quantitative properties of biological systems. This is

a valuable tool for modellers who seek realistic values for model parameters. Additionally, during the model construction it can be useful to conduct a sensitivity analysis to ensure that slight variations in parameter values do not produce completely different and/or surrealistic system behaviors.

Despite the increasing availability of quantitative knowledge regarding gene expression, to the best of our knowledge, none of the existing simulation tools presently offer a rigorous justification of the values used in their model. The parameters are usually sampled from large distributions to allow a wide variety of possible dynamical behavior (e.g., from quasi-linear to step-like regulatory functions) [66, 69]. Alternatively, parameter values can be required as input from the user, as it is the case in [67, 71, 74]. However the choice of values in the model often seems arbitrary [65, 68, 72, 73]. An interesting approach has been proposed by Haynes et al. [70]. In their model, the transcription, translation, transcript, and protein decay rates are sampled from values experimentally measured for real genes in *S. cerevisiae*. It should be noted however that this limits the validity of the simulations to this organism. Moreover, this approach is only possible for well-characterized model organisms for which abundant and reliable quantitative information is available.

In conclusion, it is important to anchor the mathematical RNA models in the biological reality not only via the represented molecules and interactions, but also through the quantitative information which is used to simulate the different reactions involved in gene expression.

3.5 Experimental Noise in In Silico Data

Even though the results of transcriptomics and other omics experiments provide an estimation of transcripts or other molecules level, they do not exactly reflect their precise *in vivo* abundance. Each step of the sample preparation process introduces to some extent bias in the quantitative estimation of the molecular concentrations. Such bias in turn impedes our ability to detect correlation between molecular profiles, or introduces spurious correlations, and needs to be accounted for when developing a reverse engineering approach. Consequently, when assessing the performance of such methods, it is important to test their robustness against increasing level of noise in the data.

Accordingly, several pipelines of data simulation include a step to add experimental noise in the resulting simulated expression profiles to mimic errors and bias introduced by the used measurement technology [65, 66, 70–73]. This step is particularly relevant for deterministic models, which produce data deprived of both biological and experimental noise. On the opposite, stochastic models already introduce some kind of variability in the dynamic profiles. The generation of *in silico* experimental noise is often based on models linking the measured intensity obtained with

a particular technology (e.g., microarray or RNASeq [126]) to the true underlying concentration [127–129]. Alternatively, a simple Gaussian noise can be added to the simulated data to introduce variation in order to blur existing correlations among molecular profiles [65, 71–73]. Mendes et al. proposed to use a Gamma distribution for experimental noise, as microarray data are often found to display a non-Gaussian noise and as the Gamma distribution is not centered around its mean.

4 Concluding Remarks

Biological systems are characterized by great complexity. From a systems perspective, regulatory networks are shaped according to specific properties that can be described mathematically. From a mechanistic perspective, gene expression is regulated at each step of the lifetime of the different gene products that are transcripts and proteins. Their synthesis, activity, and decay are tightly controlled by a vast array of factors ranging from proteins to noncoding transcripts and small molecules. Additionally, these processes are affected by an inherent stochasticity which induces variability in the molecular profiles. Statistical models provide a rich framework to represent this complexity, and it is now possible to generate *in silico* graphs resembling real networks, or to simulate biologically plausible noisy dynamic expression data. A statistical model must be carefully designed to accurately reflect the underlying processes, as, for example, determining the list of regulators of a molecule, quantifying the effect of regulatory factors, or assigning a value to the different reaction rates or other parameters.

In this chapter, we particularly focused on the use of GRN models for the simulation of expression data that can serve as benchmark for the testing of network inference algorithms. Indeed it is important that these simulations provide realistic data to allow researchers to draw meaningful conclusions about the performances of reverse engineering methods. However, beyond the problem of simulating expression data, all the identified regulatory relationships allow the modeller to account for a fine description of the underlying biological mechanisms, when such a level of detail is required.

We know that all models presume to a greater or lesser extent a simplification of the underlying biology. As we have shown in this chapter, most simulation models currently consider transcription regulation only, and exclude noncoding RNAs and possibly even proteins. While these simplifications can be justified by our insufficient knowledge about the processes at play or by computational limitations, it results in inadequate models as they overlook the complexity of gene regulation. We would however like to moderate this desire for increasingly detailed models that

could be more indicative of the underlying biological and technical influences in the data. Indeed, they allow a great flexibility in the inferred interactions, but this can become problematic and result in overfitting and in the detection of spurious regulations. The need for a trade-off calls for the use of additional data for the reverse engineering problem as well as advanced statistical tools accounting for the missing information.

Notes

1. The path length between a pair of nodes is defined as the length of the shortest path connecting the two nodes.
2. The fold-change of a gene is defined as the ratio of its transcription rate in the presence of a high concentration of regulatory molecules over its transcription rate in the absence of regulator. From equation (6) it is easy to see that the transcription rate of gene i tends towards $\alpha_0 \cdot FC_{ij}$ when x_j becomes large, hence the fold-change tends to $\frac{\alpha_0 \cdot FC_{ij}}{\alpha_0} = FC_{ij}$.
3. The use of the product, rather than the sum, ensures that if the concentration of a repressor is high enough to silence the gene (resulting in an individual effect close to 0) the overall transcription rate will also tend to 0 regardless of the quantity of activators present. It also implies that the overall fold-change obtained for large quantities of the different activators is the product of the fold-changes individually induced by each activator, which is justified thermodynamically in [130] and [103].

Acknowledgements

We are very grateful for enriching discussions and suggestions on this manuscript made by Samantha Baldwin and Susan Thomson (Plant and Food Research, Lincoln, New Zealand). We would like to thank the reviewers of this chapter for their useful comments. MV was partly supported by a visiting professor scholarship from Aix-Marseille University.

References

1. Conesa A, Madrigal P, Tarazona S, Gomez-Cabrer D, Cervera A, McPherson A, Szczesniak MW, Gaffney DJ, Elo LL, Zhang X, Mortazavi A (2016) A survey of best practices for RNA-seq data analysis. *Genome Biol* 17(1):13
2. Auer PL, Doerge RW (2010) Statistical design and analysis of RNA sequencing data. *Genetics* 185(2):405–416
3. Backman TWH, Girke T (2016) systemPipeR: NGS workflow and report generation environment. *BMC Bioinf* 17(1):388
4. Dona MS, Prendergast LA, Mathivanan S, Keerthikumar S, Salim A (2017) Powerful differential expression analysis incorporating network topology for next-generation sequencing data. *Bioinformatics* 33(10): 1505–1513

5. Rigaill G, Balzergue S, Brunaud V, Blondet E, Rau A, Rogier O, Caius J, Maugis-Rabusseau C, Soubigou-Taconnat L, Aubourg S, Lurin C, Martin-Magniette ML, Delannoy E (2016) Synthetic data sets for the identification of key ingredients for RNA-seq differential analysis. *Brief Bioinf* 19(1):65–76
6. Pai AA, Pritchard JK, Gilad Y (2015) The genetic and mechanistic basis for variation in gene regulation. *PLoS Genet* 11(1):e1004857
7. Zlatanova J, Van Holde KE (2016) Molecular biology: structure and dynamics of genomes and proteomes. Garland Sciences
8. Maston GA, Evans SK, Green MR (2006) Transcriptional regulatory elements in the human genome. *Ann Rev Genom Hum Genet* 7(1):29–59
9. Balaji S, Babu MM, Iyer LM, Luscombe NM, Aravind L (2006) Comprehensive analysis of combinatorial regulation using the transcriptional regulatory network of yeast. *J Mol Biol* 360(1):213–227
10. Ravasi T, Suzuki H, Cannistraci CV, Katayama S, Bajic VB, Tan K, Akalin A, Schmeier S, Kanamori-Katayama M, Bertin N, Carninci P, Daub CO, Forrest ARR, Gough J, Grimmond S, Han JH, Hashimoto T, Hide W, Hofmann O, Kawaji H, Kubosaki A, Lassmann T, van Nimwegen E, Ogawa C, Teasdale RD, Tegnér J, Lenhard B, Teichmann SA, Arakawa T, Ninomiya N, Murakami K, Tagami M, Fukuda S, Imamura K, Kai C, Ishihara R, Kitazume Y, Kawai J, Hume DA, Ideker T, Hayashizaki Y (2010) An atlas of combinatorial transcriptional regulation in mouse and man. *Cell* 140(5):744–752
11. Schilstra MJ, Nehaniv CL (2008) Bio-Logic: gene expression and the laws of combinatorial logic. *Artif Life* 14(1):121–133
12. Castel SE, Martienssen RA (2013) RNA interference in the nucleus: roles for small RNAs in transcription, epigenetics and beyond. *Nat Rev Genet* 14(2):100–112
13. Catalanotto C, Cogoni C, Zardo G (2016) MicroRNA in control of gene expression: an overview of nuclear functions. *Int J Mol Sci* 17(10):1712
14. Gonzalez-Zulueta M, Bender CM, Yang AS, Nguyen TD, Tornout JM, Jones PA, Beart RW (1995) Methylation of the 5' CpG island of the p16/CDKN2 tumor suppressor gene in normal and transformed human tissues correlates with gene silencing. *Cancer Res* 55(20):4531–4535
15. Herman JG, Baylin SB (2003) Gene silencing in cancer in association with promoter hypermethylation. *N Engl J Med* 349(21):2042–2054
16. Jones PA, Baylin SB (2007) The epigenomics of cancer. *Cell* 128(4):683–692
17. Li B, Carey M, Workman JL (2007) The role of chromatin during transcription. *Cell* 128(4):707–719
18. Kozak M (2005) Regulation of translation via mRNA structure in prokaryotes and eukaryotes. *Gene* 361(1–2):13–37
19. Sonenberg N, Hinnebusch AG (2009) Regulation of translation initiation in eukaryotes: mechanisms and biological targets. *Cell* 136(4):731–745
20. Gebauer F, Hentze MW (2004) Molecular mechanisms of translational control. *Nat Rev Mol Cell Biol* 5(10):827–835
21. Halbeisen RE, Galgano A, Scherrer T, Gerber AP (2008) Post-transcriptional gene regulation: from genome-wide studies to principles. *Cell Mol Life Sci* 65(5):798–813
22. Merchante C, Stepanova AN, Alonso JM (2017) Translation regulation in plants: an interesting past, an exciting present and a promising future. *Plant J* 90(4):628–653
23. Hutvágner G, Zamore PD (2002) A microRNA in a multiple-turnover RNAi enzyme complex. *Science* 297(5589):2056–2060
24. Valencia-Sánchez MA, Liu J, Hannon GJ, Parker R (2006) Control of translation and mRNA degradation by miRNAs and siRNAs. *Genes Dev* 20(5):515–524
25. Jackson RJ, Standart N (2007) How do microRNAs regulate gene expression? *Sci. STKE* 2007(367):re1
26. Kong YW, Cannell IG, de Moor CH, Hill K, Garside PG, Hamilton TL, Meijer HA, Dobbyn HC, Stoneley M, Spriggs KA, Willis AE, Bushell M (2008) The mechanism of micro-RNA-mediated translation repression is determined by the promoter of the target gene. *Proc Natl Acad Sci U S A* 105(26):8866–71
27. Wu L, Belasco JG (2008) Let me count the ways: mechanisms of gene regulation by miRNAs and siRNAs. *Mol Cell* 29(1):1–7
28. Tu K, Yu H, Hua YJ, Li YY, Liu L, Xie L, Li YX (2009) Combinatorial network of primary and secondary microRNA-driven regulatory mechanisms. *Nucleic Acids Res* 37(18):5969–5980

29. Friedman RC, Farh KKH, Burge CB, Bartel DP (2009) Most mammalian mRNAs are conserved targets of microRNAs. *Genome Res* 19(1):92–105
30. Guil S, Esteller M (2015) RNA-RNA interactions in gene regulation: the coding and noncoding players. *Trends Biochem Sci* 40(5):248–256
31. Wright PR, Georg J, Mann M, Sorescu DA, Richter AS, Lott S, Kleinkauf R, Hess WR, Backofen R (2014) CopraRNA and IntaRNA: predicting small RNA targets, networks and interaction domains. *Nucleic Acids Res* 42:W1
32. Salari R, Backofen R, Sahinalp SC (2010) Fast prediction of RNA-RNA interaction. *Algorithms Mol Biol* 5(1):5
33. Lai D, Meyer IM (2016) A comprehensive comparison of general RNA-RNA interaction prediction methods. *Nucleic acids Res* 44(7):e61
34. Engreitz JM, Sirokman K, McDonel P, Shishkin AA, Surka C, Russell P, Grossman SR, Chow AY, Guttman M, Lander ES (2014) RNA-RNA interactions enable specific targeting of noncoding RNAs to nascent pre-mRNAs and chromatin sites. *Cell* 159(1):188–199
35. Liang H, Li WH (2007) MicroRNA regulation of human protein-protein interaction network. *RNA (New York, NY)* 13(9):1402–1408
36. Henkin TM (2008) Riboswitch RNAs: using RNA to sense cellular metabolism. *Genes Dev* 22(24):3383–3390
37. Biggs PJ, Collins LJ (2011) RNA networks in prokaryotes I: CRISPRs and riboswitches. *Adv Exp Med Biol* 722:209–220
38. Serganov A, Patel DJ (2012) Metabolite recognition principles and molecular mechanisms underlying riboswitch function. *Ann Rev Biophys* 41(1):343–370
39. Wang Y, Liu CL, Storey JD, Tibshirani RJ, Herschlag D, Brown PO (2002) Precision and functional specificity in mRNA decay. *Proc Natl Acad Sci U S A* 99(9):5860–5865
40. Yang E, van Nimwegen E, Zavolan M, Rajewsky N, Schroeder M, Magnasco M, Darnell JE (2003) Decay rates of human mRNAs: correlation with functional characteristics and sequence attributes. *Genome Res* 13(8):1863–1872
41. Kuwano Y, Kim HH, Abdelmohsen K, Pullmann R, Martindale JL, Yang X, Gorospe M (2008) MKP-1 mRNA stabilization and translational control by RNA-binding proteins HuR and NF90. *Mol Cell Biol* 28(14):4562–4575
42. Mattick JS, Makunin IV (2006) Non-coding RNA. *Hum Mol Genet* 15 Spec No 1:R17–29
43. Cooper GM (2000) Regulation of protein function. In: *The cell: a molecular approach*, 2nd edn. Sinauer Associates, Sunderland
44. Walsh CT, Garneau-Tsodikova S, Gatto GJ (2005) Protein posttranslational modifications: the chemistry of proteome diversifications. *Angew Chem Int Ed* 44(45):7342–7372
45. Hunter T (1995) Protein kinases and phosphatases: the Yin and Yang of protein phosphorylation and signaling. *Cell* 80(2):225–236
46. Lizcano JM, Alessi DR (2002) The insulin signalling pathway. *Current Biology* 12(7):R236–R238
47. Szklarczyk D, Morris JH, Cook H, Kuhn M, Wyder S, Simonovic M, Santos A, Doncheva NT, Roth A, Bork P, Jensen LJ, Von Mering C (2017) The STRING database in 2017: quality-controlled protein-protein association networks, made broadly accessible. *Nucleic Acids Res* 45(D1):D362–D368
48. Olson TS, Dice JF (1989) Regulation of protein degradation rates in eukaryotes. *Curr Opin Cell Biol* 1(6):1194–1200
49. Varshavsky A (2005) Regulated protein degradation. In: *Trends in biochemical sciences*, vol 30, pp 283–286
50. Lecker SH (2006) Protein degradation by the ubiquitin-proteasome pathway in normal and disease states. *J Am Soc Nephrol* 17(7):1807–1819
51. Belle A, Tanay A, Bitincka L, Shamir R, O’Shea EK (2006) Quantification of protein half-lives in the budding yeast proteome. *Proc Natl Acad Sci U S A* 103(35):13004–13009
52. Vogel C, Marcotte EM (2012) Insights into the regulation of protein abundance from proteomic and transcriptomic analyses. *Nat Rev Genet* 13(4):227–232
53. Gilad Y, Rifkin SA, Pritchard JK (2008) Revealing the architecture of gene regulation: the promise of eQTL studies. *Trends Genet* 24(8):408–415
54. Jansen RC, Nap JP (2001) Genetical genomics: the added value from segregation. *Trends Genet* 17(7):388–391

55. Gaffney DJ (2013) Global properties and functional complexity of human gene regulatory variation. *PLoS Genet* 9(5):e1003501
56. Veyrieras JB, Kudaravalli S, Kim SY, Dermitzakis ET, Gilad Y, Stephens M, Pritchard JK (2008) High-resolution mapping of expression-QTLs yields insight into human gene regulation. *PLoS Genet* 4(10):e1000214
57. Albert FW, Kruglyak L (2015) The role of regulatory variation in complex traits and disease. *Nat Rev Genet* 16(4):197–212
58. Bessière C, Taha M, Petitprez F, Vandel J, Marin JM, Bréhelin L, Lèbre S, Lecellier CH (2018) Probing instructions for expression regulation in gene nucleotide compositions. *PLoS Comput Biol* 14(1):e1005921
59. Rinn JL, Chang HY (2012) Genome regulation by long noncoding RNAs. *Ann Rev Biochem* 81(1):145–166
60. Quinn JJ, Chang HY (2016) Unique features of long non-coding RNA biogenesis and function. *Nat Rev Genet* 17(1):47–62
61. Wang KC, Chang HY (2011) Molecular mechanisms of long noncoding RNAs. *Mol Cell* 43(6):904–914
62. Mercer TR, Dinger ME, Mattick JS (2009) Long non-coding RNAs: insights into functions. *Nat Rev Genet* 10(3):155–159
63. Ponting CP, Oliver PL, Reik W (2009) Evolution and functions of long noncoding RNAs. *Cell* 136(4):629–641
64. Geisler S, Coller J (2013) RNA in unexpected places: long non-coding RNA functions in diverse cellular contexts. *Nat Rev Mol Cell Biol* 14(11):699–712
65. Mendes P, Sha W, Ye K (2003) Artificial gene networks for objective comparison of analysis algorithms. In: *Bioinformatics*, vol 19
66. den Buleke T, Van Leemput K, Naudts B, van Remortel P, Ma H, Verschoren A, De Moor B, Marchal K, Van den Bulcke T, Van Leemput K, Naudts B, van Remortel P, Ma H, Verschoren A, De Moor B, Marchal K (2006) {SynTReN}: a generator of synthetic gene expression data for design and analysis of structure learning algorithms. *BMC Bioinf* 7(1):43
67. Ribeiro AS, Lloyd-Price J (2007) SGN Sim, a stochastic genetic networks simulator. *Bioinformatics* 23(6):777–779
68. Roy S, Werner-Washburne M, Lane T (2008) A system for generating transcription regulatory networks with combinatorial control of transcription. *Bioinformatics* 24(10):1318–1320
69. Di Camillo B, Toffolo G, Cobelli C (2009) A gene network simulator to assess reverse engineering algorithms. *Ann N Y Acad Sci* 1158:125–142
70. Haynes BC, Brent MR (2009) Benchmarking regulatory network reconstruction with GRENDel. *Bioinformatics* 25(6):801–807
71. Hache H, Wierling C, Lehrach H, Herwig R (2009) GeNGe: systematic generation of gene regulatory networks. *Bioinformatics* 25(9):1205–1207
72. Schaffter T, Marbach D, Floreano D (2011) GeneNetWeaver: in silico benchmark generation and performance profiling of network inference methods. *Bioinformatics* 27(16):2263–2270
73. Pinna A, Soranzo N, Hoeschele I, de la Fuente A (2011) Simulating systems genetics data with SysGenSIM. *Bioinformatics* 27(17):2459–2462
74. Tripathi S, Lloyd-Price J, Ribeiro A, Yli-Harja O, Dehmer M, Emmert-Streib F (2017) sgnesR: an R package for simulating gene expression data from an underlying real gene network structure considering delay parameters. *BMC Bioinf* 18(1):325
75. Erdős P, Rényi A (1959) On random graphs. *Publ Math Debrecen* 6:290–297
76. Kauffman SA (1969) Metabolic stability and epigenesis in randomly constructed genetic nets. *J Theor Biol* 22(3):437–467
77. Barabási AL, Oltvai ZN (2004) Network biology: understanding the cell's functional organization. *Nat Rev Genet* 5(2):101–113
78. Jeong H, Tombor B, Albert R, Oltval ZN, Barabási AL (2000) The large-scale organization of metabolic networks. *Nature* 407(6804):651–654
79. Wagner A, Fell DA (2001) The small world inside large metabolic networks. *Proc R Soc B: Biol Sci* 268(1478):1803–1810
80. Albert R (2007) Network inference, analysis, and modeling in systems biology. *Plant cell* 19(11):3327–3338
81. Watts DJ, Strogatz SH (1998) Collective dynamics of ‘small-world’ networks. *Nature* 393(6684):440–442
82. Barabási AL, Albert R (1999) Emergence of scaling in random networks. *Science* 286(5439):509–512

83. Albert R, Barabási AL (2000) Topology of evolving networks: local events and universality. *Phys Rev Lett* 85(24):5234–5237
84. Featherstone DE, Broadie K (2002) Wrestling with pleiotropy: genomic and topological analysis of the yeast gene expression network. *BioEssays* 24(3):267–274
85. Ravasz E, Somera AL, Mongru DA, Oltvai ZN, Barabási AL (2002) Hierarchical organization of modularity in metabolic networks. *Science* 297(5586):1551–1555
86. Bollobás B, Borgs C, Chayes J, Riordan O (2003) Directed scale-free graphs. Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms, pp 132–139
87. Guelzim N, Bottani S, Bourgine P, Képès F (2002) Topological and causal structure of the yeast transcriptional regulatory network. *Nat Genet* 31(1):60–63
88. Sanguinetti G, Noirel J, Wright PC (2008) MMG: A probabilistic tool to identify submodules of metabolic pathways. *Bioinformatics* 24(8):1078–1084
89. Milo R, Shen-Orr SS, Itzkovitz S, Kashtan N, Chklovskii D, Alon U (2002) Network motifs: Simple building blocks of complex networks. *Science* 298(5594):824–827
90. Shen-Orr SS, Milo R, Mangan S, Alon U (2002) Network motifs in the transcriptional regulation network of Escherichia coli. *Nat Genet* 31(1):64–68
91. Zhu D, Qin ZS (2005) Structural comparison of metabolic networks in selected single cell organisms. *BMC Bioinf* 6:8
92. Alon U (2007) Network motifs: theory and experimental approaches. *Nat Rev Genet* 8(6):450–461
93. Rosenfeld N, Elowitz MB, Alon U (2002) Negative autoregulation speeds the response times of transcription networks. *J Mol Biol* 323(5):785–793
94. Mangan S, Alon U (2003) Structure and function of the feed-forward loop network motif. *Proc Natl Acad Sci U S A* 100(21):11980–11985
95. Alon U (2006) An introduction to systems biology: design principles of biological circuits. CRC Press, Boca Raton
96. Pržulj N, Corneil DG, Jurisica I (2004) Modeling interactome: scale-free or geometric? *Bioinformatics* 20(18):3508–3515
97. Marbach D, Schaffter T, Mattiussi C, Floreano D (2009) Generating realistic in silico gene networks for performance assessment of reverse engineering methods. *J Comput Biol* 16(2):229–239
98. de Jong H (2002) Modeling and simulation of genetic regulatory systems: a literature review. *J Comput Biol J Comput Mol Cell Biol* 9(1):67–103
99. Higham DJ (2008) Modeling and simulating chemical reactions. *SIAM Rev* 50(2):347–368
100. D’haeseleer P, Wen X, Fuhrman S, Somogyi R (1999) Linear modeling of mRNA expression levels during CNS development and injury. *Pac Symp Biocomput* 52:41–52
101. Yeung MKS, Tegnér J, Collins JJ (2002) Reverse engineering gene networks using singular value decomposition and robust regression. *Proc Natl Acad Sci U S A* 99(9):6163–8
102. Ackers GK, Johnson AD, Shea MA (1982) Quantitative model for gene regulation by lambda phage repressor. *Proc Natl Acad Sci U S A* 79(4):1129–1133
103. Bintu L, Buchler NE, Garcia HG, Gerland U, Hwa T, Kondev J, Phillips R (2005) Transcriptional regulation by the numbers: models. *Curr Opin Genet Dev* 15(2):116–124
104. Marbach D, Prill RJ, Schaffter T, Mattiussi C, Floreano D, Stolovitzky G (2010) Revealing strengths and weaknesses of methods for gene network inference. *Proc Natl Acad Sci U S A* 107(14):6286–6291
105. Hache H, Lehrach H, Herwig R (2009) Reverse engineering of gene regulatory networks: a comparative study. *EURASIP J Bioinf Syst Biol* 2009(1):617281
106. Cao Y, Samuels DC (2009) Discrete stochastic simulation methods for chemically reacting systems. *Methods Enzymol* 454(08):115–140
107. Ross IL, Browne CM, Hume DA (1994) Transcription of individual genes in eukaryotic cells occurs randomly and infrequently. *Immunol Cell Biol* 72(2):177–185
108. McAdams HH, Arkin A (1999) It’s a noisy business! Genetic regulation at the nanomolar scale. *Trends Genet* 15(2):65–69

109. Wilkinson DJ (2009) Stochastic modelling for quantitative description of heterogeneous biological systems. *Nat Rev Genet* 10(2):122–133
110. Wilkinson DJ (2012) Stochastic modelling for systems biology, 2nd edn. CRC Press, Boca Raton
111. Gillespie DT (2007) Stochastic simulation of chemical kinetics. *Ann Rev Phys Chem* 58(1):35–55
112. El Samad H, Khammash M, Petzold L, Gillespie D (2005) Stochastic modelling of gene regulatory networks. *Int J Robust Nonlinear Control* 15(15):691–711
113. Gillespie DT (1977) Exact stochastic simulation of coupled chemical reactions. *J Phys Chem* 81(25):2340–2361
114. Gibson MA, Bruck J (2000) Efficient exact stochastic simulation of chemical systems with many species and many channels. *J Phys Chem A* 104(9):1876–1889
115. McCollum JM, Peterson GD, Cox CD, Simpson ML, Samatova NF (2006) The sorting direct method for stochastic simulation of biochemical systems with varying reaction execution behavior. *Comput Biol Chem* 30(1):39–49
116. Pahle J (2009) Biochemical simulations: Stochastic, approximate stochastic and hybrid approaches. *Brief Bioinf* 10(1):53–64
117. Turner TE, Schnell S, Burrage K (2004) Stochastic approaches for modelling in vivo reactions. *Comput Biol Chem* 28(3):165–178
118. Karlebach G, Shamir R (2008) Modelling and analysis of gene regulatory networks. *Nat Rev Mol Cell Biol* 9(10):770–780
119. Gillespie DT (2000) Chemical Langevin equation. *J Chem Phys* 113(1):297–306
120. Cao Y, Gillespie DT, Petzold LR (2005) The slow-scale stochastic simulation algorithm. *J Chem Phys* 122(1):14116
121. Wery M, Kwapisz M, Morillon A (2011) Noncoding RNAs in gene regulation. Wiley Interdiscip Rev Syst Biol Med 3(6):728–738
122. Morris KV, Mattick JS (2014) The rise of regulatory RNA. *Nat Rev Genet* 15(6):423–437
123. Holoch D, Moazed D (2015) RNA-mediated epigenetic regulation of gene expression. *Nat Rev Genet* 16(2):71–84
124. Milo R, Phillips R (2016) Cell biology by the numbers. Garland Science, Taylor & Francis Group, LLC, New York
125. Milo R, Jorgensen P, Moran U, Weber G, Springer M (2009) BioNumbers the database of key numbers in molecular and cell biology. *Nucleic Acids Res* 38(suppl. 1): D750–D753
126. Lowe R, Shirley N, Bleackley M, Dolan S, Shafee T (2017) Transcriptomics technologies. *PLoS Comput Biol* 13(5):e1005457
127. Rocke DM, Durbin B (2001) A model for measurement error for gene expression arrays. *J Comput Biol* 8(6):557–569
128. Irizarry RA, Warren D, Spencer F, Kim IF, Biswal S, Frank BC, Gabrielson E, Garcia JG, Geoghegan J, Germino G, Griffin C, Hilmer SC, Hoffman E, Jedlicka AE, Kawasaki E, Martínez-Murillo F, Morsberger L, Lee H, Petersen D, Quackenbush J, Scott A, Wilson M, Yang Y, Ye SQ, Yu W (2005) Multiple-laboratory comparison of microarray platforms. *Nat Methods* 2(5):345–349
129. Stolovitzky G, Monroe D, Califano A (2007) Dialogue on reverse-engineering assessment and methods: the DREAM of high-throughput pathway inference. *Ann N Y Acad Sci* 1115:1–22
130. Bintu L, Buchler NE, Garcia HG, Gerland U, Hwa T, Kondev J, Kuhlman T, Phillips R (2005) Transcriptional regulation by the numbers: applications. *Curr Opin Genet Dev* 15(2):125–135



Chapter 16

Scalable Inference of Ordinary Differential Equation Models of Biochemical Processes

Fabian Fröhlich, Carolin Loos, and Jan Hasenauer

Abstract

Ordinary differential equation models have become a standard tool for the mechanistic description of biochemical processes. If parameters are inferred from experimental data, such mechanistic models can provide accurate predictions about the behavior of latent variables or the process under new experimental conditions. Complementarily, inference of model structure can be used to identify the most plausible model structure from a set of candidates, and, thus, gain novel biological insight. Several toolboxes can infer model parameters and structure for small- to medium-scale mechanistic models out of the box. However, models for highly multiplexed datasets can require hundreds to thousands of state variables and parameters. For the analysis of such large-scale models, most algorithms require intractably high computation times. This chapter provides an overview of the state-of-the-art methods for parameter and model inference, with an emphasis on scalability.

Key words Parameter estimation, Uncertainty analysis, Ordinary differential equations, Large-scale models

1 Introduction

In systems biology, ordinary differential equation (ODE) models have become a standard tool for the analysis of biochemical reaction networks [1]. The ODE models can be derived from information about the underlying biochemical processes [2, 3] and allow the systematic integration of prior knowledge. ODE models are particularly valuable as they can be used to predict the temporal evolution of latent variables [4, 5]. Moreover, they provide executable formulations of biological hypotheses and therefore allow the rigorous falsification of hypotheses [6–11], thereby deepening the biological understanding. Furthermore, ODE models have been applied to derive model-based biomarkers [12–14] that enable a personalized design of targeted therapies in precision medicine.

To construct predictive models, model parameters have to be inferred from experimental data. This inference requires the repeated numerical simulation of the model. Consequently, parameter inference is computationally demanding if the required computation time for the numerical solution is high. For many applications, small- and medium-scale models, i.e., models consisting of a small number of species belonging to the core pathway, are sufficient for accurate prediction and hypothesis testing [2, 15–17]. Low-dimensional models can be derived directly or obtained from large-scale models by model reduction, e.g., by lumping multistep reactions to one-step reactions [18] or by assuming time-scale separation [19]. These small- and medium-scale models can be analyzed using established toolboxes implementing the state-of-the-art methods [20–22].

For models describing few conditions, e.g., the response of a single cell line to a small set of stimulations, the lumping and ignoring of processes might be appropriate. Yet, if a model is to be used for a wide range of conditions, e.g., to describe the responses of many cell lines to many different stimuli, a detailed mechanistic description is required [4, 23, 24] as simplifications only hold for selected conditions. Detailed mechanistic, generalizing models appear particularly valuable for precision medicine, where the model must accurately predict treatment outcomes for many different patients [25]. These comprehensive models, typically describing most species in multiple different pathways including respective crosstalk, can easily describe thousands of molecular species involved in thousands of reactions with thousands of parameters. For such models, parameter inference is often intractable as it is prohibitively computationally expensive [26, 27].

Beyond model parameters, also the model structure might be unknown, e.g., the biochemical reactions or their regulations might be unknown [28, 29]. Then, inference of model structure can be used to generate new mechanistic insights. For ODE models, this can be achieved by constructing multiple model candidates corresponding to different biological hypotheses. These hypotheses can be falsified using model selection criteria such as the Akaike Information Criterion (AIC) [30], or Bayesian Information Criterion (BIC) [31]. For large models, the high number of mutually nonexclusive hypotheses is not uncommon and typically leads to a combinatorial explosion of the number of model candidates (*see*, e.g., [32–34]). Computing the AIC or BIC for all model candidates for comparison would require parameter inference for each model candidate and may seem futile, given that parameter inference for a single model can already be challenging.

In this chapter, we will review scalable methods that render model parameter and model structure inference tractable for large-scale ODE models, which have hundreds to thousands of molecular species, biochemical reactions, and parameters. For parameter

inference, we will focus on different gradient-based optimization schemes and describe their scaling properties with respect to the number of molecular species and number of parameters. For inference of model structure, we will focus on complexity penalization schemes that allow the simultaneous inference of model structure and parameters and thus scale better than linearly with the number of model candidates. For alternative methods for learning networks of ODEs, also read Chapter 11.

2 Inference of Model Parameters

An ODE model describes the temporal evolution of the concentrations of n_x different molecular species x_i . The dynamics of \mathbf{x} are determined by the vector field f and the initial condition \mathbf{x}_0 :

$$\dot{\mathbf{x}} = f(t, \mathbf{x}, \boldsymbol{\theta}), \quad \mathbf{x}(t_0) = \mathbf{x}_0(\boldsymbol{\theta}). \quad (1)$$

Both of these functions may depend on the unknown parameters $\boldsymbol{\theta} \in \Theta \subset \mathbb{R}^{n_\theta}$ such as kinetic rates. The parameter domain Θ can constrain the parameter values to biologically reasonable numbers.

In general, \mathbf{x} and f can also be derived from a discretization of a partial differential equation model [35–38] or describe the temporal evolution of empirical moments of stochastic processes [39–41].

Experiments usually provide information about n_y different observables y_i which depend linearly or nonlinearly on the concentrations \mathbf{x} . A direct measurement of \mathbf{x} is usually not possible. The dependence of the observable on concentrations and parameters is described by:

$$\mathbf{y}(t, \boldsymbol{\theta}) = h(\mathbf{x}(t, \boldsymbol{\theta}), \boldsymbol{\theta}). \quad (2)$$

2.1 Problem Formulation

To build predictive models, the model parameters $\boldsymbol{\theta}$ have to be inferred from experimental data. This inference problem is usually formulated as an optimization problem. In this optimization problem, an objective function $J(\boldsymbol{\theta})$, describing the difference between measurements and simulation, is minimized. In the following, we will first formulate the optimization problem and then discuss the methods to solve it efficiently.

Experimental data are subject to measurement noise. A common assumption is that the measurement noise for all time points t_j and observables y_i is additive and independent, normally distributed for all time points:

$$\bar{y}_{ij} = y_i(t_j, \boldsymbol{\theta}) + \epsilon_{ij}, \quad \epsilon_{ij} \stackrel{iid}{\sim} \mathcal{N}(0, \sigma_{ij}^2(\boldsymbol{\theta})). \quad (3)$$

At each of the T time points t_j , up to n_y different measurements \bar{y}_{ij} can be recorded in the experimental data $\mathcal{D} = \{(\bar{y}_{ij})_{i=1}^{n_y}, t_j\}_{j=1}^T$. As the standard deviation of the measurement noise is potentially unknown, we model it as $\sigma_{ij}(\boldsymbol{\theta})$. This yields the likelihood function:

$$p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{j=1}^T \prod_{i=1}^{n_y} \frac{1}{\sqrt{2\pi\sigma_{ij}^2(\boldsymbol{\theta})}} \exp\left(-\frac{1}{2}\left(\frac{\bar{y}_{ij} - y_i(t_j, \boldsymbol{\theta})}{\sigma_{ij}(\boldsymbol{\theta})}\right)^2\right). \quad (4)$$

Other plausible noise assumptions include log-normal distributions, which correspond to multiplicative measurement noise [42]. Distributions with heavier tails, such as the Laplace distribution, can be used to increase robustness to outliers in the data [43].

The model can be inferred from experimental data by maximizing the likelihood (4), which yields the maximum likelihood estimate (MLE). However, the evaluation of the likelihood function, $p(\mathcal{D}|\boldsymbol{\theta})$, involves the computation of several products, which can be numerically unstable. Thus, the negative log-likelihood:

$$J(\boldsymbol{\theta}) = -\log(p(\mathcal{D}|\boldsymbol{\theta})) = \frac{1}{2} \sum_{i=1}^{n_y} \sum_{j=1}^T \log\left(2\pi\sigma_{ij}^2(\boldsymbol{\theta})\right) + \left(\frac{\bar{y}_{ij} - y_i(t_j, \boldsymbol{\theta})}{\sigma_{ij}(\boldsymbol{\theta})}\right)^2 \quad (5)$$

is often used as objective function for minimization. As the logarithm is a strictly monotonously increasing function, the minimization of $J(\boldsymbol{\theta}) = -\log(p(\mathcal{D}|\boldsymbol{\theta}))$ is equivalent to the maximization of $p(\mathcal{D}|\boldsymbol{\theta})$. Therefore, the corresponding minimization problem:

$$\boldsymbol{\theta}^* = \arg \min_{\boldsymbol{\theta} \in \Theta} J(\boldsymbol{\theta}), \quad (6)$$

will infer the MLE parameters. If the noise variance σ_{ij}^2 does not depend on the parameters $\boldsymbol{\theta}$, (5) is a weighted least-squares objective function. As we will discuss later, this least-squares structure can be exploited by several optimization methods.

If prior knowledge about the parameters is available, this can be encoded in a prior probability $p(\boldsymbol{\theta})$. According to Bayes' theorem [44], the posterior probability $p(\boldsymbol{\theta}|\mathcal{D})$ is defined by:

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D})}. \quad (7)$$

The evidence $p(\mathcal{D})$ is usually difficult to compute. However, as it is independent of $\boldsymbol{\theta}$, the respective term can be omitted for parameter inference. This yields the objective function:

$$J(\boldsymbol{\theta}) = -\log(p(\mathcal{D}|\boldsymbol{\theta})) - \log(p(\boldsymbol{\theta})), \quad (8)$$

which corresponds to the log-posterior up to the constant $\log(p(\mathcal{D}))$. The respective optimization problem yields the maximum a posteriori estimate (MAP).

2.1.1 Properties of the Optimization Problem

The optimization problem (5) is convex in $y_i(t_j, \theta)$, but usually nonconvex in θ . Thus, the objective function $J(\theta)$ can possess local minima and saddle points. Local minima can be problematic as optimization algorithms may get stuck, yielding a suboptimal agreement between experimental data and model simulation. Interestingly, recent literature suggests that saddle points might affect the efficiency of optimization more severely than local minima [45]. For unconstrained problems, saddle points and local minima are both stationary points θ^* at which the gradient vanishes

$$\nabla J(\theta^*) = 0, \quad (9)$$

that is, they both satisfy a necessary local optimality condition (see Fig. 1a left for an example). The sufficient condition for a local minimum is the positive definiteness of the Hessian $\nabla^2 J(\theta^*)$, which indicates that there are only directions of positive curvature. For a saddle point, the Hessian $\nabla^2 J(\theta^*)$ is indefinite or semi-definite, which indicates that there may be directions of negative or zero curvature.

The dependence of the number of local minima and saddle points for ODE models on the number of parameters is poorly understood. For deep learning problems, an exponential increase in the number of local minima with the number of parameters is primarily attributed to parameter permutation symmetries [47], which are rare in ODE models. Yet, for deep learning problems, saddle points are also problematic as they affect the performance of local optimization methods [45]. Arguments for an exponential increase in stationary points with the number of parameters are often based on random matrix theory [45] and rely on strong assumptions on the distribution of entries in the Hessian of the objective function $J(\theta)$. These assumptions have to be rigorously checked as they can lead to wrong conclusions, as shown for the stability of ODEs [48]. As the objective function $J(\theta)$ depends on the solution to an ODE, the validity of such assumptions is not evident and difficult to assess rigorously. For saddle points, we are not aware of any rigorous evaluation. Thus, the exact dependence of the number of local minima and saddle points on the parameter dimensionality remains elusive.

2.2 Optimization Methods

The infamous No Free Lunch Theorem for optimization [49] states that there exists no single optimization method that performs best on all classes of optimization problems. Accordingly, empirical evidence as well as a careful analysis of the problem structure should be considered when selecting a suitable optimization

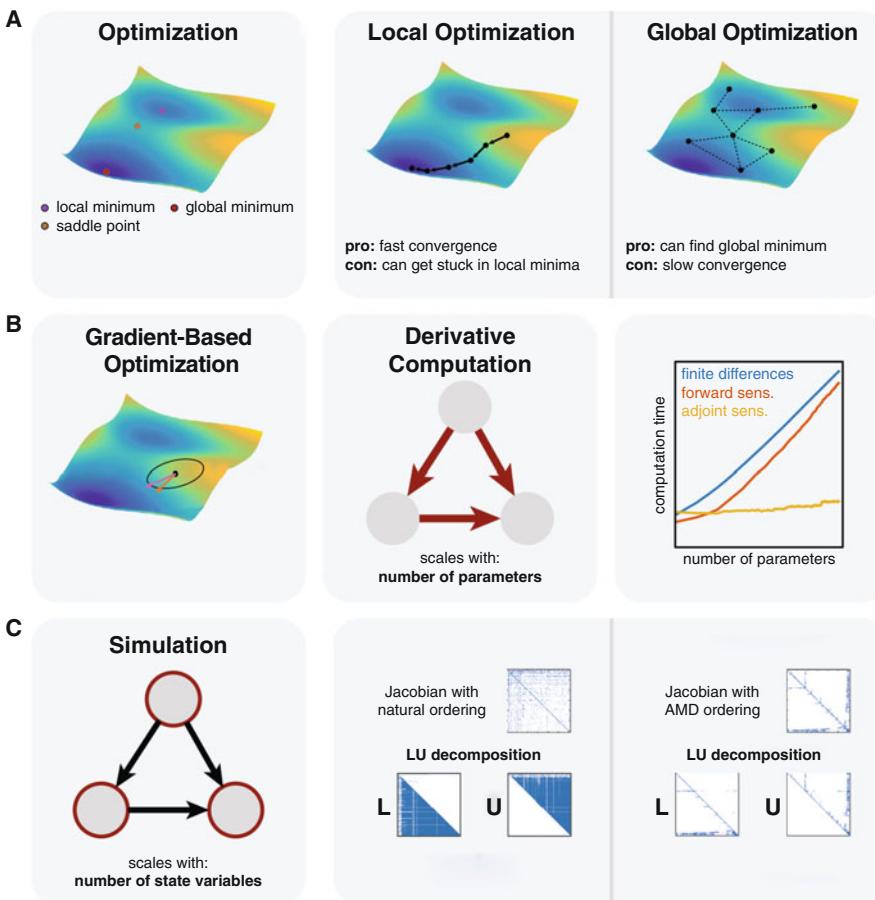


Fig. 1 Overview of numerical methods and scaling properties for parameter inference. **(a)** Schematic scaling properties of optimization. Icons and properties of local and global optimizations are shown on the right. **(b)** Examples of gradient-based methods to determine parameter updates. Computation times of different approaches for gradient computation are shown on the right. **(c)** Schematic scaling properties of simulation algorithms. Dense and sparse direct linear solvers are compared on the right. Reordering was performed using the Approximative Minimum Degree (AMD) ordering algorithm [46]

method. The plethora of different optimization methods commonly used in systems biology can be classified as:

1. **local** and **global** methods, as well as
2. **gradient-based** and **derivative-free** methods.

Local methods search for local optima, while global methods search for global minima. The separation into local and global methods is often not clear-cut. Thus, methods, such as simulated annealing, are sometimes classified as local and sometimes as global methods [50–52]. Therefore, the following paragraph contains many soft statements that should only serve as guidelines. Gradient-based methods exploit first and potentially higher-order derivatives of the objective function, while derivative-free methods solely use the objective function.

Table 1
Examples for local and global, as well as derivative-free and gradient-based optimization algorithms

	Derivative-free	Gradient-based
Local	Pattern search [54]	Gradient descent [56]
	Nelder–Mead [57]	Newton’s method [56]
	Hill climbing [58]	Levenberg–Marquardt [59, 60]
Global	Genetic algorithm [61]	Multi-start [51]
	Particle swarm [62]	Scatter search [63]
	Simulated annealing [64]	Clustering search [65]

Local methods construct a sequence of trial points that successively decrease the objective function values (*see* Fig. 1a middle). This procedure is usually faster than global methods, but can get stuck in local minima [53]. Most local derivative-free methods are direct search methods [54]. In contrast to local methods, global methods often rely on a population of trial points which are iteratively refined (*see* Fig. 1a right). This can increase the chance of reaching the global minimum, but usually slower [53]. Global derivative-free methods mostly employ stochastic schemes, which are often inspired by nature [55], while global gradient-based methods usually perform repeated local optimizations. Examples of local and global as well as respective derivative-free and gradient-based methods are given in Table 1.

Not all global methods are guaranteed to converge to the global minimum [51]. Convergence to the global minimum is only guaranteed for rigorous and (asymptotically) complete global methods, such as branch-and-bound [66] and grid search [51]. As long as only local information, i.e., function values and respective parameter derivatives, is available, the termination of these methods will require exponentially expensive dense search [51, 67]. The termination of global methods is crucial, as it might be relatively easy to find the global minimum but comparably hard to guarantee that it is indeed the global minimum. Global information, such as Lipschitz constants, is rarely available for ODE problems, such that dense, i.e., exhaustive, search would be necessary for guaranteed convergence. As the parameters θ are generally continuous, dense search is rarely possible. Instead, meta-heuristics for termination and optimization are employed.

For many meta-heuristic methods, there exists little to no theoretical justification or convergence proofs [51]. Others may converge with probability arbitrarily close to 1, but might only do so after infinitely many function evaluations. In practice, many meta-heuristic algorithms even fail to work reliably for smooth,

convex problems with few parameters [68]. In fact, for some algorithms, non-convergence can even be proven mathematically [69]. Eventually, even a rigorous convergence guarantee will be useless if the convergence rate is too slow for practical purposes. For most methods, there exists a plethora of disparate variants, which renders comprehensive analysis of convergence proofs and convergence rates challenging. This is quite unsatisfying from a theoretical perspective. In practice, reasonable results can be obtained using global optimization methods [70–72]. Yet, usually no guarantees of global optimality can be given.

For the remainder of this chapter, we will primarily focus on global gradient-based optimization methods, which typically rely on repeated local optimization. For these meta-heuristic methods, a local optimization is started at (random) points in parameter space. The termination of these methods usually relies on a specified maximum number of local optimizations, but also Bayesian methods can be applied [73]. For the convergence to the global minimum, the local optimization has to be started in the region of attraction of the global optimum. Thus, the probability of convergence will depend on the relative volume of the region of attraction with respect to the search domain Ω . Several adaptive methods, such as scatter search [63] or clustering search [65], try to improve the chance of starting a local optimization in the region of attraction of the global optimum, but rely on the embeddedness of the global optimum [74]. The embeddedness of the global optimum characterizes how well local minima cluster and determines how indicative the objective function value at the starting point is of the chance of converging to the global minimum. We are not aware of any analysis of embeddedness of the global minimum for models in systems biology, but it is likely to be problem dependent. The resulting rate of convergence will be determined by the rate of convergence of the local method and the probability to sample a starting point from the region of attraction of the global optimum.

For most methods that employ repeated local optimization, the individual local optimization runs can trivially be run in parallel [75], which enables efficient use of high-performance computing structure. Moreover, multiple global runs can be asynchronously parallelized to enhance efficiency through cooperativity [76]. Following recent studies [38, 71], we deem this repeated local optimization a suitable candidate for scalable optimization and will in the following discuss the properties of respective local gradient-based methods (*see* Fig. 1b) in more detail.

2.2.1 Line-Search Methods

Line-search methods are local optimization methods that iteratively update the parameter values in direction $s \in \mathbb{R}^{n_\theta}$, such that the objective function value $J(\theta)$ is successively reduced:

$$\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} + \gamma \cdot \mathbf{s} \quad s.t. \quad J(\boldsymbol{\theta}_k) < J(\boldsymbol{\theta}_{k-1}), \gamma > 0, \quad (10)$$

where $\gamma \in \mathbb{R}_+$ controls the step size. For line-search, the update direction \mathbf{s} is fixed first and then a suitable γ is determined. The alternative to line-search methods are trust-region methods which first define a maximum step length and then find a suitable update direction. We will discuss trust-region methods in more detail in the following subsection. The classification in line-search or trust-region methods can be ambiguous and may depend on the specific implementation of the method. This will be discussed in more detail at the end of this subsection. In this chapter, we will follow the classification of Nocedal and Wright, who also provide an excellent discussion of the topic [56].

Line-search methods are particularly appealing as they reduce the possibly high-dimensional optimization problem to a sequence of one-dimensional problems of finding good values for γ . To ensure convergence, γ has to meet certain conditions [77, 78]. Methods to determine the step size that satisfy these conditions are sometimes referred to as globalization techniques. Note that enforcing these conditions only guarantees convergence to a local stationary point, i.e., a local minimum, local maximum, or saddle point, but not to a global minimum [56].

It is more or less well established that for local optimization, gradient-based methods should be used when applicable. Kolda et al. state in their review on direct search methods that “Today, most people’s first recommendation (including ours) to solve an unconstrained problem for which accurate first derivatives can be obtained would not be a direct search method, but rather a gradient-based method” [79]. Lewis et al. claim that “the success of quasi-Newton methods, when applicable, is now undisputed” [80]. Gradient-based methods, such as quasi-Newton methods, are applicable when the gradient (and the Hessian) of the objective function is continuous and can be computed accurately. By definition, the gradient $\nabla J(\boldsymbol{\theta})$ is only continuous with respect to parameters when the objective function is continuously differentiable. Higher-order continuous differentiability corresponds to continuity of the respective higher-order derivatives, such as the Hessian. For many noise distributions, such as the normal and log-normal distribution, the negative log-likelihood is infinitely often continuously differentiable with respect to the observables, given that a finite number of measurements are considered. Thus, the continuity of derivatives of the objective function (5) only depends on the continuity of derivatives of the model output (2). However, for particular noise distributions, such as the Laplace distribution, the negative log-likelihood may not be differentiable with respect to the model outputs. In the following, we will assume that both, (2) and (5), are twice continuously differentiable with respect to the parameters.

For continuously differentiable objective functions, an intuitive choice for \mathbf{s} is the gradient:

$$\mathbf{s}_{\text{grad}} = -\nabla J(\boldsymbol{\theta}_{k-1}). \quad (11)$$

Optimization methods using this search direction are called gradient descent methods. Locally, the gradient provides the steepest descent with respect to the euclidean norm. This means that the gradient points in the direction \mathbf{d} , with unit length in the euclidean norm, which yields the strongest decrease of $J(\boldsymbol{\theta})$ in a neighborhood around $\boldsymbol{\theta}$:

$$\frac{\nabla J(\boldsymbol{\theta})}{\|\nabla J(\boldsymbol{\theta})\|_2} = \underset{\mathbf{d}: \|\mathbf{d}\|_2=1}{\operatorname{argmin}} \nabla J(\boldsymbol{\theta})^T \mathbf{d}. \quad (12)$$

Yet, depending on the objective function, this neighborhood might be arbitrarily small, resulting in small values γ . This, for instance, is the case for objective functions with curved ridges, e.g., the Rosenbrock function [81], which can arise from (nonlinear) dependencies between parameters. Moreover, gradient descent methods take small steps in the vicinity of saddle points [45], which can lead to high iteration numbers or premature termination in individual optimization runs.

The issue of small step sizes is addressed in the Newton's method by including the Hessian $\nabla^2 J(\boldsymbol{\theta}_{k-1})$, which encodes the curvature, in the calculation of the search direction \mathbf{s} :

$$\mathbf{s}_{\text{newt}} = -\nabla^2 J(\boldsymbol{\theta}_{k-1})^{-1} \nabla J(\boldsymbol{\theta}_{k-1}). \quad (13)$$

For the classical Newton's method, the step size is fixed to $\gamma = 1$. However, many modern implementations implement Newton's method as line-search using 1 as default value and adapting the step size if necessary [56].

As the Hessian is symmetric, highly efficient methods such as Cholesky factorization can be used to solve this problem for convex optimization problems. However, for ODE models, the computation of the Hessian itself will usually be computationally far more expensive than the computation of the Newton step. Thus, the computational cost of solving (13) is usually negligible for objective functions depending on ODE models.

For nonconvex problems, the computation of the Newton step (13) may be an ill-posed or not even well-defined problem [56]. Moreover, the Newton step might not be a descent direction. The Newton step is only a direction of descent if the scalar product with the gradient is negative:

$$\mathbf{s}_{\text{newt}}^T \cdot \nabla J(\boldsymbol{\theta}_{k-1}) < 0. \quad (14)$$

By substituting the formula for the Newton step, we obtain the following inequality:

$$-\nabla J(\boldsymbol{\theta}_{k-1})^T (\nabla^2 J(\boldsymbol{\theta}_{k-1}))^{-1} \nabla J(\boldsymbol{\theta}_{k-1}) < 0, \quad (15)$$

which is globally satisfied only if the inverse of the Hessian $\nabla^2 J(\boldsymbol{\theta}_{k-1})^{-1}$ is positive definite, i.e., the problem (6) is convex. As previously discussed, (6) is typically nonconvex, thus simple Newton steps will not always yield a direction of descent. Moreover, in the vicinity of a saddle point, the Newton step may point in the direction of the saddle point, thus attracting the optimizer to saddle points [45, 82].

In the literature, several modifications of Newton's method that always yield descent directions have been proposed [59, 60, 83]. The Gauss–Newton [83] method exploits the least-squares structure of the objective function (5) and constructs a positive semi-definite approximation to $\nabla^2 J(\boldsymbol{\theta}_{k-1})$. Levenberg [59] and Marquardt [60] independently extended this method by introducing a dampening term in the step equation. This yields the Levenberg–Marquardt method:

$$-(\hat{\mathbf{H}}(\boldsymbol{\theta}_{k-1}) + \lambda \mathbf{I})\mathbf{s} = \nabla J(\boldsymbol{\theta}_{k-1}), \quad (16)$$

where $\hat{\mathbf{H}}$ is the positive semi-definite Gauss–Newton approximation to the Hessian, $\lambda \geq 0$ is the dampening factor, and \mathbf{I} is the identity matrix. The magnitude of the dampening factor λ regulates the conditioning of (16). The geometric interpretation of λ is that it allows an interpolation between a gradient and a Gauss–Newton step, where $\lambda = 0$ corresponds to a pure approximate Newton step. Due to the positive definiteness of the Gauss–Newton approximation, the respective methods cannot follow directions of negative curvature and are thus not attracted to saddle points, but again limited to small step sizes in the vicinity of saddle points [45].

As the Gauss–Newton method is limited to least-squares problems, the traditional formulation of the Levenberg–Marquardt method has the same limitation. However, it is possible to apply the dampening of the Hessian without the Gauss–Newton approximation. The resulting algorithms are often still referred to as Levenberg–Marquardt method [84]. In such a setting, the dampening is often chosen according to the smallest negative eigenvalue [56], using, e.g., the Lanczos method [85], to ensure the construction of a direction of descent. In the vicinity of saddle points, these methods can be modified to also follow directions of negative curvature [86].

An alternative to determine the dampening factor λ is the use of a trust-region method. A trust-region method fixes $\|\gamma s\| = \Delta$ and then determines an approximately matching λ [56]. Thus,

Levenberg–Marquardt algorithms can be implemented as line-search methods or as trust-region methods, depending on how λ and γ are computed. In the following, we will discuss trust-region methods more generally.

2.2.2 Trust-Region Methods

Line-search methods determine a search direction d first and then identify a good step size γ . Trust-region methods do the converse, by specifying a maximum step size first and then identifying a good search direction [56, 87, 88]. This allows trust-region methods to make large steps close to saddle points and always yield descent directions. Within the trust region, $J(\theta)$ is replaced by a local approximation, giving rise to the trust-region subproblem. Most trust-region algorithms use the objective function derivatives to construct a quadratic trust-region subproblem:

$$\min_{\mathbf{s} \in B_{\mathbf{D}, \Delta}(\theta)} \frac{1}{2} \mathbf{s}^T \nabla^2 J(\theta) \mathbf{s} + \mathbf{s}^T \nabla J(\theta), \quad (17)$$

where $B_{\mathbf{D}, \Delta}(\theta) = \{\mathbf{s} : \|\mathbf{D}(\mathbf{s} - \theta)\|_2 \leq \Delta\}$ is the trust region. The trust region is an ellipsoid with radius Δ and scaling matrix \mathbf{D} around the current parameter θ . The size of the trust region can be adapted over the course of iterations. Trust-region methods that do not use quadratic approximations use other local approximations, e.g., via radial basis functions [89].

Trust-region methods that solve the subproblem (17) exactly are not attracted to saddle points and not limited to small step sizes [45]. However, the quadratic problem (17) is usually difficult to solve exactly and is approximatively solved instead [56]. For convex problems, the dogleg method [90], which employs a linear combination of gradient and Newton step, can be applied. For nonconvex problems, the two-dimensional subspace minimization method [91, 92] can be used. The two-dimensional subspace minimization method dampens the Hessian and can be seen as a trust-region variant of the Levenberg–Marquardt method. The dogleg and two-dimensional subspace minimization method both reduce the trust-region subproblem to a two-dimensional problem, which renders the computational cost of determining the update step from a given gradient *per se* independent of the number of parameters of the underlying problem. This feature makes them particularly suited for large-scale problems. However, the dampening of the Hessian can again lead to small step sizes close to saddle points [45].

2.2.3 Implementation and Practical Considerations

State-of-the-art parameter inference toolboxes for computational biology, such as D2D [21], PESTO [93], MEIGO [94], and COPASI [20], feature a mix of local and global methods which include derivative-free and gradient-based methods (see Table 2). In terms of global, derivative-free methods, most toolboxes

Table 2
Implementations and interfaces of optimization methods in popular computational biology toolboxes

Toolbox	Global	Local	
		Derivative-free	Gradient-based
COPASI [20]	Evolutionary Programming [95]	Nelder Mead [57]	Levenberg–Marquardt [59, 60]
	Genetic algorithm [96]	Pattern search [54]	Steepest descent [56]
	Particle swarm [62]	PRAXIS [97]	Truncated Newton [98]
	Random search [99]		
	Simulated annealing [64]		
	Scatter search [63]		
	SRES [100]		
D2D [21]	fminsearchbnd (MATLAB)		arNLS (custom)
	Genetic algorithm (MATLAB)		CERES (Google)
	Multi-start (custom)		fmincon (MATLAB)
	Pattern search (MATLAB)		lsqnonlin (MATLAB)
	Particle swarm (custom)		STRSCNE [101]
	Simulated annealing (MATLAB)		TRESNEI [102]
MEIGO [94]	fminsearchbnd (MATLAB)	DHC [58]	fmincon (MATLAB)
	NOMAD [103]	Pattern Search [104]	IpOpt [105]
	Multi-start(custom)	SOLNP [106]	lsqnonlin (MATLAB)
	Scatter search [63]		MISQP [107]
			N2FB [108]
			NL2SOL [108]
PESTO [93]	Multi-start (custom)	BOBYQA [109]	fmincon (MATLAB)
	Particle swarm [110]	DHC [58]	lsqnonlin (MATLAB)
	MEIGO [94]		

Some toolboxes may feature variants of the cited algorithms. Some entries may be names of functions that feature multiple different algorithms

provide interfaces to Particle Swarm and Pattern Search methods. In terms of local, gradient-based methods, most toolboxes feature various flavors of the trust-region algorithm. All MATLAB toolboxes provide interfaces to the fmincon and lsqnonlin routines from the MATLAB Optimization Toolbox. Only COPASI provides the implementation of more basic algorithms such as Levenberg–Marquardt [59, 60], Truncated Newton, and Steepest Descent. In terms of global optimization schemes, all toolboxes employ either multi-start or scatter search algorithms [63].

Choosing a particular optimization method from this plethora of choices is not an easy task. There are no exhaustive studies that compare the full range of different optimization methods on a large set of problems. In some studies, gradient-based optimization algorithms perform best [38, 71], but others also show that derivative-free methods can perform well [111]. In general, a rigorous evaluation of optimization algorithms is highly involved, as there are small differences in the implementations of various algorithms. For example, STRSCNE [101], RESNEI [102], NL2SOL [108], lsqnonlin, and fmincon all implement trust-region algorithms and even for expert users it may be difficult to pinpoint differences between individual implementations. A recent study suggests substantial differences in the efficiency of various implementations of trust-region algorithms and identified lsqnonlin to be the best performing algorithm [112]. Even for a single implementation, the specification of hyper-parameters can have substantial impact on performance. Many algorithms require user specifications of technical parameters. Finding good values for these hyper-parameters may be challenging for nonexpert users and default values may not work for all problems.

As few researchers are experts in a large number of different optimization methods, a rigorous evaluation of multiple different algorithms is challenging. To circumvent this problem, a recent study [113] suggested a set of benchmark problems on which other researchers are invited to evaluate their algorithms. Yet, few algorithms have been evaluated on that benchmark so far [75, 76]. Complementarily, [114] suggests the construction of statistical models to assess the performance of methods and the effect of hyper-parameters.

The different optimization algorithms we outlined in this section rely on evaluations of the objective function, its gradient, or even its Hessian. In the following sections, we will discuss the methods to evaluate these terms.

2.3 Simulation

The objective function and its gradient are typically not available in closed form, but have to be computed numerically. For large-scale models, the computational cost of computing the objective function and its gradient is high, which makes parameter estimation computationally demanding. Depending on the class of the employed model and simulation algorithm, the computation time will depend on different features of the underlying model, which we will discuss in detail in the following.

The timescales of biochemical processes span multiple orders of magnitude [115, 116]. As comprehensive models often cover a large variety of different biological processes, they are particularly prone to possess multiple timescales [117]. This results in the stiffness of corresponding ODEs [118]. As the stiffness of the equations typically depends on the choice of parameters, it is rarely possible to assess the stiffness *a priori*. Consequently, it is always

advisable to use implicit solvers, which can adequately handle stiffness, for parameter inference [119].

2.3.1 Implicit Methods

For stiff problems, implicit differential equation solvers from the fully implicit Runge–Kutta solver family [120], the Singly Diagonally Implicit Runge–Kutta solver family [121], or the Rosenbrock solver family [122] should be used. These solvers compute the state variables at the next time step ξ_i based on the state variables at previous iterations $x(\xi_{i-1}), x(\xi_{i-2}), x(\xi_{i-3}), \dots$ by solving an implicit equation:

$$G(\mathbf{x}(\xi_i), \mathbf{x}(\xi_{i-1}), \mathbf{x}(\xi_{i-2}), \mathbf{x}(\xi_{i-3}), \dots) = 0,$$

where the function G depends on the choice of the method and on the right-hand side of the differential equation f .

For single-step methods, such as the Runge–Kutta-type solvers, the function G will only depend on $x(\xi_{i-1})$, and not on previous values. For implicit Runge–Kutta solvers, a system of linear equations with $n_x \cdot s$ equations has to be solved in every iteration [123]. Here, s is the number of stages, which is a particular property of a Runge–Kutta solver which determines the order of the method.

For multistep methods, the function G will also depend on previous values of \mathbf{x} . A popular implementation of the multistep method is the implicit linear multistep Backwards Differentiation Formula (BDF) implemented in the CVODES solver [124]. In every iteration i , the BDF solves an equation of the form:

$$h_i \beta_{i,0} f(\xi_i, \mathbf{x}(\xi_i), \boldsymbol{\theta}) + \sum_{j=0}^q \alpha_{i,j} \mathbf{x}(\xi_{i-j}) = 0,$$

where q is the order of the method, and $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are the coefficients that are determined in every iteration. The order q and the step size h_i will determine the local error of the numerical solution [124] and are often chosen adaptively. This implicit equation is typically solved using Newton’s method [123, 125]. For the BDF, the function G depends on $\dot{\mathbf{x}}$ and thus on f . Consequently, the Newton solver computes multiple solutions to linear systems defined by the Jacobian $\nabla_{\mathbf{x}} f(t, \mathbf{x}, \boldsymbol{\theta})$ of the right-hand side of the differential equation at every integration step. As this linear system only has n_x equations, in contrast to the $n_x \cdot s$ equations for single-step methods, the computational cost of solving the linear system increases less strongly with the number of state variables n_x .

The computation time of the BDF method primarily depends on two factors: (1) the evaluation time of the function f and the Jacobian $\nabla_{\mathbf{x}} f(t, \mathbf{x}, \boldsymbol{\theta})$, which usually scales linearly with n_x and (2) the time to solve the linear systems defined by $\nabla_{\mathbf{x}} f(t, \mathbf{x}, \boldsymbol{\theta})$.

The matrix $\nabla_x f(t, \mathbf{x}, \boldsymbol{\theta})$ is typically not symmetric and neither positive nor negative definite. For such unstructured problems, LU decomposition (see Fig. 1c middle), which factorizes $\nabla_x f(t, \mathbf{x}, \boldsymbol{\theta})$ into a lower-triangular matrix \mathbf{L} and an upper-triangular matrix \mathbf{U} , is the method of choice to solve the linear system, as long as \mathbf{L} and \mathbf{U} can be stored in memory. After performing the decomposition, the solution to the linear systems can be computed by matrix multiplication. When no additional structure of the matrix is exploited, the computational complexity of matrix multiplication with the state-of-the-art algorithms increases at least with exponent 2.376 with respect to n_x [126] and thus dominates the computation time for sufficiently large n_x .

2.3.2 Sparse Implicit Methods

For ODE models arising from discretization of partial differential equations, the Jacobian can usually be brought into banded form. For such banded matrices, specialized solvers that scale with the number of off-diagonals of the Jacobian have been developed [127]. Unfortunately, ODE models of biochemical reaction networks cannot generally be brought into a banded structure. For example, in polymerization reactions that include dissociation of monomers, the monomer species will always be influenced by all other species and the number of off-diagonals in the Jacobian will be equal to n_x . Other frequently occurring motifs, such as feedback loops and single highly interactive species [128], will also increase the number of necessary off-diagonals.

As alternative to banded solvers, sparse solvers have been introduced in the context of circuit simulations [129]. For sparse solvers, the computation time depends on the number of nonzero entries in $\nabla_x f(t, \mathbf{x}, \boldsymbol{\theta})$, which scales with the number of biochemical reactions. The sparse solver relies on an approximate minimum degree (AMD) ordering [46] which is a graph theoretical approach that can be used to minimize the fill-in of the \mathbf{L} and \mathbf{U} matrices of the LU-decomposition (see Fig. 1C right). Currently, no formulas for the expected speedup or the general scaling with respect to nonzero entries exist. For biochemical reaction networks, the application of such a sparse solver seems reasonable [119], but no rigorous evaluation of the scaling has been performed.

2.3.3 Implementation and Practical Considerations

Most toolboxes use CVODES [124] or LSODA [130] for simulation (see Table 3). In contrast to CVODES, which only implements an implicit solver, LSODA dynamically switches between explicit and implicit solvers. To the best of our knowledge, no comparison between LSODA and CVODES has ever been published. However, LSODA does not provide an interface to the Clark Kent LU solver (KLU) [129] or any other sparse solver and thus might perform poorly on large-scale problems with sparsity structure. Another notable difference is that only few toolboxes analytically

Table 3
Implementations of simulation methods in popular computational biology toolboxes

Toolbox	Simulation library	Jacobian	Dense solver	Sparse solver
AMICI [75]	CVODES [124]	Symbolic	✓	✓
COPASI [20]	LSODA [130]	Numeric	✓	✗
D2D [21]	CVODES [124]	Symbolic	✓	✓
libRoadRunner [22]	CVODES [124]	Numeric	✓	✗

compute the Jacobian of the right-hand side and provide it to the solver. The symbolic processing is necessary for the sparse representation, but also likely to be beneficial for dense solvers. Thus, D2D [21] and AMICI [75] are also the only general purpose simulation libraries for systems biology that allow the use of the sparse KLU solver.

For explicit solvers, no linear system has to be solved and the algorithm largely consists of elementary operations which can be efficiently parallelized on GPUs [131]. For explicit solvers, also parallelized solvers are available [132]. However, the computational overhead of parallelization is usually too high, unless models with several thousand state variables are considered [132].

All of the considered toolboxes allow the definition of ODE models in the Systems Biology Markup Language (SBML) [133]. This also allows for the definition of models in terms of biochemical reactions. Here, COPASI and libRoadRunner aim for a full support of SBML features, while AMICI and D2D only support a subset of SBML features.

Sparse numerical solvers can be used to efficiently compute the numerical solution to the ODE, which are required for objective function evaluation. They can also be used to compute objective function gradients as solution to one or more ODEs. Several different gradient computation approaches exist and in the following we will discuss the three most common approaches.

2.4 Gradient Calculation

Providing an accurate gradient to the objective function is essential for gradient-based methods [56, 71, 134]. For ODE constrained optimization problems, the gradient of the objective function can be computed based on the parametric derivative of the solution to the ODE. These derivatives are often called the sensitivities of the model. Several approaches to compute sensitivities for ODE models exist, including finite differences [135] as well as the direct approach via forward sensitivity analysis [136, 137].

2.4.1 Finite Differences and Forward Sensitivity Analysis

For finite differences, the entries of the gradient are approximated according to:

$$\frac{dJ}{d\theta_k} \approx \frac{J(\boldsymbol{\theta} + a \mathbf{e}_k) - J(\boldsymbol{\theta} - b \mathbf{e}_k)}{a + b},$$

with $a, b \geq 0$ and the k^{th} unit vector \mathbf{e}_k . In practice, forward differences ($a = \zeta, b = 0$), backward differences ($a = 0, b = \zeta$), and central differences ($a = \zeta, b = \varepsilon$), with $\zeta \ll 1$, are widely used. As the evaluation of $J(\boldsymbol{\theta} + a \mathbf{e}_k)$ and $J(\boldsymbol{\theta} - b \mathbf{e}_k)$ may require additional solutions to the model ODE, the scaling of finite differences with respect to the number of parameters is also linear.

For forward sensitivity analysis, the entries of the gradient of the objective function are computed according to:

$$\frac{dJ}{d\theta_k} = - \sum_{i=1}^{n_y} \sum_{j=1}^T \frac{\partial J}{\partial y_i(t_j, \boldsymbol{\theta})} s_{i,k}^y(t_j, \boldsymbol{\theta}) + \frac{\partial J}{\partial \theta_k},$$

with $s_{i,k}^y(t, \boldsymbol{\theta})$ denoting the sensitivity of output y_i at time point t_j with respect to parameter θ_k . This output sensitivity can be computed by applying the total derivative to the functions h :

$$s_{i,k}^y(t_j, \boldsymbol{\theta}) = \left. \frac{\partial h_i}{\partial x} \right|_{x(t, \boldsymbol{\theta}), \boldsymbol{\theta}} s_k^x(t_j, \boldsymbol{\theta}) + \left. \frac{\partial h_i}{\partial \theta_k} \right|_{x(t, \boldsymbol{\theta}), \boldsymbol{\theta}}$$

with $s_k^x(t, \boldsymbol{\theta})$ denoting the sensitivity of the state x with respect to θ_k . The state sensitivity is defined as solution to the ODE system:

$$\dot{s}_k^x(t, \boldsymbol{\theta}) = \left. \frac{\partial f}{\partial x} \right|_{x(t, \boldsymbol{\theta}), \boldsymbol{\theta}} s_k^x(t, \boldsymbol{\theta}) + \left. \frac{\partial f}{\partial \theta_k} \right|_{x(t, \boldsymbol{\theta}), \boldsymbol{\theta}}, \quad s_k^x(t_0, \boldsymbol{\theta}) = \left. \frac{\partial x_0}{\partial \theta_k} \right|_{\boldsymbol{\theta}}.$$

Thus, forward sensitivity analysis requires the computation of a solution to an ODE system of the same size as the model ODE for every gradient entry. Consequently, the scaling with respect to the number of parameters is linear (see Fig. 1b right).

2.4.2 Adjoint Sensitivity Analysis

The linear scaling of forward sensitivity analysis and finite differences can be computationally prohibitively demanding for large-scale models with thousands of parameters. The alternative adjoint approach, which computes the objective function gradient via adjoint sensitivity analysis, has long been deemed to be computationally more efficient for systems with many parameters [137]. In other research fields, e.g., for partial differential equation constrained optimization problems, adjoint sensitivity analysis [125] has been adopted in the past decades. In contrast, in the systems biology community there are only isolated applications of adjoint sensitivity analysis [138–140].

In the mathematics and engineering community, adjoint sensitivity analysis is frequently used to compute the gradients of a functional with respect to the parameters if the functional depends on the solution of a differential equation [141]. In these

applications, measurements are continuous in time and $J(\theta)$ is assumed to be a functional of the solution $\mathbf{x}(t)$ of a differential equation. However, this approach can also be applied to discrete-time measurements and in contrast to forward sensitivity analysis, adjoint sensitivity analysis does not rely on the state sensitivities $\mathbf{s}_k^x(t)$, but on the adjoint state $\mathbf{p}(t)$.

For discrete-time measurements—the usual case in systems and computational biology, the adjoint state is piecewise continuous in time and defined by a sequence of backward differential equations [75]. For $t > t_N$, the adjoint state is zero, $\mathbf{p}(t) = 0$. Starting from this end value, the trajectory of the adjoint state is calculated backwards in time, from the last measurement $t = t_N$ to the initial time $t = t_0$. At the measurement time points t_N, \dots, t_1 , the adjoint state is reinitialized as:

$$\mathbf{p}(t_j) = \lim_{t \rightarrow t_j^+} \mathbf{p}(t) + \frac{\partial J}{\partial x}, \quad (18)$$

which usually results in a discontinuity of $\mathbf{p}(t)$ at t_j . Starting from the end value $\mathbf{p}(t_j)$ as defined in (18), the adjoint state evolves backwards in time until the next measurement point t_{j-1} or the initial time t_0 is reached. This evolution is governed by the time dependent linear ODE:

$$\dot{\mathbf{p}} = - \left(\frac{\partial f}{\partial x} \right)^T \mathbf{p}. \quad (19)$$

The repeated evaluation of (18) and (19) until $t = t_0$ yields the trajectory of the adjoint state. Given this trajectory, the gradient of the objective function with respect to the individual parameters is

$$\frac{dJ}{d\theta_k} = - \int_{t_0}^{t_N} \mathbf{p}^T \frac{\partial f}{\partial \theta_k} dt - \mathbf{p}(t_0)^T \frac{\partial \mathbf{x}_0}{\partial \theta_k} + \frac{\partial J}{\partial \theta_k}. \quad (20)$$

The key advantage of this approach is that (20), which has to be evaluated for every parameter separately, can be evaluated very efficiently, while (19), which is computationally more demanding, only has to be solved once [75]. As (19) is of the same dimensionality as the original ODE model, this allows the computation of gradients at the cost of roughly two solutions of the original ODE model. In practice, the adjoint sensitivity approach has an almost constant scaling with respect to the number of parameters.

2.4.3 Implementation and Practical Considerations

Many toolboxes rely on finite differences to compute gradients (see Table 4). D2D [21] and AMICI [75] are two notable examples that allow the computation of gradients via sensitivity analysis, but only AMICI allows adjoint sensitivity analysis.

Table 4

Implementations of gradient computation methods in popular computational biology tool-boxes

Toolbox	Finite differences	Forward sensitivity	Adjoint sensitivity
AMICI [75]	✗	✓	✓
COPASI [20]	✓	✗	✗
D2D [21]	✓	✓	✗
libRoadRunner [22]	✓	✗	✗

2.5 Hessian Computation

In addition to the gradient, Newton-type methods also require the Hessian $\nabla^2 J(\boldsymbol{\theta})$ of the objective function. The numerical evaluation of the Hessian can be challenging as the dependence of the computational complexity on the number of parameters n_θ is one order higher than for the gradient: The computation time for finite differences and forward sensitivities scales quadratically with the number of parameters [142, 143]. For adjoint sensitivities, the computation time depends linearly on the number of parameters [144].

2.5.1 Gauss–Newton Approximation

For independent, normally distributed measurement noise, as assumed in (5), and known noise parameters σ , the optimization problem (6) is of least-squares type. This structure can be exploited by using Gauss–Newton (GN)-type [145] algorithms, which ignore second-order partial derivatives in the Hessian. The respective approximations of the Hessian coincide with the Fisher information matrix (FIM) [146] of the respective parameter estimate. The key advantage of this approach is that the FIM can be computed for the same cost as one gradient using forward sensitivity analysis.

2.5.2 Quasi-Newton Approximation

For problems that are not of least-squares type, quasi-Newton methods such as the Broyden–Fletcher–Goldfarb–Shanno (BFGS) [147, 148] algorithm can be used. The BFGS algorithm iteratively computes approximations to the Hessian based on updates which are derived from the outer products of the gradient $\nabla J(\boldsymbol{\theta})$ of the objective function. The resulting approximation is guaranteed to be positive definite, as long as the Wolfe condition [78] is satisfied in every iteration and the initial approximation is positive definite. As previously discussed, positive definiteness ensures descent directions for line-search methods, but will generally lead to small step sizes in the vicinity of saddle points. The symmetric rank 1 (SR1) algorithm addresses this problem by allowing for negative- and indefinite approximations [149]. This procedure facilitates the application of optimization methods

Table 5
Implementations of (approximative) Hessian computation methods in popular computational biology toolboxes

Toolbox	FIM / GN	Hessian	BFGS	SR1
AMICI [75]	SE	SE	N/A	N/A
COPASI [20]	FD	×	Truncated Newton	×
D2D [21]	SE/FD	×	fmincon	arNLS_SR1
libRoadRunner [22]	×	×	N/A	N/A
MEIGO [20]	N/A	N/A	fmincon, IpOpt	IpOpt
PESTO [93]	N/A	N/A	fmincon	×

For BFGS and SR1, we list the function or option that allows the respective approximation. SE = sensitivity equations, FD = finite differences, GN = Gauss–Newton, N/A = Not applicable

which avoid saddle points by allowing directions of negative curvature [150].

Quasi-Newton versions are generally cheap to compute, as they only require simple algebraic manipulations of the gradient. Algorithms based on limited memory variants such as L-BFGS [151, 152] or L-SR1 [150] have been applied to machine learning problems with millions of parameters [153].

2.5.3 Implementation and Practical Considerations

The implementation of methods for the computation of (approximate) Hessians is quite disparate across toolboxes (*see* Table 5). AMICI is the only toolbox that allows sensitivity-based computation of the Hessian. Most other toolboxes use FIM/Gauss–Newton, BFGS, or SR1 approximations. Most of the iterative approximations BFGS and SR1 are implemented as part of the optimization algorithm and it is not possible to use them with other methods. Only D2D provides a relatively flexible implementation of SR1. For the FIM approximation and the exact Hessian computations, the implementations are usually transferable between optimization methods. In theory, the computation of the exact Hessian, with adjoint sensitivity analysis, and the FIM, with forward sensitivity analysis, both scale linearly with the number of parameters and only the exact Hessian can be used to construct methods that avoid saddle points [45]. In practice, the effect of using the FIM over the Hessian on the efficiency of respective optimization methods has not been studied for systems biology problems. For problems with thousands of parameters, the computation of both may be challenging and the BFGS and SR1 approximations become more appealing.

In this section, we split the parameter inference problem into three parts: optimization, simulation, and gradient computation

and discussed respective scaling properties. These techniques generalize to other model analysis techniques that require optimization or gradient computation, such as uncertainty analysis [154, 155], experimental design [156], and the inference of model structure. A detailed discussion of all these methods is beyond the scope of this book chapter, but in the following we will discuss the inference of model structure in more detail.

3 Inference of Model Structure

In many applications, it is not apparent which biochemical species and reactions are necessary to describe the dynamics of a biochemical process. In this case, the structure of the ODE model (1), i.e., vector field $f(\mathbf{x}, \boldsymbol{\theta})$ and initial condition $\mathbf{x}_0(\boldsymbol{\theta})$, has to be inferred from the experimental data. The selection should compromise between goodness-of-fit and complexity. Following the concept of Occam's razor [157], one tries to control variability associated with over-fitting while protecting against the bias associated with underfitting.

In the following, we formulate the problem of model structure inference. We introduce and discuss criteria that select models out of a set of candidate models and describe approaches to reduce the number of candidate models. We outline the scalability of the approaches and their computational complexity.

3.1 Model Selection Criteria

Given a set of candidate models M_1, M_2, \dots, M_{n_M} , the aim of model inference is to find a model or a set of models which: (i) describe the data available and (ii) generalize to other datasets [158]. The choice of model can be made with several selection criteria, differing among others in asymptotic consistency [159], asymptotic efficiency [146], and computational complexity. If the true model is included in the set of candidate models, a consistent criterion will asymptotically select the true model with probability one and an efficient criterion will select the model that minimizes the mean squared error of the prediction.

While the concepts in the previous sections followed the frequentist approach, some of the concepts presented in this section are Bayesian. For these approaches, prior knowledge about the parameters is incorporated and the posterior probability (7) is analyzed instead of the likelihood function.

One popular criterion is the Bayes factor [160], which has been shown to be asymptotically consistent for a broad range of models (e.g., [161, 162]); however, for the case of general ODE models, no proofs for asymptotic efficiency and consistency are available for all the criteria presented in this section. Bayes' theorem yields the posterior model probability:

$$p(M_m|\mathcal{D}) = \frac{p(\mathcal{D}|M_m)p(M_m)}{p(\mathcal{D})} \quad (21)$$

with marginal likelihood:

$$p(\mathcal{D}|M_m) = \int_{\Theta_m} p(\mathcal{D}|\boldsymbol{\theta}_m)p(\boldsymbol{\theta}_m|M_m)d\boldsymbol{\theta}_m \quad (22)$$

with model prior $p(M_m)$ and marginal probability $p(\mathcal{D}) = \sum_j p(\mathcal{D}|M_j)P(M_j)$. The Bayes factor of models M_1 and M_2 is the ratio of the corresponding marginal likelihoods:

$$B_{12} = \frac{p(\mathcal{D}|M_1)}{p(\mathcal{D}|M_2)}. \quad (23)$$

The Bayes factor describes how much more likely it is that the data are generated from M_1 instead of M_2 . A Bayes factor $B_{12} > 100$ is often considered decisive for rejecting model M_2 [163]. The Bayes factor intrinsically penalizes model complexity by integrating over the whole parameter space of each model. Bayes factors can be approximated by Laplace approximation, which has a low computational complexity but provides only a local approximation. To enable a more precise computation of the Bayes factors, bridge sampling [164], nested sampling [165], thermodynamic integration [7], or related methods can be employed. These approaches evaluate the integral defining the marginal likelihood $p(\mathcal{D}|M_m)$. As the approaches require a large number of function evaluations, the methods are usually computationally demanding and the computational complexity is highly problem-dependent. Thus, efficient sampling methods are required.

For high-dimensional or computationally demanding problems, the calculation of Bayes factors might be intractable and computationally less expensive model selection criteria need to be employed. A model selection criterion which is based on the MLE, instead of a marginal likelihood (an integral over the whole parameter space), is the Bayesian Information Criterion (BIC) [31]. The BIC value for model M_m is

$$\text{BIC}_m = -2 \log(p(\mathcal{D}|\boldsymbol{\theta}_m^*)) + \log(|\mathcal{D}|) n_{\theta_m}. \quad (24)$$

For structural identifiable models, the BIC provides in the limit of large sample sizes information about the Bayes factors,

$$\lim_{|\mathcal{D}| \rightarrow \infty} \frac{-2 \log B_{12} - (\text{BIC}_1 - \text{BIC}_2)}{-2 \log B_{12}} = 0. \quad (25)$$

From information theoretical arguments, the Akaike Information Criterion (AIC):

Table 6
Decisions based on the Bayes factor and differences in BIC and AIC values [160, 163, 166]

B_{ml}	$\text{BIC}_m - \min_l \text{BIC}_l$	$\text{AIC}_m - \min_l \text{AIC}_l$	Decision
1 – 3	0 – 2	0 – 4	Do not reject model M_m
3 – 100	2 – 10	4 – 10	–
> 100	> 10	> 10	Reject model M_m

$$\text{AIC}_m = -2 \log(p(\mathcal{D}|\boldsymbol{\theta}_m^*)) + 2n_{\theta_m}, \quad (26)$$

has been derived [30]. Low BIC and AIC values are preferable and differences above 10 are assumed to be substantial (*see* Table 6 and [160, 166]).

For model selection in many problem classes, the AIC is asymptotically efficient, but not consistent, while the BIC is asymptotically consistent, but not efficient [167–169].

When incorporating prior information about parameters, the priors can conceptually be treated as additional data points and, thus, be part of the likelihood to still allow the use of BIC and AIC. Also, extensions of the criteria exist, such as the corrected AIC [170], which provides a correction for finite sample sizes. Also, other extended versions of the criteria have been developed (*see*, e.g., [171]); however, the discussion of these is beyond the scope of this chapter.

For the comparison of nested models M_m and M_l , i.e., $\boldsymbol{\theta}_m \in \Theta_m$ and $\boldsymbol{\theta}_l \in \Theta_l$ where Θ_m is a subset of Θ_l , the likelihood ratio test can be applied [172], which is an efficient test [173]. The likelihood ratio is defined as:

$$\Lambda = \frac{p(\mathcal{D}|\boldsymbol{\theta}_m^*)}{p(\mathcal{D}|\boldsymbol{\theta}_l^*)} \leq 1, \quad (27)$$

and model M_m is rejected if Λ is below a certain threshold which is obtained using Wilks' theorem [172]. This theorem states that it holds in the large sample limit (*see* [172] for further details)

$$2(\log p(\mathcal{D}|\boldsymbol{\theta}_l^*) - \log p(\mathcal{D}|\boldsymbol{\theta}_m^*)) \sim \chi^2(\cdot | n_{\theta_l} - n_{\theta_m}). \quad (28)$$

Given a certain α level, model M_m is rejected if

$$\int_0^{2(\log p(\mathcal{D}|\boldsymbol{\theta}_l^*) - \log p(\mathcal{D}|\boldsymbol{\theta}_m^*))} \chi^2(\psi | n_{\theta_l} - n_{\theta_m}) d\psi \geq 1 - \alpha. \quad (29)$$

While only Bayes factors and the likelihood ratio test are proven to be valid for non-identifiable parameters, the use of AIC and BIC can be problematic for these cases.

The discussion of further criteria, such as the log-pointwise predictive density [174] or cross-validation (*see, e.g.*, [175]), which evaluate the predictive quality of the model, is beyond the scope of this chapter.

3.1.1 Implementation and Practical Considerations

AIC and BIC are rather simple to compute and are, among others, available in PESTO [93]. From the previously discussed toolboxes of this review, PESTO also provides sampling methods that can be employed to calculate Bayes factors, such as parallel tempering. Other toolboxes which can be employed for computing Bayes factors are, among others, BioBayes [176], MultiNest [177], or the C++ toolbox BCM [178].

3.2 Reduction of Number of Models

For most models, computing Bayes' factors is computationally demanding compared to optimization and the evaluation of AIC, BIC, or likelihood ratio. Yet, if the number of candidate models n_M is large, even the evaluation of AIC and BIC can become limiting as n_M optimization problems have to be solved. For non-nested models, the model selection criterion of choice needs to be calculated for each model to determine the optimal model.

In this section, we consider a nested set of candidate models. In this case, all candidate models are a special case of a comprehensive model and can be constructed by fixing a subset of the parameters to specific values (Fig. 2a). For the remainder of this chapter, we will assume that we can split the model parameters θ into general parameters $\eta \in \mathbb{R}^{n_\eta}$, which are present in all models, and difference parameters $r \in \mathbb{R}^{n_r}$, which encode the nesting between models. Moreover, without loss of generality, it is assumed that $r_i = 0, i = 1, \dots, n_r$, corresponds to the simplest model and $r_i \neq 0, i = 1, \dots, n_r$, corresponds to the most complex model (*see* Fig. 2a). These difference parameters could, for example, be the kinetic rates of hypothesized reactions [33] or scaling factors for possibly cell-type or condition-specific parameters (*see, e.g.*, [32]). Such settings yield a total of 2^{n_r} candidate models, where n_r is limited by n_θ . Thus, for models with a high number of parameters, also a high number of nested models are possible. When n_r and n_θ are both high, the inference of model parameters and thus the inference of model structure is challenging.

3.2.1 Forward-Selection and Backward-Elimination

In statistics, stepwise regression is an often-used approach to reduce the number of models that need to be tested. This comprises forward-selection and backward-elimination (*e.g.*, [158]) and combinations of both [179]. Forward-selection is a bottom-up approach which starts with the least complex model and successively activates individual difference parameters (*i.e.*, setting $r_i \neq 0$) until a sufficient agreement with experimental data is achieved, evaluated using a model selection criterion (Fig. 2b). In contrast, backward-elimination is a top-down approach starting

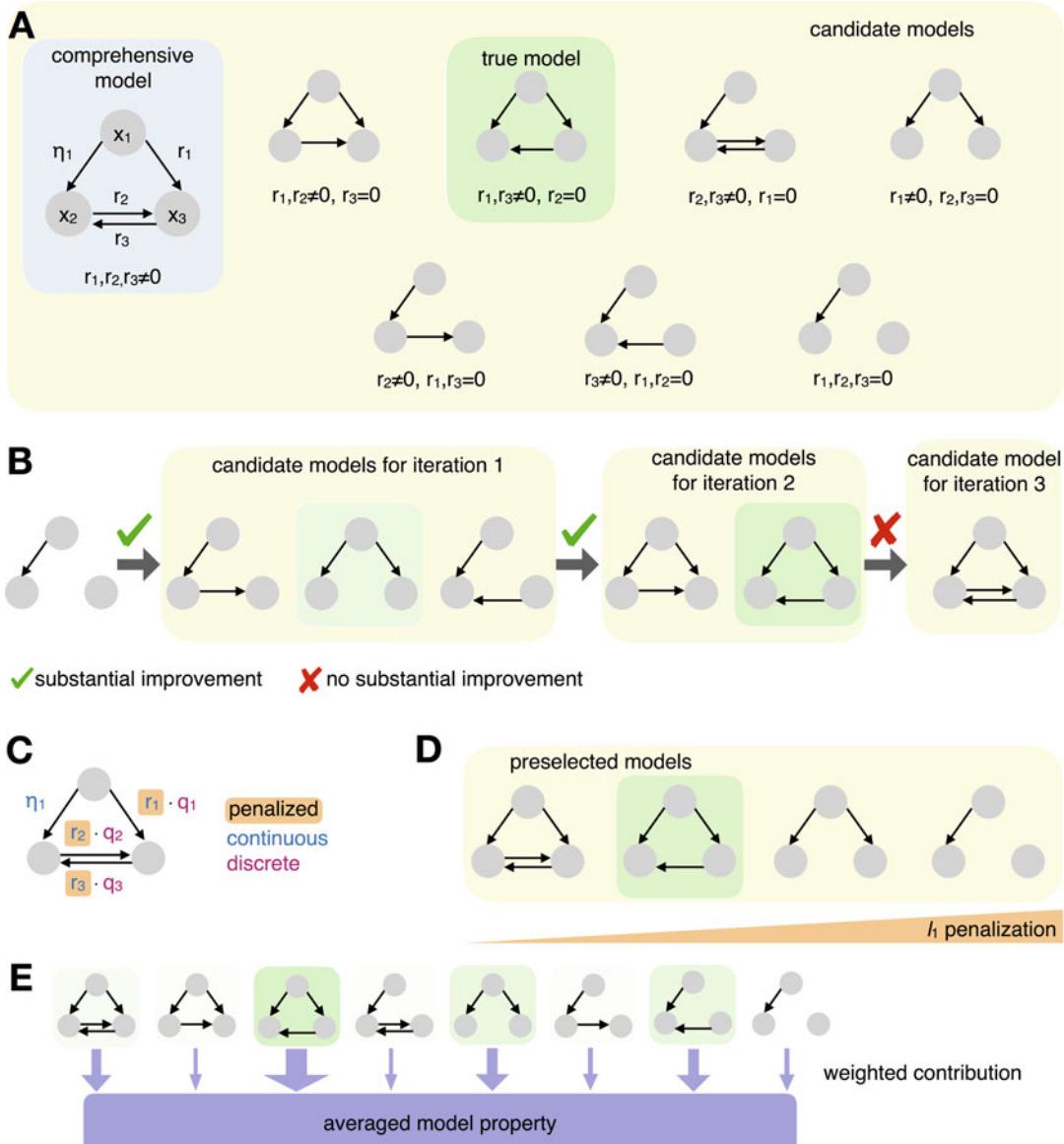


Fig. 2 Illustration of methods for model reduction. **(a)** Set of candidate models, varying in the existence of connections between nodes x_1, x_2 , and x_3 . In total, there are $2^{n_r} = 2^3$ models with at least $n_\eta = 1$ parameters. **(b)** Illustration of forward-selection starting from minimal model. In the first iteration, the model with $r_1 \neq 0, r_2, r_3 = 0$ is selected (green) and in the second iteration the model with $r_1, r_3 \neq 0, r_2 = 0$. The full model is rejected based on the selection criteria. **(c)** To apply l_0 penalization, an MINLP problem needs to be solved, comprising continuous parameters η and \mathbf{r} and discrete parameters $\mathbf{q} \in \{0, 1\}^{n_r}$. **(d)** l_1 penalization reduces the number of potential models to a set of preselected models by increasing the penalization and thus forcing parameters r to zero. **(E)** Illustration of model averaging. The thickness of the arrows corresponds to the posterior probability, Akaike weight, or BIC weight and indicates the contribution of the model to the averaged model properties

with the most complex model, successively deactivating individual difference parameters (i.e., setting $r_i = 0$) that are not required for a good fit to the data.

Forward-selection and backward-elimination reduce the number of models that need to be compared with the model selection criteria described before from 2^{n_r} to at most $\frac{n_r(n_r+1)}{2}$. However, they are both greedy approaches and do not guarantee to find the globally least complex candidate model that explains the data.

3.2.2 l_0 Penalized Objective Function

An alternative approach is to penalize the number of parameters in the objective function. This can be achieved by imposing an l_0 penalization on the objective function (see, e.g., [180]):

$$J_{l_0}(\boldsymbol{\eta}, \mathbf{r}) = J(\boldsymbol{\eta}, \mathbf{r}) + \lambda \left(n_\eta + \sum_{i=1}^{n_r} (1 - \delta_{r_i}) \right), \quad \text{with } \delta_z = \begin{cases} 1 & \text{if } z = 0 \\ 0 & \text{otherwise} \end{cases} \quad (30)$$

This l_0 penalization supports sparsity, i.e., reduces the model such that only a minimum number of difference parameters are used. As all models contain at least n_η parameters, only n_r contributes to changes in the complexity. For $\lambda = 1$, the objective function J_{l_0} is the AIC divided by two. For $\lambda = \frac{1}{2} \log(|\mathcal{D}|)$, the objective function J_{l_0} is the BIC divided by two. Accordingly, minimization of J_{l_0} can provide the best model according to different information criteria. To directly assess the predictive power, λ can also be determined using cross-validation.

Following [181], the objective function (30) allows for the formulation of structure inference as a mixed-integer nonlinear programming (MINLP) problem:

$$J_{MINLP}(\boldsymbol{\eta}, \mathbf{r}, \mathbf{q}) = J(\boldsymbol{\eta}, \text{diag}(\mathbf{rq}^T)) + \lambda \sum_{i=1}^{n_r} q_i, \quad (31)$$

with real-valued $\boldsymbol{\eta} \in \mathbb{R}^{n_\eta}$, $\mathbf{r} \in \mathbb{R}^{n_r}$, and integer-valued $\mathbf{q} \in \{0, 1\}^{n_r}$. The optimization is done simultaneously for all parameters, $\boldsymbol{\eta}, \mathbf{r}, \mathbf{q}$. This objective function is neither differentiable nor continuous with respect to \mathbf{q} . Thus, gradients with respect to the discrete parameters will not be informative for optimization. This limits the choice of optimization algorithms to derivative-free and specialized gradient-based solvers, such as the MISQP algorithm [182]. Besides the above-described and commonly used methods, further approaches, among others belief propagation [10] or iteratively reweighted least squares [183], can be employed, under certain model assumptions.

For the MINLP (31) resulting from the l_0 penalized objective functions, only the comprehensive model is estimated. This, however, results in a more complex optimization problem which suffers from a high-dimensional parameter space.

3.2.3 l_1 Penalized Objective Function

To simplify the optimization problem, the l_0 norm is often replaced by its convex relaxation, the l_1 norm [184]. This yields the penalized objective function:

$$J_{l_1}(\boldsymbol{\eta}, \mathbf{r}) = J(\boldsymbol{\eta}, \mathbf{r}) + \lambda \sum_{i=1}^{n_r} |r_i|, \quad (32)$$

with $r_i \in \mathbb{R}$, which is forced to be zero for higher λ corresponding to the situations where the parameter θ_i has no effect. In linear regression, l_1 penalization is more commonly known as Lasso [184], while in signal processing it is usually referred to as Basis Pursuit [185]. The l_1 norm is continuous, but not differentiable in zero. Thus, specialized solvers have been developed which handle the non-differentiability at zero [32].

For the special case of linear regression models, $J_{l_0}(\boldsymbol{\eta}, \mathbf{r})$ and $J(\boldsymbol{\eta}, \mathbf{r})$ are convex. As the l_1 norm is a convex relaxation of the l_0 norm, the resulting objective function is also convex. Thus, it can be shown that the estimated parameters r_i are unique and continuous with respect to λ [186]. Moreover, r_i can be shown to be piecewise linear with respect to λ , which allows an implementation that can efficiently compute solutions for all values of λ [186]. For ODE models, $J_{l_1}(\boldsymbol{\eta}, \mathbf{r})$ and $J(\boldsymbol{\eta}, \mathbf{r})$ will generally be nonconvex and r_i may be nonunique and discontinuous which is challenging for numerical methods. Thus, equation (32) is usually minimized for varying penalization strengths λ , until a reduced set of model candidates is selected. As the l_1 norm is an approximation of the l_0 norm, model selection should subsequently be performed on the reduced set of model candidates using the criteria introduced in Subheading 3.1 (Fig. 2d).

The computational complexity of the l_1 penalization depends on the number of different penalization strengths that are used. With a higher number of tested penalizations, it is more likely to obtain the globally optimal model, while the lower number of tested penalizations decreases the computational effort.

3.2.4 Implementation and Practical Considerations

Only few toolboxes implement the methods that allow simultaneous inference of model parameters and structure. MEIGO implements the MISQP algorithm, while D2D implements the l_1 penalization via a modification of the fmincon routine [32].

3.3 Model Averaging

For large sets of candidate models and limited data, it frequently happens that not a single model is chosen by the model selection criterion. Instead, a set of models is plausible, cannot be rejected, and should be considered in the subsequent analysis. In this case, model averaging can be employed to predict the behavior of the process (Fig. 2e).

Given that a certain parameter is comparable between models, an average estimate can be derived as:

$$\mathbb{E}[\theta_j] = \sum_m w_m \theta_{m,j} \quad (33)$$

with w_m denoting the weight of model M_m and $\theta_{m,j}$ denoting the MLE of the parameter for model M_m [166, 187]. Accordingly, uncertainties can be evaluated, e.g., the variance of the optimal values:

$$\text{var}[\theta_j] = \frac{1}{n_M - 1} \sum_m (w_m \theta_{m,j} - \mathbb{E}[\theta_j])^2. \quad (34)$$

The weights capture the plausibility of the model. An obvious choice is the posterior probability $p(M_m|\mathcal{D})$. Alternatively, the Akaike weights:

$$w_m = \frac{\exp(-\frac{1}{2}\text{AIC}_m)}{\sum_{i=1}^{n_M} \exp(-\frac{1}{2}\text{AIC}_i)} \quad (35)$$

or the BIC weights:

$$w_m = \frac{\exp(-\frac{1}{2}\text{BIC}_m)}{\sum_{i=1}^{n_M} \exp(-\frac{1}{2}\text{BIC}_i)}. \quad (36)$$

can be employed. The weights for models that are not plausible are close to zero and, thus, these models do not influence the averaged model.

4 Discussion

This chapter provided an overview about the methods for parameter inference and structure inference for ODE models of biochemical systems. For parameter inference, we discussed local optimization methods and identified the number of stationary points as key determinant of computational complexity. In the context of local optimization, we identified gradient-based optimization methods as suitable method as the computational complexity of determining the parameter update for optimization from the gradient of the objective function is *per se* independent of the number of state variables and number of model parameters. Still, numerical optimization requires the computation of the ODE solution, which scales with the number of molecular species, and the computation of respective derivatives, which scales with the number of parameters. In both cases, we discussed scaling properties of the state-of-art algorithms and identified adjoint

sensitivity analysis and sparse solvers as most suitable methods for large-scale problems.

We believe that key challenges to improve the scalability of parameter inference lie in the treatment of stationary points of the objective function, such as local minima and saddle points. In contrast to deep learning problems [45], the dependence of the number of stationary points on the underlying (ODE) model remains poorly understood [188, 189] and should be evaluated. Local optimization methods that can account for saddle points and local minima have been developed [45, 51, 65], but lack implementations in computational biology toolboxes and evaluations on ODE models of biochemical systems.

For structure inference, large-scale models often also give rise to a large set of different model candidates. Many model comparison criteria require parameter inference for all model candidates, which is rarely feasible if the number of model candidates is high. We discussed an l_1 penalization-based approach that allows the simultaneous inference of model parameters and structure. We believe that key challenges to improve the scalability of structure inference lie in the treatment of non-differentiability of the l_1 norm, which prohibits the application of standard gradient-based optimization algorithms. Methods such as iteratively reweighted least-squares were developed decades ago [190] but were not adopted for ODE models.

We anticipate that, with the advent of whole cell models [27, 191, 192] and other large-scale models [193, 194], the demand for scalable methods will drastically increase in the coming years. However, already for medium-scale models, which are much more commonplace, parameter inference and in particular structure inference can be challenging. Accordingly, there is a growing demand for novel methods with better scaling properties.

References

- Klipp E, Herwig R, Kowald A, Wierling C, Lehrach H (2005) Systems biology in practice. Wiley-VCH, Weinheim
- Kitano H (2002) Systems biology: a brief overview. *Science* 295(5560):1662–1664
- Kitano H (2002) Computational systems biology. *Nature* 420(6912):206–210
- Adlung L, Kar S, Wagner MC, She B, Chakraborty S, Bao J, Lattermann S, Boerries M, Busch H, Wuchter P, Ho AD, Timmer J, Schilling M, Höfer T, Klingmüller U (2017) Protein abundance of AKT and ERK pathway components governs cell type-specific regulation of proliferation. *Mol Syst Biol* 13(1):904
- Buchholz VR, Flossdorf M, Hensel I, Kretschmer L, Weissbrich B, Gräf P, Ver schoor A, Schiemann M, Höfer T, Busch DH (2013) Disparate individual fates compose robust CD8+ t cell immunity. *Science* 340(6132):630–635
- Intosalmi J, Nousiainen K, Ahlfors H, Lähdesmäki H (2016) Data-driven mechanistic analysis method to reveal dynamically evolving regulatory networks. *Bioinformatics* 32(12):i288–i296
- Hug S, Schwarzfischer M, Hasenauer J, Marr C, Theis FJ (2016) An adaptive scheduling scheme for calculating Bayes factors with thermodynamic integration using Simpson’s rule. *Stat Comput* 26(3):663–677
- Hross S, Fiedler A, Theis FJ, Hasenauer J (2016) Quantitative comparison of compet-

- ing PDE models for Pom1p dynamics in fission yeast. In: Findeisen R, Bullinger E, Balsa-Canto E, Bernaerts K (eds) Proc. 6th IFAC conf. found. syst. biol. eng., IFAC-PapersOnLine, vol 49, pp 264–269
9. Toni T, Ozaki Yi, Kirk P, Kuroda S, Stumpf MPH (2012) Elucidating the *in vivo* phosphorylation dynamics of the ERK MAP kinase using quantitative proteomics data and Bayesian model selection. *Mol Biosyst* 8:1921–1929
10. Molinelli EJ, Korkut A, Wang W, Miller ML, Gauthier NP, Jing X, Kaushik P, He Q, Mills G, Solit DB, Pratilas CA, Weigt M, Braunstein A, Pagnani A, Zecchina R, Sander C (2013) Perturbation biology: inferring signaling networks in cellular systems. *PLoS Comput Biol* 9(12):e1003290
11. Schilling M, Maiwald T, Hengl S, Winter D, Kreutz C, Kolch W, Lehmann WD, Timmer J, Klingmüller U (2009) Theoretical and experimental analysis links isoform-specific ERK signalling to cell fate decisions. *Mol Syst Biol* 5:334
12. Fey D, Halasz M, Dreidax D, Kennedy SP, Hastings JF, Rauch N, Munoz AG, Pilkington R, Fischer M, Westermann F, Kolch W, Khodenko BN, Croucher DR (2015) Signaling pathway models as biomarkers: patient-specific simulations of JNK activity predict the survival of neuroblastoma patients. *Sci Signal* 8(408):ra130
13. Eduati F, Doldà-Martelli V, Klinger B, Cokealaer T, Sieber A, Kogera F, Dorel M, Garnett MJ, Blüthgen N, Saez-Rodriguez J (2017) Drug resistance mechanisms in colorectal cancer dissected with cell Type-Specific dynamic logic models. *Cancer Res* 77(12):3364–3375
14. Hass H, Masson K, Wohlgemuth S, Paragas V, Allen JE, Sevecka M, Pace E, Timmer J, Stelling J, MacBeath G, Schoeberl B, Raue A (2017) Predicting ligand-dependent tumors from multi-dimensional signaling features. *NPJ Syst Biol Appl* 3(1):27
15. Maiwald T, Hass H, Steiert B, Vanlier J, Engesser R, Raue A, Kipkeew F, Bock HH, Kaschek D, Kreutz C, Timmer J (2016) Driving the model to its limit: Profile likelihood based model reduction. *PLoS One* 11(9):e0162366
16. Snowden TJ, van der Graaf PH, Tindall MJ (2017) Methods of model reduction for large-scale biological systems: a survey of current methods and trends. *B Math Biol* 79(7):1449–1486
17. Transtrum MK, Qiu P (2016) Bridging mechanistic and phenomenological models of complex biological systems. *PLoS Comput Biol* 12(5):1–34
18. Dano S, Madsen MF, Schmidt H, Cedersund G (2006) Reduction of a biochemical model with preservation of its basic dynamic properties. *FEBS J* 273(21):4862–4877
19. Klonowski W (1983) Simplifying principles for chemical and enzyme reaction kinetics. *Biophys Chem* 18(2):73–87
20. Hoops S, Sahle S, Gauges R, Lee C, Pahle J, Simus N, Singhal M, Xu L, Mendes P, Kummer U (2006) COPASI – a COmplex PAthway SImluator. *Bioinformatics* 22(24):3067–3074
21. Raue A, Steiert B, Schelker M, Kreutz C, Maiwald T, Hass H, Vanlier J, Tönsing C, Adlung L, Engesser R, Mader W, Heinemann T, Hasenauer J, Schilling M, Höfer T, Klipp E, Theis FJ, Klingmüller U, Schöberl B, JTimmer (2015) Data2Dynamics: a modeling environment tailored to parameter estimation in dynamical systems. *Bioinformatics* 31(21):3558–3560
22. Somogyi ET, Bouteiller JM, Glazier JA, König M, Medley JK, Swat MH, Sauro HM (2015) libRoadRunner: a high performance SBML simulation and analysis library. *Bioinformatics* 31(20):3315–3321
23. Santos SDM, Verveer PJ, Bastiaens PIH (2007) Growth factor-induced MAPK network topology shapes Erk response determining PC-12 cell fate. *Nat Cell Biol* 9(3):324–330
24. Yao J, Pilko A, Wollman R (2016) Distinct cellular states determine calcium signaling response. *Mol Syst Biol* 12(12):894
25. Ogilvie LA, Kovachev A, Wierling C, Lange BMH, Lehrach H (2017) Models of models: a translational route for cancer treatment and drug development. *Front Oncol* 7: 219
26. Schillings C, Sunnåker M, Stelling J, Schwab C (2015) Efficient characterization of parametric uncertainty of complex (bio)chemical networks. *PLoS Comput Biol* 11(8):e1004457
27. Babtie AC, Stumpf MPH (2017) How to deal with parameters for whole-cell modelling. *J R Soc Interface* 14(133):20130505

28. Ocone A, Haghverdi L, Mueller NS, Theis FJ (2015) Reconstructing gene regulatory dynamics from high-dimensional single-cell snapshot data. *Bioinformatics* 31(12):i89–i96
29. Kondofersky I, Fuchs C, Theis FJ (2015) Identifying latent dynamic components in biological systems. *IET Syst Biol* 9(5):193–203
30. Akaike H (1973) Information theory and an extension of the maximum likelihood principle. In: 2nd international symposium on information theory, Tsahkadsor, Armenian SSR, Akademiai Kiado, vol 1, pp 267–281
31. Schwarz G (1978) Estimating the dimension of a model. *Ann Stat* 6(2):461–464
32. Steiert B, Timmer J, Kreutz C (2016) L1 regularization facilitates detection of cell type-specific parameters in dynamical systems. *Bioinformatics* 32(17):i718–i726
33. Klimovskaia A, Ganscha S, Claassen M (2016) Sparse regression based structure learning of stochastic reaction networks from single cell snapshot time series. *PLoS Comput Biol* 12(12):e1005234
34. Loos C, Moeller K, Fröhlich F, Hucho T, Hasenauer J (2018) A hierarchical, data-driven approach to modeling single-cell populations predicts latent causes of cell-to-cell variability. *Cell Syst* 6(5):593–603
35. Hock S, Hasenauer J, Theis FJ (2013) Modeling of 2D diffusion processes based on microscopy data: Parameter estimation and practical identifiability analysis. *BMC Bioinf* 14(Suppl 10):S7
36. Hross S (2016) Parameter estimation and uncertainty quantification for image based systems biology. Ph.D. thesis, Technische Universität München
37. Menshykau D, Germann P, Lemereux L, Iber D (2013) Simulating organogenesis in COMSOL: Parameter optimization for PDE-based models. In: Proceedings of COMSOL conference, Rotterdam
38. Hross S, Hasenauer J (2016) Analysis of CFSE time-series data using division-, age- and label-structured population models. *Bioinformatics* 32(15):2321–2329
39. Fröhlich F, Thomas P, Kazeroonian A, Theis FJ, Grima R, Hasenauer J (2016) Inference for stochastic chemical kinetics using moment equations and system size expansion. *PLoS Comput Biol* 12(7):e1005030
40. Ruess J, Lygeros J (2015) Moment-based methods for parameter inference and experiment design for stochastic biochemical reaction networks. *ACM T Math Softwares Model Comput S* 25(2):8
41. Munsky B, Trinh B, Khammash M (2009) Listening to the noise: random fluctuations reveal gene network parameters. *Mol Syst Biol* 5:318
42. Kreutz C, Bartolome Rodriguez MM, Maiwald T, Seidl M, Blum HE, Mohr L, Timmer J (2007) An error model for protein quantification. *Bioinformatics* 23(20):2747–2753
43. Maier C, Loos C, Hasenauer J (2017) Robust parameter estimation for dynamical systems from outlier-corrupted data. *Bioinformatics* 33(5):718–725
44. Puga JL, Krzywinski M, Altman N (2015) Bayes' theorem. *Nat Methods* 12(3):277–278
45. Dauphin YN, Pascanu R, Gulcehre C, Cho K (2014) Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In: Advances in neural information processing systems 26 (NIPS 2014), pp 2933–2941
46. Amestoy PR, Davis TA, Duff IS (1996) An approximate minimum degree ordering algorithm. *SIAM J Matrix Anal A* 17(4):886–905
47. Anandkumar A, Ge R (2016) Efficient approaches for escaping higher order saddle points in non-convex optimization. In: Conference on learning theory, pp 81–102
48. Kirk P, Rolando DM, MacLean AL, Stumpf MP (2015) Conditional random matrix ensembles and the stability of dynamical systems. *New J Phys* 17(8):083025
49. Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. *IEEE Trans Evol Comput* 1(1):67–82
50. Goffe WL, Ferrier GD, Rogers J (1994) Global optimization of statistical functions with simulated annealing. *J Econom* 60(1–2):65–99
51. Neumaier A (2004) Complete search in continuous global optimization and constraint satisfaction. *Acta Numer* 13:271–369
52. Johnson DS, McGeoch LA (1997) The traveling salesman problem: a case study in local optimization. In: Local search in combinatorial optimization, vol 1, pp 215–310
53. Ashyraliyev M, Fomekong-Nanfack Y, Kaandorp JA, Blom JG (2009) Systems biology: parameter estimation for biochemical models. *FEBS J* 276(4):886–902

54. Hooke R, Jeeves TA (1961) “Direct Search” solution of numerical and statistical problems. *J ACM* 8(2):212–229
55. Fister Jr I, Yang XS, Fister I, Brest J, Fister D (2013) A brief review of nature-inspired algorithms for optimization. *Elektrotehniski Vestnik* 80(3):116–122
56. Nocedal J, Wright S (2006) Numerical optimization. Springer Science & Business Media, Berlin/Heidelberg
57. Nelder JA, Mead R (1965) A simplex method for function minimization. *Comput J* 7(4):308–313
58. De La Maza M, Yuret D (1994) Dynamic hill climbing. *AI expert* 9:26–26
59. Levenberg K (1944) A method for the solution of certain non-linear problems in least squares. *Q Appl Math* 2(2):164–168
60. Marquardt DW (1963) An algorithm for least-squares estimation of non-linear parameters. *SIAM J Appl Math* 11(22):431–441
61. Holland JH (1992) Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. MIT Press, Cambridge
62. Kennedy J (2011) Particle swarm optimization. In: Encyclopedia of machine learning. Springer, Boston, pp 760–766
63. Egea JA, Rodriguez-Fernandez M, Banga JR, Marti R (2007) Scatter search for chemical and bio-process optimization. *J Global Optim* 37(3):481–503
64. Kirkpatrick S, Gelatt~ CD Jr, Vecchi MP (1983) Optimization by simulated annealing. *Science* 220(4598):671–680
65. Kan AR, Timmer GT (1987) Stochastic global optimization methods part I: clustering methods. *Math Program* 39(1):27–56
66. Lawler EL, Wood DE (1966) Branch-and-bound methods: A survey. *Oper Res* 14(4):699–719
67. Törn A, Zilinskas A (1989) Global optimization. Lecture notes in computer science, vol 350. Springer-Verlag, Berlin
68. Rios LM, Sahinidis NV (2013) Derivative-free optimization: a review of algorithms and comparison of software implementations. *J Global Optim* 56(3):1247–1293
69. Rudolph G (1994) Convergence analysis of canonical genetic algorithms. *IEEE Trans Neural Netw* 5(1):96–101
70. Moles CG, Mendes P, Banga JR (2003) Parameter estimation in biochemical pathways: a comparison of global optimization methods. *Genome Res* 13:2467–2474
71. Raue A, Schilling M, Bachmann J, Matteson A, Schelke M, Kaschek D, Hug S, Kreutz C, Harms BD, Theis FJ, Klingmüller U, Timmer J (2013) Lessons learned from quantitative dynamical modeling in systems biology. *PLoS One* 8(9):e74335
72. Banga JR (2008) Optimization in computational systems biology. *BMC Syst Biol* 2:47
73. Boender CGE, Rinnooy Kan AHG (1987) Bayesian stopping rules for multistart global optimization methods. *Math Program* 37(1):59–80
74. Törn A, Ali MM, Viitanen S (1999) Stochastic global optimization: problem classes and solution techniques. *J Global Optim* 14(4):437–447
75. Fröhlich F, Kaltenbacher B, Theis FJ, Hase-nauer J (2017) Scalable parameter estimation for genome-scale biochemical reaction networks. *PLoS Comput Biol* 13(1):e1005331
76. Penas DR, González P, Egea JA, Banga JR, Doallo R (2015) Parallel metaheuristics in computational biology: an asynchronous cooperative enhanced scatter search method. *Procedia Comput Sci* 51:630–639
77. Armijo L (1966) Minimization of functions having Lipschitz continuous first partial derivatives. *Pac J Math* 16(1):1–3
78. Wolfe P (1969) Convergence conditions for ascent methods. *SIAM Rev* 11(2):226–235
79. Kolda TG, Lewis RM, Torczon V (2003) Optimization by direct search: new perspectives on some classical and modern methods. *SIAM Rev* 45(3):385–482
80. Lewis RM, Torczon V, Trosset MW (2000) Direct search methods: then and now. *J Comput Appl Math* 124(1):191–207
81. Rosenbrock HH (1960) An automatic method for finding the greatest or least value of a function. *Comput J* 3(3):175–184
82. Yuan Yx (2015) Recent advances in trust region algorithms. *Math Program* 151(1):249–281
83. Hartley HO (1961) The modified Gauss–Newton method for the fitting of non-linear regression functions by least squares. *Technometrics* 3(2):269–280
84. Nesterov Y, Polyak B (2006) Cubic regularization of newton method and its global performance. *Math Program* 108(1):177–205
85. Lanczos C (1950) An iteration method for the solution of the eigenvalue problem

- of linear differential and integral operators. United States Governm. Press Office, Los Angeles
86. Nash SG (1984) Newton-type minimization via the Lanczos method. *SIAM J Numer Anal* 21(4):770–788
 87. Byrd RH, Schnabel RB, Shultz GA (1987) A trust region algorithm for nonlinearly constrained optimization. *SIAM J Numer Anal* 24(5):1152–1170
 88. Sorensen DC (1982) Newton's method with a model trust region modification. *SIAM J Numer Anal* 19(2):409–426
 89. Wild SM, Regis RG, Shoemaker CA (2008) ORBIT: Optimization by radial basis function interpolation in trust-regions. *SIAM J Sci Comput* 30(6):3197–3219
 90. Steihaug T (1983) The conjugate gradient method and trust regions in large scale optimization. *SIAM J Numer Anal* 20(3):626–637
 91. Byrd RH, Schnabel RB, Shultz GA (1988) Approximate solution of the trust region problem by minimization over two-dimensional subspaces. *Math Program* 40(1):247–263
 92. Branch MA, Coleman TF, Li Y (1999) A subspace, interior, and conjugate gradient method for large-scale bound-constrained minimization problems. *SIAM J Sci Comput* 21(1):1–23
 93. Stapor P, Weindl D, Ballnus B, Hug S, Loos C, Fiedler A, Krause S, Hross S, Fröhlich F, Hasenauer J (2017) PESTO: Parameter EStimation TOOlbox. Bioinformatics btx676
 94. Egea JA, Henriques D, Cokelaer T, Villaverde AF, MacNamara A, Danciu DP, Banga JR, Saez-Rodriguez J (2014) MEIGO: an open-source software suite based on metaheuristics for global optimization in systems biology and bioinformatics. *BMC Bioinf* 15:136
 95. Fogel DB, Fogel LJ, Atmar JW (1991) Meta-evolutionary programming. In: 1991 Conference record of the twenty-fifth Asilomar conference on signals, systems and computers, 1991, pp 540–545. IEEE, New York
 96. Michalewicz Z (2013) Genetic Algorithms + Data Structures = Evolution Programs. Springer Science & Business Media, Berlin/Heidelberg
 97. Brent RP (2013) Algorithms for minimization without derivatives. Courier Corporation, North Chelmsford
 98. Dembo RS, Steihaug T (1983) Truncated-newtono algorithms for large-scale unconstrained optimization. *Math Program* 26(2):190–212
 99. Solis FJ, Wets RJB (1981) Minimization by random search techniques. *Math Oper Res* 6(1):19–30
 100. Runarsson TP, Yao X (2000) Stochastic ranking for constrained evolutionary optimization. *IEEE Trans Evolut Comput* 4(3): 284–294
 101. Bellavia S, Macconi M, Morini B (2004) STRSCNE: a scaled trust-region solver for constrained nonlinear equations. *Comput Optim Appl* 28(1):31–50
 102. Morini B, Porcelli M (2012) TRESNEI, a Matlab trust-region solver for systems of nonlinear equalities and inequalities. *Comput Optim Appl* 51(1):27–49
 103. Le Digabel S (2011) Algorithm 909: NOMAD: nonlinear optimization with the MADS algorithm. *ACM Trans Math Software* 37(4):44
 104. Kelley CT (1999) Iterative methods for optimization. SIAM, Philadelphia
 105. Wächter A, Biegler LT (2006) On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Math Program* 106(1):25–57
 106. Ye Y (1989) SOLNP users' guide. Tech. rep., Department of Management Sciences, University of Iowa
 107. Exler O, Lehmann T, Schittkowski K (2012) MISQP: a Fortran subroutine of a trust region SQP algorithm for mixed-integer nonlinear programming-user's guide. Tech. rep., Department of Computer Science, University of Bayreuth
 108. Dennis JE Jr, Gay DM, Welsch RE (1981) Algorithm 573: NL2sol—an adaptive nonlinear least-squares algorithm [E4]. *ACM Trans Math Software* 7(3):369–383
 109. Powell MJ (2009) The BOBYQA algorithm for bound constrained optimization without derivatives. Tech. rep., Cambridge NA Report NA2009/06, University of Cambridge, Cambridge
 110. Vaz AIF, Vicente LN (2009) PSwarm: a hybrid solver for linearly constrained global derivative-free optimization. *Optim Method Softw* 24(4–5):669–685
 111. Degasperis A, Fey D, Kholodenko BN (2017) Performance of objective functions and optimisation procedures for parameter estimation

- in system biology models. *NPJ Syst Biol Appl* 3(1):20
112. Wieland FG (2016) Implementation and assessment of optimization approaches for parameter estimation in systems biology. Tech. rep., University of Freiburg
113. Villaverde AF, Henriques D, Smallbone K, Bongard S, Schmid J, Cicin-Sain D, Crombach A, Saez-Rodriguez J, Mauch K, Balsa-Canto E, Mendes P, Jaeger J, Banga JR (2015) BioPreDyn-bench: a suite of benchmark problems for dynamic modelling in systems biology. *BMC Syst Biol* 9(8)
114. Kreutz C (2016) New concepts for evaluating the performance of computational methods. *IFAC-PapersOnLine* 49(26):63–70
115. Shamir M, Bar-On Y, Phillips R, Milo R (2016) SnapShot: timescales in cell biology. *Cell* 164(6):1302–1302
116. Hasenauer J, Jagiella N, Hross S, Theis FJ (2015) Data-driven modelling of biological multi-scale processes. *J Coupled Syst Multi-scale Dynam* 3(2):101–121
117. Smallbone K, Mendes P (2013) Large-scale metabolic models: from reconstruction to differential equations. *Ind Biotechnol* 9(4):179–184
118. Resat H, Petzold L, Pettigrew MF (2009) Kinetic modeling of biological systems. *Methods Mol Biol* 541:311–335
119. Gonnet P, Dimopoulos S, Widmer L, Stelling J (2012) A specialized ODE integrator for the efficient computation of parameter sensitivities. *BMC Syst Biol* 6:46
120. Butcher JC (1964) Implicit Runge-Kutta processes. *Math Comp* 18(85):50–64
121. Alexander R (1977) Diagonally implicit Runge-Kutta methods for stiff O.D.E.'s. *SIAM J Numer Anal* 14(6):1006–1021
122. Rosenbrock HH (1963) Some general implicit processes for the numerical solution of differential equations. *Comput J* 5(4):329–330
123. Zhang H, Sandu A (2014) FATODE: a library for forward, adjoint, and tangent linear integration of ODEs. *SIAM J Sci Comput* 36(5):C504–C523
124. Serban R, Hindmarsh AC (2005) CVODES: an ODE solver with sensitivity analysis capabilities. *ACM Trans Math Software* 31(3):363–396
125. Hindmarsh AC, Brown PN, Grant KE, Lee SL, Serban R, Shumaker DE, Woodward CS (2005) SUNDIALS: Suite of Nonlinear and Differential/Algebraic Equation Solvers. *ACM Trans Math Software* 31(3):363–396
126. Coppersmith D, Winograd S (1990) Matrix multiplication via arithmetic progressions. *J Symb Comp* 9(3):251–280
127. Thorson J (1979) Gaussian elimination on a banded matrix
128. Barabasi AL, Oltvai ZN (2004) Network biology: understanding the cell's functional organization. *Nat Rev Genet* 5(2):101–113
129. Davis TA, Palamadai Natarajan E (2010) Algorithm 907: KLU, a direct sparse solver for circuit simulation problems. *ACM Trans Math Software* 37(3):36
130. Petzold L (1983) Automatic selection of methods for solving stiff and nonstiff systems of ordinary differential equations. *SIAM J Sci Stat Comp* 4(1):136–148
131. Tangherloni A, Nobile MS, Besozzi D, Mauri G, Cazzaniga P (2017) LASSIE: simulating large-scale models of biochemical systems on GPUs. *BMC Bioinformatics* 18(1):246
132. Demmel JW, Gilbert JR, Li XS (1999) An asynchronous parallel supernodal algorithm for sparse Gaussian elimination. *SIAM J Matrix Anal A* 20(4):915–952
133. Hucka M, Finney A, Sauro HM, Bolouri H, Doyle JC, Kitano H, Arkin AP, Bornstein BJ, Bray D, Cornish-Bowden A, Cuellar AA, Dronov S, Gilles ED, Ginkel M, Gor V, Goryanin II, Hedley WJ, Hodgman TC, Hofmeyr JH, Hunter PJ, Juty NS, Kasberger JL, Kremling A, Kummer U, Le-Novère N, Loew LM, Lucio D, Mendes P, Minch E, Mjolsness ED, Nakayama Y, Nelson MR, Nielsen PF, Sakurada T, Schaff JC, Shapiro BE, Shimizu TS, Spence HD, Stelling J, Takahashi K, Tomita M, Wagner J, Wang J (2003) The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics* 19(4):524–531
134. Griewank A, Walther A (2008) Evaluating derivatives, 2nd edn. Society for Industrial and Applied Mathematics, Philadelphia
135. Milne-Thompson L (1933) The calculus of finite differences. Macmillan, London
136. Dickinson RP, Gelinas RJ (1976) Sensitivity analysis of ordinary differential equation systems—a direct method. *J Comput Phys* 21(2):123–143
137. Kokotovic P, Heller J (1967) Direct and adjoint sensitivity equations for parameter optimization. *IEEE Trans Autom Contr* 12(5):609–610

138. Lu J, Muller S, Machné R, Flamm C (2008) SBML ODE solver library: extensions for inverse analysis. In: Proc. 5th int. W. comp. syst. biol.
139. Fujarewicz K, Kimmel M, Swierniak A (2005) On fitting of mathematical models of cell signaling pathways using adjoint systems. *Math Bio Eng* 2(3):527–534
140. Lu J, August E, Koeppl H (2012) Inverse problems from biomedicine: inference of putative disease mechanisms and robust therapeutic strategies. *J Math Biol* 67(1):143–168
141. Plessix RE (2006) A review of the adjoint-state method for computing the gradient of a functional with geophysical applications. *Geophys J Int* 167(2):495–503
142. Balsa-Canto E, Banga JR, Alonso AA, Vassiliadis VS (2001) Dynamic optimization of chemical and biochemical processes using restricted second-order information. *Comput Chem Eng* 25(4):539–546
143. Vassiliadis VS, Canto EB, Banga JR (1999) Second-order sensitivities of general dynamic systems with application to optimal control problems. *Chem Eng Sci* 54(17):3851–3860
144. Özuyurt DB, Barton PI (2005) Cheap second order directional derivatives of stiff ODE embedded functionals. *SIAM J Sci Comput* 26(5):1725–1743
145. Björck Å (1996) Numerical methods for least squares problems. SIAM, Philadelphia
146. Fisher RA (1922) On the mathematical foundations of theoretical statistics. *Philos Trans R Soc London, Ser A* 222:309–368
147. Fletcher R, Powell MJ (1963) A rapidly convergent descent method for minimization. *Comp J* 6(2):163–168
148. Goldfarb D (1970) A family of variable-metric methods derived by variational means. *Math Comp* 24(109):23–26
149. Byrd RH, Khalfan HF, Schnabel RB (1996) Analysis of a symmetric rank-one trust region method. *SIAM J Optim* 6(4):1025–1039
150. Ramamurthy V, Duffy N (2017) L-SRI: A second order optimization method for deep learning, under review as a conference paper at ICLR 2017
151. Nocedal J (1980) Updating quasi-newton matrices with limited storage. *Mathematics of computation* 35(151):773–782
152. Liu DC, Nocedal J (1989) On the limited memory BFGS method for large scale optimization. *Math Program* 45(1):503–528
153. Andrew G, Gao J (2007) Scalable training of l1-regularized log-linear models. In: Proceedings of the 24th international conference on Machine learning - ICML '07. ACM, Philadelphia, pp 33–40
154. Raue A, Kreutz C, Maiwald T, Bachmann J, Schilling M, Klingmüller U, Timmer J (2009) Structural and practical identifiability analysis of partially observed dynamical models by exploiting the profile likelihood. *Bioinformatics* 25(25):1923–1929
155. Girolami M, Calderhead B (2011) Riemann manifold Langevin and Hamiltonian Monte Carlo methods. *J R Statist Soc B* 73(2):123–214
156. Vanlier J, Tiemann CA, Hilbers PAJ, van Riel NAW (2012) A Bayesian approach to targeted experiment design. *Bioinformatics* 28(8):1136–1142
157. Blumer A, Ehrenfeucht A, Haussler D, Warmuth MK (1987) Occam's razor. *Inform Process Lett* 24(6):377–380
158. Hastie T, Tibshirani R, Friedman J (2009) The elements of statistical learning, vol 2. Springer, New York
159. Shibata R (1980) Asymptotically efficient selection of the order of the model for estimating parameters of a linear process. *Ann Stat* 8(1):147–164
160. Kass RE, Raftery AE (1995) Bayes factors. *J Am Stat Assoc* 90(430):773–795
161. Wang M, Sun X (2014) Bayes factor consistency for nested linear models with a growing number of parameters. *J Stat Plan Infer* 147:95–105
162. Choi T, Rousseau J (2015) A note on Bayes factor consistency in partial linear models. *J Stat Plan Infer* 166:158–170
163. Jeffreys H (1961) Theory of probability, 3rd edn. Oxford University Press, Oxford
164. Meng XL, Wong WH (1996) Simulating ratios of normalizing constants via a simple identity: a theoretical exploration. *Stat Sin* 6(4):831–860
165. Skilling J (2006) Nested sampling for general Bayesian computation. *Bayesian Anal* 1(4):833–359

166. Burnham KP, Anderson DR (2002) Model selection and multimodel inference: a practical information-theoretic approach, 2nd edn. Springer, New York, NY
167. Shibata R (1981) An optimal selection of regression variables. *Biometrika* 68(1):45–54
168. Kuha J (2004) AIC and BIC: comparisons of assumptions and performance. *Sociol Method Res* 33(2):188–229
169. Acquah HDG (2010) Comparison of Akaike information criterion (AIC) and Bayesian information criterion (BIC) in selection of an asymmetric price relationship. *J Dev Agric Econ* 2(1):001–006
170. Hurvich C, Tsia CL (1989) Regression and time series model selection in small samples. *Biometrika* 76(2):297–307
171. Chen J, Chen Z (2008) Extended Bayesian information criteria for model selection with large model spaces. *Biometrika* 95(3):759–771
172. Wilks SS (1938) The large-sample distribution of the likelihood ratio for testing composite hypotheses. *Ann Math Stat* 9(1):60–62
173. Neyman J, Pearson ES (1992) On the problem of the most efficient tests of statistical hypotheses. In: Breakthroughs in statistics. Springer, New York, pp 73–108
174. Gelman A, Hwang J, Vehtari A (2014) Understanding predictive information criteria for Bayesian models. *Stat Comp* 24(6):997–1016
175. Arlot S, Celisse A (2010) A survey of cross-validation procedures for model selection. *Stat Surv* 4:40–79
176. Vyshemirsky V, Girolami M (2008) BioBayes: a software package for Bayesian inference in systems biology. *Bioinformatics* 24(17):1933–1934
177. Feroz F, Hobson M, Bridges M (2009) Multi-nest: an efficient and robust Bayesian inference tool for cosmology and particle physics. *Mon Not R Astron Soc* 398(4):1601–1614
178. Thijssen B, Dijkstra TM, Heskes T, Wessels LF (2016) BCM: toolkit for Bayesian analysis of computational models using samplers. *BMC Syst Biol* 10(1):100
179. Kaltenbacher B, Offtermatt J (2011) A refinement and coarsening indicator algorithm for finding sparse solutions of inverse problems. *Inverse Probl Imaging* 5(2):391–406
180. Liu Z, Li G (2016) Efficient regularized regression with L0 penalty for variable selection and network construction. *Comput Math Methods Med*, 3456153
181. Rodriguez-Fernandez M, Rehberg M, Klemmung A, Banga JR (2013) Simultaneous model discrimination and parameter estimation in dynamic models of cellular systems. *BMC Syst Biol* 7(1):76
182. Henriques DR, M Saez-Rodriguez J, Banga JR (2015) Reverse engineering of logic-based differential equation models using a mixed-integer dynamic optimization approach. *Bioinformatics* 31(18):2999–3007
183. Daubechies I, DeVore R, Fornasier M, Güntürk CS (2010) Iteratively reweighted least squares minimization for sparse recovery. *Commun Pur Appl Math* 63(1):1–38
184. Tibshirani R (1996) Regression shrinkage and selection via the lasso. *J R Statist Soc B* 58(1):267–288
185. Chen SS, Donoho DL, Saunders MA (2001) Atomic decomposition by basis pursuit. *SIAM Rev* 43(1):129–159
186. Efron B, Hastie T, Johnstone I, Tibshirani R (2004) Least angle regression. *Ann Stat* 32(2):407–499
187. Wassermann L (2000) Bayesian model selection and model averaging. *J Math Psychol* 44(1):92–107
188. Mannakee BK, Ragsdale AP, Transtrum MK, Gutenkunst RN (2016) Sloppiness and the geometry of parameter space. Springer International Publishing, Cham, pp 271–299
189. Transtrum MK, Machta BB, Sethna JP (2011) Geometry of nonlinear least squares with applications to sloppy models and optimization. *Phys Rev E* 83:036701
190. Holland PW, Welsch RE (1977) Robust regression using iteratively reweighted least-squares. *Commun Stat Theory* 6(9): 813–827
191. Karr JR, Sanghvi JC, Macklin DN, Gutschow MV, Jacobs JM, Bolival~Jr B, Assad-Garcia N, Glass JI, Covert MW (2012) A whole-cell computational model predicts phenotype from genotype. *Cell* 150(2): 389–401
192. Tomita M, Hashimoto K, Takahashi K, Shimizu TS, Matsuzaki Y, Miyoshi F, Saito K, Tanida S, Yugi K, Venter JC, Hutchison~III CA (1999) E-CELL: software environment for whole-cell simulation. *Bioinformatics* 15(1):72–84
193. Fröhlich F, Kessler T, Weindl D, Shadrin A, Schmiester L, Hache H, Muradyan A, Schuette M, Lim JH, Heinig M, Theis F,

- Lehrach H, Wierling C, Lange B, Hase-nauer J (2017) Efficient parameterization of large-scale mechanistic models enables drug response prediction for cancer cell lines. *bioRxiv*, 174094
194. Büchel F, Rodriguez N, Swainston N, Wrzodek C, Czauderna T, Keller R, Mittag F, Schubert M, Glont M, Golebiewski M, van-Iersel M, Keating S, Rall M, Wybrow M, Hermjakob H, Hucka M, Kell DB, Müller W, Mendes P, Zell A, Chaouiya C, Saez-Rodriguez J, Schreiber F, Laibe C, Dräger A, Novére NL (2013) Path2Models: Large-scale generation of computational models from biochemical pathway maps. *BMC Syst Biol* 7(116)

INDEX

A

- Affinity mass-spectrometry, 165
Akaike Information Criterion (AIC), 386
Analysis of variance (ANOVA), 80–82
Arabidopsis thaliana, 76, 90
 Bio-PEPA Eclipse Plug-in framework, 77
 circadian clock, 252
 clock gene interactions, 89
 DIURNAL database, 87
 gene expression samples, 87
 iCheMA method, 87–89
 and network modifications, 78
 P2013 and F2014 networks, 87–89
Area under the curve (AUROC) score, 80–82
Area under the precision recall curve (AUPREC)
 score, 80–82
Automatic relevance determination (ARD) method, 58

B

- Bagging algorithm, 200
Bayesian analysis, 254
Bayesian Best Subset Regression (BBSR)
 approach, 175–176
Bayesian Dirichlet Equivalent uniform (BDE), 122
Bayesian Information Criterion (BIC), 65, 386
 score, 122
Bayesian model score, 306
Bayesian networks, 11–13, 27, 115, 118, 169
 causal interpretation via node intervention, 118–119
 causal Markov condition, 124
 causal sufficiency assumption, 124
 definition, 120–121
D-maps, 120
d-separation, 119–120
experimental setup
 conditional probabilities, 125, 126
 data generation, 126
 four-node network, 129–131
 generalized additive model, 127, 128
 graph edit distance, 126, 135
 Kullback–Leibler (KL) divergence, 126–127, 135
 network reconstruction, 126
 7-node BN, 125, 127–129

- real biological and regulatory gene
 network, 130–133
 results, 127–133
faithfulness condition, 124
graphs and directed acyclic graphs, 119
I-maps, 120
intervention, 123–124
Markov blanket of node, 120, 121
Markov equivalence, 124–125
object-oriented, 121
parameter learning, 122–123
perfect maps, 120
prior-based framework for, 172–174
probabilistic independence, 120
vs. structural intervention distance, 127
structure learning
 constraint-based algorithms, 122
 hybrid algorithms, 122
 overview, 121–122
 score-based algorithms, 122
see also Dynamic Bayesian networks (DBNs)
Bayesian nonparametric models, 255
Bayesian spline autoregression (BSA) method, 59–60
 MATLAB programs, 74
Bayes' rule, 255
Biological system
 inner organization, 112
 networks, 113
 vs. statistical representation, 348
Bottom-up approaches, 251
Brute-force approach, 199
Bulk expression data, 236

C

- Causal biological networks, see Causality
Causal gene networks, 95
 Findr program, 106
 computing environment, 98–99
 description, 98
 genome–transcriptome variation data, 96
 question formalization, 96–98
see also Whole-transcriptome causal gene network
 inference

- Causality, 116–117
 in gene network reconstruction, 117–118
- Causal Markov condition, 124
- Causal models, 39–41
- Causal structure identification, graphical model, 264–265
- Causal sufficiency assumption, 124
- Changepoint process, in DBN model, 34–37
 D. melanogaster, 41–44
- Chromatin Immuno-Precipitation followed by sequencing (ChIP-seq), 5
- Circadian regulatory network prediction, in *A. thaliana*,
 see *Arabidopsis thaliana*
- Conditional independence test, 96
- Consensus network algorithms, 288
- Constraint-based algorithms, 122
- Correlation networks, 7–8
- D**
- Data-driven methods
 correlation networks, 7–8
 information theoretic scores, 8–9
 regression-based method, 9–10
- DBNs, *see* Dynamic Bayesian networks (DBNs)
- Decision trees, advantages, 195
- Dependency networks, 169
 prior-based framework for, 174–176
- Dialogue for Reverse Engineering of Models (DREAM), 17
- Differential equation methods, 15–16
- Directed acyclic graphs (DAGs), 12, 118, 119, 169
- DNA methylation, 352
- DREAM5 data, 291
 results on, 295–296
- Drosophila melanogaster*, 292–293
 non-homogeneous DBN methods, 41
 segmentation network, 276–278
 TESLA, 41–43
 time-varying DBN, 41–43
 wing muscle development network, 44
- Dynamical models
 differential equation methods, 15–16
 dynamic Bayesian networks, 14–15
- Dynamic Bayesian networks (DBNs), 14–15
 causal model, 40
 discrete time index, 27
 feedforward, 27, 29, 30
 graphical representations of, 28
 inference, 29–30
 multiple non-identical networks,
 estimation of, 31–32
 and ordinary differential equations, 30–31
- dynGENIE3, 207–208

E

- Elastic net method, 55
 software implementation, 73
- Ensemble methods, 200
- Epigenetic modifications, 3
- Escherichia coli* (*E. coli*), 292
 gene expression, regulation of, 3
 transcriptional response of, 3
- Expression-based network inference, 167–168
 description, 165–166
 differential equations-based methods, 166
 information theoretic methods, 166
 per-gene methods, 166
 per-module methods, 166
 probabilistic graphical models
 components, 166
 dependency networks, 169
 directed acyclic graphs (Bayesian networks), 169
 from gene expression data, 170–171
 parameter prior approach, 172
 structure prior approach, 172
 undirected graphical models, 169
- Extra-Trees method, 200

F

- Findr program, 106
 computing environment, 98–99
 description, 98
- Fuzzy decision trees, 211

G

- Gaussian Bayesian networks (BGe), 66–67, 75
- Gaussian graphical models (GGM), 10–11, 52–54, 115
 application areas, 143–144
 description, 143
 horizontal integration, 144
 multiattribute networks
 gene/protein regulatory network
 inference, 156–158
 need for, 152–153
 neighborhood selection approach, 153–155
 penalizer selection, 155
 separate variant method, 156
 simulation study, 155–156
- proteomic and transcriptomic data integration,
 in cancer, 144–145
- software implementation, 73
- sparse inference
 graphical-Lasso algorithm, 148–149
 graph of conditional dependency structure, 146
 high-dimensional settings, 147
 irrepresentability conditions, 149

- ℓ_1 -regularization, 147, 148
maximum likelihood estimator, 146–147
model selection issues, 151–152
neighborhood selection approach, 149–151
Wishart distribution, 147
- Gaussian process, 61–64
GP4GRN software package, 75
Jump3 framework, 219
- Gaussian process dynamical systems (GPDS)
in Bayesian setting, 257, 258
causal structure, 255
identification, graphical model, 264–265
characteristics, 256
dynamical system, 255
expression profiles, 260
hierarchical
context-dependent networks, 267
hyperparent, 270
network topology, 265–266
repressilator system, 269
- hyperparameters, 263–264
inference in spatiotemporal systems
in *Drosophila melanogaster*, 276
Drosophila segmentation network, 276–278
embryogenesis, 274
modified repressilator network, 275
one-dimensional representation, 274
- inference in unknown systems, 261–263
orthologous
applications, 272
joint inference (JI), 271
network leveraging (NL), 271
Weisfeiler-Lehman (WL) kernel, 273
- prior distribution over functions, 256
simulation with, 265
- Gene expression, 1
control, at transcriptional and post-translational
level, 162
data, 271
- GeneNetWeaver (GNW) generator, 326
- Gene/protein regulatory network inference, 156–158
- Gene regulation, 162–164
biological *vs.* statistical representation, 348
DNA methylation, 352
genetic variation role, 356–357
long noncoding RNAs (lncRNAs), 357–359
modelling gene expression
assigning values, model parameters, 375–376
experimental noise in *in silico* data, 376–377
mathematical frameworks and regulation
functions, 365–374
model simplifications, 374–375
- topological properties, regulatory
networks, 359–365
- mRNA decay regulation, 354
post-translational regulation, 350
protein, 349
activity regulation, 354–355
coding gene, 351
decay regulation, 355–356
riboswitch, 354
transcription factors, combinatorial regulation, 350
- Gene regulatory network (GRN) inference, 2
affinity mass-spectrometry, 165
Boolean networks, 164
chromatin immunoprecipitation based methods, 165
data-driven methods, 7–10
differential equation based models, 164
dynamical models, 14–16
evaluation, 17–18
experimental techniques, 164–165
expression-based network inference, 165–171
genetic perturbation assays, 164–165
mathematical models for, 164
multi-network models, 16–17
precision-recall (PR) curves, 18
prior-based framework, 171–176
probabilistic graphical models, 164
probabilistic models, 10–13
receiver operating characteristic (ROC) curves, 18
reverse engineering, 5–7
software tools, 19–20
two-hybrid assays, 165
- Genetic variation role, 356–357
- GENIE3, 10, 196
area under the precision-recall curve, 205–207
computational complexity, 205
context-specific network inference, 210–211
dynGENIE3, 207–208
feature selection problem, 203
GENIE3-SG-joint, 209
GENIE3-SG-sep, 209
goal of, 202
iRafNet method, 210
limitation of, 211
ODE-based approach, 209
output normalization, 204–205
parameter analysis, 205–207
procedure illustration, 203, 204
running times of, 205, 206
scalability of, 212
SCENIC framework, 209
single-cell data analysis, 209
software availability, 205

- GO annotation task, 312–313
 Graphical-Lasso algorithm, 11, 148–149

H

- Hamming–Ipsen–Mikhailov (HIM) distance, 325
 Hessian computation, 404–406
 Hierarchical Bayesian regression (HBR) models, 56–58
 software implementation, 74
 High-throughput measurements techniques, 4–5
 Hybrid algorithms, 122
 Hybrid approach, 218

I

- Improved Chemical Model Averaging (iCheMA) method
 vs. CheMA method, 86
 MCMC sampling scheme for, 71, 72
 Michaelis–Menten parameters, 67, 68
 posterior inference, 68–71
 probabilistic graphical model representation, 68, 69
 regulator–regulatee interactions, posterior
 probability of, 71
 software implementation, 75
- Inferelator algorithm, 16, 175–176
 Information theoretic scores, 8–9
 Integer linear programming (ILP) approach, 183–185
 Inter Associative Markov Blanket (IAMB) algorithm, .. 126
 Inverse probability, 254
 Ipsen–Mikhailov (IM) distance, 328
 iRafNet method, 210
 IRMA network, 17

J

- Jaccard's similarity index, 158
 Joint Random Forest (JRF), 210–211
 Jump3 framework
 code
 modelling results, 231–232
 predictions, 230–231
 run jump3, 227–230
 see also Run jump3
 computational complexity, 226–227
 nonparametric part, 220–226
 decision trees, latent output variable, 221–223
 ensemble, decision trees, 223–224
 importance measure, 224–226
 kinetic parameters, 226
 parametric part, 219–220
 performance, 227
 perturbation, 218–219
 stochastic differential equation (SDE), 219–220

K

- Kullback–Leibler (KL) divergence, 126–127, 135

L

- Lasso method, *see* Least Absolute Shrinkage and Selection Operator (Lasso) method
 Least Absolute Shrinkage and Selection Operator
 (Lasso) method, 54–55
 software implementation, 73
 LeMoNe algorithm, 211
 Lemon-Tree software package, 308–309
 Likelihood ratio tests, 102, 104
 Linear regression approach, 9–10
 Long noncoding RNAs (lncRNAs), 357–359
 L1 regularization, 11

M

- Machine learning
 description, 196
 generalization error, 197
 learning sample, 197
 training error, 197
- Mammalian NF κ B signalling, 252
 Markov blanket, of node, 120, 121
 Markov chain Monte Carlo (MCMC) methods, 30
 Markov equivalent networks, 12, 124–125
 Maximum likelihood estimator (MLE), 146–147
 Mendelian randomization, 96
 MERLIN-P method, 174–175
 Messenger RNA (mRNA), 349
 Metropolis–Hastings sampler, 30
 Microarray technology, 4
 Mixture Bayesian network (MBN) model, 64–65
 EM algorithm, 75
- Modelling gene expression
 assigning values, model parameters, 375–376
 experimental noise in *in silico* data, 376–377
 mathematical frameworks and regulation
 functions, 365–374
 model simplifications, 374–375
 topological properties, regulatory networks, 359–365
- Modern biological data sets, 114
 Modified Elastic Net (MEN) regression approach, 175
 Module networks
 atherosclerotic *vs.* nonatherosclerotic
 arteries, 315–318
 Bayesian model score, 306
 differential module network model, 313–314
 differential networks, 304
 figures task, 312
 Ganesh task, 310
 GO annotation task, 312–313
 Lemon-Tree software package, 308–309
 model, 304–307
 optimization algorithm, 307–308, 314
 principle of, 303
 regulators task, 311–312

- revamp task, 310–311
 tight clusters task, 310
- Molecular time-course data
 causal models, 39–41
 dynamic Bayesian networks
 discrete time index, 27
 feedforward, 27, 29, 30
 graphical representations of, 28
 inference, 29–30
 multiple non-identical networks,
 estimation of, 31–32
 and ordinary differential equations, 30–31
 nonlinear models, 32–33
 notations and problem set-up, 26–27
 time-varying networks, 33–39
- mRNA decay regulationn, 354
- mRNA transcription rate estimation
 finite difference quotient, 82
 Gaussian process smoothing approach, 82
 network reconstruction accuracy, 84–85
 numerical *vs.* analytical gradient schemes, 82, 83
- Multiattribute GGMs
 gene/protein regulatory network inference, 156–158
 need for, 152–153
 neighborhood selection approach, 153–155
 penalizer selection, 155
 separate variant method, 156
 simulation study, 155–156
- Multi-network models, 16–17
- Multiscale regulatory networks, *see* Multiattribute GGMs
- Multi-task Group LASSO (MTG-LASSO), 182
- Mutual information matrices (MIMs), 299
- Mutual information networks, 8–9
- N**
- Neighborhood selection approach
 multiattribute GGMs, 153–155
 sparse GGMs, 149–151
- NetSI, 326, 328
- Network inference (NI)
 integrating unsupervised learning and
 differential NI, 239–240
 dynamic NI, 240–242
 profile-specific NI, 242–243
 regulatory processes, 245
- scRNA-seq, 235
- single-cell ATAC-seq, 246
- stability
 differential, 340–341
 2D multidimensional scaling (MDS), 327
- Escherichia coli* transcriptional
 regulatory network, 331
 evaluation of, 327
- GENIE3, 332
- GO:0007187 pathway, 336–337
 high performance computing (HPC) clusters, 331
 NetSI, 326
 SynTRen simulator, 337–340
- Next generation sequencing (NGS) technology, 4, 5
- Nonlinear dynamical systems, 274, 278
- Nonlinear models, 32–33
- O**
- On/off model, gene expression, 219
- Ordinary differential equations (ODE)
 models, 32, 33, 252
 Akaike Information Criterion (AIC), 386
 Bayesian Information Criterion (BIC), 386
 DBNs and, 30–31
 model parameters inference
 gradient calculation, 401–404
 Hessian computation, 404–406
 optimization methods, 389–398
 problem formulation, 387–389
 simulation, 398–401
 model structure inference
 model averaging, 412–413
 model selection criteria, 406–409
 reduction, 409–412
- Overfitting, 197, 198
- P**
- Parameter prior approach, 172
- Partial correlation networks, 8
- Pearson correlation, 8
- Per-gene methods, 166
- Per-module methods, 166
- Physical module networks (PMNs), 177–179
- Post-pruning procedure, 199
- Post-transcriptional regulation, 375
- Post-translational regulation, 350
- Pre-pruning procedure, 199–200
- Prior-based framework
 for Bayesian networks, 172–174
 for dependency networks
 Inferelator algorithm, 175–176
 MERLIN-P method, 174–175
- Probabilistic graphical models (PGMs)
 components, 166
 dependency networks, 169
 directed acyclic graphs (Bayesian networks), 169
 from gene expression data, 170–171
 parameter prior approach, 172
 structure prior approach, 172
 undirected graphical models, 169
- Probabilistic models
 Bayesian networks, 11–13
 Gaussian graphical models, 10–11

- Protein activity regulationn, 354–355
 Protein-coding gene, 351
 Protein decay regulationn, 355–356
 Proteomic and transcriptomic data integration,
 in cancer, 144–145
 Pseudo-time methods, 45
- Q**
- Quantitative mass spectrometry, 4
 Quantitative Trait Locus (QTL) detection, 117
- R**
- Random forest, 200
 RankSum, 285
 Rate/gradient estimation, 71–73
 Real data, advantage and disadvantage, 76
 Realistic data
 disadvantage, 76
 generation of
 ANOVA, 80–82
 Bio-PEPA framework, 77, 79
 elementary molecular reactions, 77
 mRNA concentration time series, 79
 network inference scoring schemes, 80
 ordinary differential equations, 76, 77
 protein concentration profiles, 79, 80
 Receiver operating characteristic (ROC) curve, 80
 Regression-based method, 9–10
 Regression trees
 Bagging algorithm, 200
 brute-force approach, 199
 ensemble methods, 200
 example, 197, 198
 Extra-Trees method, 200
 parameters, 200–201
 post-pruning procedure, 199
 pre-pruning procedure, 199–200
 random forest, 200
 variable importance measures, 201–202
 Regulatory networks, 161
 expression-based network inference, 165–171
 inference in practice
 candidate regulators, 186–187
 collecting expression samples, 185–186
 consensus MERLIN-P network and
 modules, 187–188
 Gene Expression Omnibus (GEO), 185–186
 integrate upstream regulators and network, 188
 prior knowledge for regulatory edges, 187
 validation and evaluation, 188–190
 structure and function, 164
 Repressilator, 252
 Restrict–maximize heuristic, 122
 Revamp task, 310–311
- Reverse engineering GRNs
 degree distributions, 6
 degree of a node, 6
 directed networks, 5–7
 in-degree and out-degree, 6
 undirected networks, 5, 7
 weighted networks, 6, 7
 Reverse-phase protein arrays (RPPA) technique, 145, 157
 Riboswitch, 354
 RNA, affymetrix array, 157
 RNA sequencing, 4, 114
 Run jump3
 additional information, 230
 algorithm outputs, 229
 candidate regulators, 229, 230
 Extra-Trees algorithm, 229
 gene names, 231
 links, 230
 links prediction in file, 231
 MATLAB file, 227
 noise parameters, 228
 regulatory links, 230
- S**
- Score-based algorithms, 122
 Sequential information sharing
 binomial prior, 38–39
 exponential prior, 37–38
 schematic illustration, 37
 Signaling and dynamic regulatory events miner
 (SDREM), 179–181
 Signaling networks, 162
 integrating with gene expression data
 physical module networks, 177–179
 signaling and dynamic regulatory events
 miner, 179–181
 stepwise integration strategy, 181–185
 Single-cell RNA sequencing (scRNA-seq), 235, 242
 Single-cell transcriptomic data, 45
 advantages, 236
 inference using perturbational data, 243–244
 inter-cellular heterogeneity, 236
 outputted network by clustering, 238
 tools, 240
 zero-inflation, 237
 Software tools
 ARACNe, 19
 Banjo, 20
 CatNet, 19
 CLR, 19
 G1DBN, 20
 GeneNet, 19
 GENIE3, 19
 GRENITS, 20

- Inferelator, 20
 MRNET, 19
 netbenchmark, 20
 TSNI, 20
 WGCNA, 19
 Sparse Bayesian regression (SBR) method, 58–59
 MATLAB implementation, 74
 Sparse Candidate algorithm, 122
 Sparse GGMs
 graphical-Lasso algorithm, 148–149
 graph of conditional dependency structure, 146
 high-dimensional settings, 147
 irrepresentability conditions, 149
 \mathbf{I}_1 -regularization, 147, 148
 maximum likelihood estimator, 146–147
 model selection issues, 151–152
 neighborhood selection approach, 149–151
 Wishart distribution, 147
 Spatiotemporal GPDS (STGPDS) model, 276
 Stability selection procedure, 151
 STAGE AAW-IMA differential module network, 317
 StaRS (Stability approach to Regularization
 Selection), 151–152
 State-space models (SSM), 60–61
 implementation, 74
 Stepwise integration strategy, of multiple data types
 defining signaling networks upstream of
 modules, 183–185
 integer linear programming approach, 183–185
 MERLIN, 182
 MTG-LASSO, 182
 proteomics data to module, 182–183
 transcriptional network inference, 182
 Stochastic differential equation (SDE),
 Jump3 framework, 219–220
 Structural equation modeling (SEM), 115
 Structural intervention distance (SID), 125
 Structure learning, Bayesian networks
 constraint-based algorithms, 122
 hybrid algorithms, 122
 overview, 121–122
 score-based algorithms, 122
 Structure prior approach, 172
 Supervised learning, 196–197
 Synthetic networks
 generation, 289–290
 results on, 293–295
 Systematic consensus analysis
 aggregation, 287–288
 normalization, 286–287
 Systems biology
 challenges, 49
 Systems Biology Markup Language (SBML), 401
- T**
- Tight clusters task, 310
 TIGRESS, 10
 Time-course data, learning molecular networks using,
 see Molecular time-course data
 Time-varying networks, 17, 33
 change point process, in DBN model, 34–37
 sequential information sharing
 binomial prior, 38–39
 exponential prior, 37–38
 schematic illustration, 37
 Time-varying sparse regression (Tesla), 55–56
 software implementation, 73
 Top-down approaches, 252
 TopkNet, 285
 Topological overlap matrix (TOM), 326
 Transcription factors (TFs), 162, 348
 combinatorial regulation, 350
 Transcription regulation, 350
 methematical formulation of, 50
 Bayesian spline autoregression method, 59–60
 Elastic Net method, 55
 Gaussian Bayesian networks, 66–67
 Gaussian graphical models, 52–54
 Gaussian processes, 61–64
 hierarchical Bayesian regression models, 56–58
 Improved Chemical Model Averaging, 67–72
 Lasso method, 54–55
 Mixture Bayesian network model, 64–65
 notations, 51–52
 rate/gradient estimation, 71–73
 sparse Bayesian regression, 58–59
 state-space models, 60–61
 time-varying sparse regression (Tesla), 55–56
 Translation regulation, 352–354
 Tree-based method
 advantage, 195–196
 GENIE3, 196
 area under the precision-recall curve, 205–207
 computational complexity, 205
 context-specific network inference, 210–211
 dynGENIE3, 207–208
 feature selection problem, 203
 GENIE3-SG-joint, 209
 GENIE3-SG-sep, 209
 goal of, 202
 iRafNet method, 210
 limitation of, 211
 ODE-based approach, 209
 output normalization, 204–205
 parameter analysis, 205–207
 procedure illustration, 203, 204
 running times of, 205, 206

- Tree-based method (*cont.*)
- scalability of, 212
 - SCENIC framework, 209
 - single-cell data analysis, 209
 - software availability, 205
 - goal of, 195
 - unsupervised inference, 211
- Tumorigenesis, 144
- Two-hybrid assays, 165
- U**
- Underfitting, 197, 198
- Undirected graphical models, 169
- Unsupervised GRN ensemble
- assembling pairwise matrices, 298–299
 - Borda* count, 285
 - consensus network algorithms, 288
 - DREAM5 data, 291
 - results on, 295–296
 - heterogeneous networks, 284
 - homogeneous networks, 284
 - homogeneous *vs.* heterogeneous network
 - scenario, 288–289
 - methods, 284–285
- RankSum, 285
- real data, 291–293
 - results on, 296–298
- W**
- Whole-transcriptome causal gene network inference
- Cytoscape, network visualization in, 102
 - Findr installation, 99
 - Geuvadis dataset, 99, 101
 - network reconstruction, 99–100
 - out-degree distribution, 101
 - regulation probabilities, 100
 - statistical methods, 103
 - data normalization, 102
 - likelihood ratio tests, 102, 104
 - local false discovery rate, 102, 105
 - posterior probabilities, of alternative hypotheses, 104–105
 - P-values, 104
 - subtest combination, 105
 - top hub genes list, 101–102