Spring 5-15-2015

Application of Machine Learning to Mapping and Simulating Gene Regulatory Networks

Hien-haw Liow
Washington University in St. Louis

WASHINGTON UNIVERSITY IN ST. LOUIS

Department of Mathematics

Dissertation Examination Committee:

Edward Spitznagel, Chair

Michael Brent, Co-Chair

Barak Cohen

Renato Feres

Victor Wickerhauser

Application of Machine Learning to Mapping and Simulating Gene Regulatory Networks

by

Hien-haw Liow

A dissertation presented to the
Graduate School of Arts and Sciences
of Washington University in
partial fulfillment of the
requirements for the degree
of Doctor of Philosophy

May 2015

St. Louis, Missouri

TABLE OF CONTENTS

Page

Page

Page

Page

Page

Page

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to thank my advisors Professor Michael Brent and Professor Edward Spitznagel for guiding me through my PhD study. Their wisdom and dedication to science have been a great inspiration to me and their generosity in sharing their expertise has given vital force to my research work.

I would also like to thank my committee members Professor Barak Cohen, Professor Renato Feres, and Professor Victor Wickerhauser for taking their time to evaluate my work and for providing advice from different perspectives.

My gratitudes also go to my colleagues and friends in the Brent Lab: Dr. Brian Haynes, Zeke Maier, Drew Michael, Brian Chen, and Holly Brown whose teamwork and advice I have delighted in and benefited from.

I would like to thank my roommate and fellow mathematician Dr. Qingyun Wang, whose elegant view of life and of science has reaffirmed my love for my field of study.

Finally, I would like to thank my mother Huiping Wu and my father Xuguang Liu, who have celebrated with me in moments of breakthroughs and encouraged me in moments of confusion. The optimistic attitude and the passion for life they put in me have carried me along through my seven-year quest of doctoral study.

Dedicated to Huiping Wu and Xuguang Liu

ABSTRACT OF THE DISSERTATION

Application of Machine Learning to Mapping and Simulating Gene Regulatory Networks

by

Hien-haw LIOW (LIU Xuanhao)

Doctor of Philosophy in Mathematics

Washington University in St. Louis, 2015

Edward Spitznagel, Chair

Michael Brent, Co-Chair

This dissertation explores, proposes, and examines methods of applying modern machine learning and Bayesian statistics in the quantitative and qualitative modeling of gene regulatory networks using high-throughput gene expression data. A semi-parametric Bayesian model based on random forest is developed to infer quantitative aspects of gene regulation relations; a parametric model is developed to predict gene expression levels solely from genotype information. Simulation of network behavior is shown to complement regression analysis greatly in capturing the dynamics of gene regulatory networks. Finally, as an application and extension of novel approaches in gene expression analysis, new methods of discovering topological structure of gene regulatory networks are developed and shown to provide improvement over existing methods.

1. Introduction

1.1 Motivation

A gene regulatory network (GRN) is a collection of DNA segments in a cell which interact with each other indirectly (through their RNA and protein expression products) and with other substances in the cell to govern the gene expression levels of mRNA and proteins. GRNs are an important module of cells' ability to respond to environmental changes and are essential to cell lineage differentiation in multicellular organisms. Understanding the mechanisms of GRNs aids the study of human diseases, pharmaceutical research, and industrial genetic engineering.

Various experiment methods provide data for the study of GRNs. Methods such as ChIP-chip, ChIP-seq, and ChIP-exo give insights to the binding locations of proteins on DNA sequences; micro-array and RNA-seq can measure the mRNA abundance of different genes in cells. These methods generate high-throughput data whose processing demands not only strong computational power but also advanced statistical methods.

Using data from the unicellular model species *Saccharomyces cerevisiae* and *Cryptococcus neoformans*, this dissertation develops and discusses mathematical models and computational approaches for investigating GRNs and simulating their behavior.

1.2 Background

1.2.1 Dynamics of Gene Regulatory Networks

Genes interact with each other through the proteins that they encode. Gene sequences are transcribed to mRNAs which are translated to proteins, some of which affect the transcription rates of genes by binding to their promoter sequences and interacting with the RNA synthesis mechanism.

In *Saccharomyces cerevisiae* and *Cryptococcus neoformans*, alternative splicing of mRNAs is rare and it can be considered that each expressed gene is transcribed to one unique mRNA sequence which is then translated to one unique protein. By considering the transcription rate of each gene as a function of protein levels, the dynamics of a GRN (without biological noise) can be modeled as

$$\begin{cases} \frac{\mathrm{d}[protein_i]}{\mathrm{d}t} & = \tau_i[mRNA_i] - \delta_i[protein_i] \\ \frac{\mathrm{d}[mRNA_i]}{\mathrm{d}t} & = f_i([protein_1], [protein_2], \ldots) - d_i[mRNA_i] \end{cases} \tag{1.1}$$

where $[\cdot]$ denotes concentration, i indexes genes, τ_i is the translation rate of gene i, δ_i the degradation of $protein_i$, d_i the degradation rate of $mRNA_i$, t is the time variable, and f_i describes the dependence of the transcription rate of gene i on protein levels.

In RNA-seq and expression micro-array experiments, mRNA concentrations are measured with some random measurement error and biological noises. Conceptually, we identify the mRNA abundance of a gene with its expression level.

1.2.2 Steady State of Gene Expression Levels

Although gene expression levels form a dynamic system and could be constantly changing, the system can reach or approximate a steady state when there is little change in the environment.

Numerous micro-array and RNA-Seq experiments are available that measure the expression levels of genes in cell cultures at single time points in steady environments. Although they do not track the time course trajectory of the gene expression profile of any single sample, the differences in genotype and environment across samples result in different expression profiles which are captured in the experiments and enable the study of gene interactions.

To study steady state expression data, we specialize (1.1) into a steady state model. Assuming

$$\begin{cases} \frac{\mathrm{d}[protein_i]}{\mathrm{d}t} & = 0 \\ \\ \frac{\mathrm{d}[mRNA_i]}{\mathrm{d}t} & = 0 \end{cases} \tag{1.2}$$

(1.1) implies that

$$\begin{cases} [protein_i] & = \frac{\tau_i}{\delta_i}[mRNA_i] \\ \\ [mRNA_i] & = \frac{1}{d_i} f_i \left(\frac{\tau_1}{\delta_1}[mRNA_1], \frac{\tau_2}{\delta_2}[mRNA_2], \ldots \right) \end{cases} \tag{1.3}$$

Denoting

$$g_i(m_1, m_2, \ldots) = \frac{1}{d_i} f_i \left(\frac{\tau_1}{\delta_1} m_1, \frac{\tau_2}{\delta_2} m_2, \ldots \right) \tag{1.4}$$

we can rewrite

$$[mRNA_i] = g_i \left([mRNA_1], [mRNA_2], \ldots \right) \tag{1.5}$$

and g_i will describe the relation between the mRNA levels of different genes. In fact, many steady state expression studies focus on investigating the relation between gene expression levels without explicitly inferring protein levels. The majority of the methods developed and discussed in this thesis are intended for the analysis of steady state expression data.

The presence of technical and biological noise in the measurement of mRNA concentrations demands proper statistical modeling.

1.2.3 Artificial Genetic Perturbations

Without time course data of expression levels, the interaction between genes has to be revealed by comparison between steady state expression arrays of different strains of an organism or that in different environments. Micro-array data of *Saccharomyces cerevisiae* by Hu et al. [14] consists of 263 mutant strains, each of which has one regulator-encoding gene deleted artificially; micro-array data of Holstege et al. [2] provides the steady state expression profiles of 1484 single-gene-deletion strains.

Expression data of mutant strains or environmental changes has been used for network structure inference [3, 4, 15, 16] and expression prediction [8, 13, 15].

1.3 Structure and Contribution

Chapter 2 studies regression analysis that predicts the expression level of a target gene from the levels of its regulator genes. In particular, we developed a method based on Bayesian Additive Regression Trees (BART) [9] that predicts target gene level from regulator levels and is competitive with published methods for the same task. Chapter 3 develops a pioneering model for predicting steady state expression levels of genes from the combination of regulator deletions. In Chapter 4, we design a specific scheme for applying models developed in Chapter 2 for the task for network structure discovery. Using this method and the latest expression data in literature, we construct a functional network for *Saccharomyces cerevisiae* that shows improved accuracy over previously published network inferences.

2. Expression Level as a Function of Regulator Concentrations

2.1 Background

A gene regulatory network can be represented by a graph whose nodes are gene expression levels and whose edges are regulations. The majority of protein-encoding genes do not participate directly in the transcriptional regulation of other genes. A small portion of the protein-encoding genes encode transcription factors, proteins of transcription factor complexes, or modifiers of transcription factors. Transcription factors then bind to the promoter sequences of genes and alter the rate that they transcribe into mRNA's. The genes who play a role in regulating the expression level of other genes are referred to as regulators.

Although interactions between regulators may involve feed back loops and other complicated structures, the relation between regulators and non-regulator genes are relatively simple: the directed edges of gene regulation graph can go into, but not out of, the set of non-regulator nodes. In this chapter, we study different ways of modeling the influence of regulator levels on non-regulator levels.

We work with gene expression arrays such as micro-array and RNA-Seq data to study regulator-target interactions. To formulate the task as statistical problem, we consider

regulator expression levels as explanatory variables and non-regulator gene expression levels as response variables; we propose and examine several models for predicting the latter from the former.

Each target gene may have multiple regulators and the modeling of the combinatorial effect is non-trivial. Moreover, the number of regulators in the studied species correspond to the number of explanatory variables, or features in the terminology of machine learning. In many cases, the number of samples are on the same level as number of features, rendering conventional regression methods subject to the risk of overfitting. An appropriate modeling approach will help understanding both the network structure and quantitative features of gene regulation and inform strategies for simulating the behavior of gene regulatory networks.

2.2 Previous Works

2.2.1 Inferelator

There are several pieces of previous works in expression prediction that incorporates the lasso [18] as their key parts. The regression lasso performs least-square fitting in a linear regression setting subject to a constraint on maximum allowed L^1 norm of the coefficient vectors. The maximum allowed L^1 norm is typically determined by maximizing the prediction accuracy in cross-validation on the training data.

One of the published method of this kind is Inferelator [15]. The explanatory variables that Inferelator passes to lasso include logarithm of regulator levels and of minimum

levels of pairs of regulators (e.g. $\min\{x_{j_1k}, x_{j_2k}\}$ where x_{jk} is the level of regulator j in sample k) representing the concentrations of heterodimer TFs; regulators whose levels are highly correlated are combined into one single feature. These features (regulator log-levels, combined regulator log-levels, minima of regulator pairs) are filtered by their correlation with the response variable and only a few features are handed to the lasso for the prediction of each response variable. A response variable in the problem formulation of Inferelator can be either the log-level of a single target gene or the average log-level of multiple co-regulated genes, as grouped by biclustering in [15]. Inferelator is designed to work with both steady state expression data and/or time-course expression data.

The minimum of the levels of a regulator pair $\min\{x_{j_1k}, x_{j_2k}\}$ in Inferelator was intended for representing the level of the protein heterodimer that is composed of one copy of regulator j_1 and one copy of regulator j_2. However, in data sets where only mRNA levels are measured and no direct measurement of protein concentration is available, it is unclear what is the constant ratio between the mRNA level x_{jk} and the concentration of its translated protein. Different regulators can have different [mRNA]/[protein] ratios due to difference in translation rate and protein degradation rate, therefore it is impossible to construct a feature to represent the minimum of the levels of two different proteins.

Another example of lasso application can be found in [5]. A gene regulatory network was built for *Drosophilia melanogaster*; using the learned network as feature selection method, the same study performed prediction of target levels from regulator levels using the lasso. It was shown that accurate understanding of network structure provides feature selection criteria that improve the accuracy of expression prediction.

2.2.2 k-Nearest Neighbors

An expression prediction method based on k-nearest neighbors (kNN) was published in [10]. There expression prediction is treated as a missing data imputation problem. A distance is calculated for each pair of genes according to the Euclidean distance between their log expression level vectors across different training samples. Based on this distance measure, each gene is assigned with a certain number of nearest neighbors, which can be interpreted as a set of co-regulated genes. The expression level of a query gene in a query sample is predicted by the average level of its neighbor genes whose levels are known in the query sample. Despite the simplicity of the model, it ranked high in the expression prediction competition in DREAM3 (*Dialogue for Reverse Engineering Assessments and Methods 3*).

The method developed in [10] can be directly applied to the task of predicting target levels from regulator levels. Although the method was originally applied on time-course expression data, the simplicity of the data allows it be applied to steady state expression data without any modification.

This method is tested and evaluated in this chapter along with other methods. Several variations of this method are available and they are tested separately. There are two factors of variation:

1) Whether the Euclidean distance between genes is calculated in the space of fold changes Y_{ij} with respect to wild type or in the space of log-fold-changes $\log(Y_{ij})$.

2) The choice of parameter k which stands for number of nearest neighbors assigned to each gene. As the original paper [10] pointed out that $k = 10, 11, \ldots, 30$ provided best performance in cross-validation on the training data, in this chapter we explore the options $k = 1, 2, \ldots, 30$. Since it is not a task of this dissertation to determine the optimal k for the kNN-bases method, we test different choices of k directly on the test data instead of determining it from cross-validation on the training data.

2.2.3 PWM-based Expression Prediction

There have been efforts to predict expression levels using models that incorporate sequence affinity information such as position weight matrix (PWM) [17] scores. One example is [11], where the product of transcription factor concentration and the exponential function of position probability matrix (PPM) [17] score at a position on the genome is considered proportional to the statistical weight of the binding between that position and the TF. The model was able to successfully predict the expression level of certain modules in the segmentation network of *Drosophila* from the levels of 8 TFs. This model requires a smaller number of parameters (proportional to the sum of the number of regulators and the number of targets, while that of a linear regression model is proportional to the product of regulator number and target number). However the complexity of the form of the optimizand could render the model fitting process very time consuming as the number of regulators and targets goes higher. This model requires prior knowledge of the PWM of the studied regulators.

2.3 Data and Evaluation Criteria

2.3.1 Data Type and Interpretation

The methods and models described in this chapter are developed on micro-array data and RNA-Seq data; the methods can be extended to any expression array data. Regulator levels are handled as explanatory variables and non-regulator gene levels as response variables. Each data set is partitioned in to training samples and test samples: the former used for supervising model fitting and the regulator levels of the latter for query features.

2.3.2 Training and Test Data

The methods are trained and tested on RNA-Seq data of *Cryptococcus neoformans* produced in Brent Lab (http://mblab.wustl.edu/) and Doering Lab (http://www.crypto.wustl.edu/) and on micro-array data of *Saccharomyces cerevisiae* published in [2].

The RNA-Seq data of *Cryptococcus neoformans* consists of genome-wide expression profiles of 318 samples, including 117 wild type (the serotype A reference strain H99) samples, 171 samples of 41 single-gene deletion strains, 3 samples of an over-expression strain and 27 samples of 10 double-gene deletion strains. Samples were cultivated and collected in different batches over a 3-year period. Each batch contains wild type samples and may or may not include mutant strains. The training data consists of expression

profiles of single-gene deletion strains and wild type samples; the test data consists of the expression profiles of double-gene deletion strains.

The micro-array data of *Saccharomyces cerevisiae* consists of genome-wide expression profiles of 1484 single-deletion strains and wild type yeast of haploid MATalpha (BY4742) and MATa(BY4741) cultivated in G418 YPD [2]. The processed data published in [2] is used, where the differential expression analysis tool LIMMA [16] was applied to the raw data, generating the p-values of differential expressions as well as estimated fold changes of expression levels with respect to wild type. The fold changes of expression levels with respect to wild type is used in this thesis as expression array data. The data consists of the expression profiles of 1484 mutant strains of single deletions of both regulator genes and non-regulator genes. To test the contribution of regulator deletion strains to the predictive power of the model, we perform 2-fold cross-validation between the set of regulator deletion strains and the set of non-regulation strains.

2.3.3 Normalization of RNA-Seq

The unit of expression levels in an analysis can potentially be arbitrary. This subsection standardizes the unit of RNA-Seq data for later analysis in this thesis.

Using the wild type expression level of a gene as the unit can provide great convenience for notations and parameterization. However, wild type samples as well as mutated samples are collected in multiple experiments and might not be readily comparable with each other due to batch effect. To correct the batch effect, we use the average wild type expression profile of each batch as its normalizer.

Denote the index set of samples in a certain batch as $batch_j$ and the index set of wild type samples WT. The index set of wild type samples in the jth batch is therefore $batch_j \cap WT$. The average wild type expression level of gene i in batch j is defined as

$$\frac{1}{|batch_j \cap WT|} \sum_{l \in batch_j \cap WT} FPKM_{il} \tag{2.1}$$

where $FPKM_{il}$ is the expression level of gene i in sample l measured in fragments per kilobase per million reads (FPKM).

Using this quantity as the normalizer, the normalized expression level of a gene i in sample $k \in batch_j$ is defined as:

$$\forall k \in batch_j, Y_{ik} = \frac{FPKM_{ik} + \nu}{\frac{\sum_{l \in batch_j \cap WT} FPKM_{il}}{|batch_j \cap WT|} + \nu} \tag{2.2}$$

where a small pseudocount $\nu > 0$ is added to both the numerator and denominator to avoid the case of zero denominator and to provide numerical stability when the expression levels is very low. The pseudocount is picked to be a number much smaller than the majority FPKM.

This definition normalizes each batch of samples using wild type information from the same batch. At the same time, it enforces the average wild type expression level of a gene as its unit of expression levels - hence Y_{ik} may also be interpreted as the fold change of expression level with respect to wild type. The normalization is applied on both the training data and the test data.

Note that under this definition, there is a positive minimum expression level for each gene i due to the usage of pseudocount:

$$Y_{ik} \geq \frac{\nu}{\max\left\{\frac{\sum_{l \in batch_j \cap WT} FPKM_{il}}{|batch_j \cap WT|} : j = 1, 2, \ldots\right\} + \nu} \tag{2.3}$$

The normalized expression level Y_{ik} will henceforth be used in all analyses in this thesis unless otherwise indicated. They will simply be referred to as "expression levels" or "fold changes".

2.3.4 Assessment of Quantitative Prediction

The performance of methods is measured in their accuracy of predicting log-fold-changes of expression levels with respect to wild type.

In particular, denoting the prediction of Y_{ik} as $\hat{Y}_{ik}$, we examine the accuracy of $\log_2(\hat{Y}_{ik})$ as the prediction of $\log_2(Y_{ik})$. Mean square error (MSE), R-Squared, and correlation are the metrics of accuracy.

Denoting the index set of test samples as S and the set of genes to predict as G, the metrics of accuracy are defined as:

$$\begin{cases} MSE = \frac{\sum_{i\in G,k\in S}\left(\log_2 \hat{Y}_{ik} - \log_2 Y_{ik}\right)^2}{|G||S|} \\ rmse = \sqrt{MSE} \\ R^2 = 1 - \frac{MSE}{s^2_{G\times S}(\log_2 Y)} \\ Corr = \frac{\sum_{i\in G,k\in S}\left((\log_2 \hat{Y}_{ik})(\log_2 Y_{ik})\right)}{\sqrt{\sum_{i\in G,k\in S}\left(\log_2 \hat{Y}_{ik}\right)^2 \sum_{i\in G,k\in S}\left(\log_2 Y_{ik}\right)^2}} \end{cases} \tag{2.4}$$

where

$$s^2_{G\times S}(\log_2 Y) = \frac{1}{|G||S|-1}\sum_{i\in G,k\in S}(\log_2 Y_{ik})^2 - \frac{1}{|G||S|(|G||S|-1)}\left(\sum_{i\in G,k\in S}\log_2 Y_{ik}\right)^2 \tag{2.5}$$

is the sample variance of $\log_2 Y$ in the test data.

2.4 Simple Parametric Models

Given the complexity of molecular mechanism of gene-protein interactions, the modeling approaches can range from detail-oriented such as the PWM-based model in [11] to highly abstracted such as a simple linear regression. In this chapter, we focus on the more abstract approaches, which requires little specific knowledge and few assumptions regarding the molecular level mechanism.

This subsection explores several options of modeling the deterministic relation between regulator levels and target levels.

The most basic model would be a linear regression model. However, a linear model cannot guarantee that the predictions of the response variable, i. e. the expression level of the target gene, are non-negative. Here we discuss several modified versions of the naïve linear regression model.

The first one enforces a cutoff at zero when the linear combination of regulator levels calculates to a negative number. It will be referred to as the *additive model with cutoff*:

$$F_i^{(add)}(x_1, x_2, \dots) = \max\left\{0, b_i + \sum_j c_{ij} x_j\right\} \tag{2.6}$$

A more common way is to model the relation between target levels and regulator levels as log-linear:

$$\log\left(F_i^{(lln)}(x_1, x_2, \dots)\right) = \log(b_i) + \sum_j c_{ij} \log(x_j) \tag{2.7}$$

This model is used in Inferelator [15]. It may be expressed equivalently as

$$F_i^{(lln)}(x_1, x_2, \dots) = b_i \prod_j x_j^{c_{ij}} \tag{2.8}$$

This model suffers from an inherent limitation in modeling repressor effects. Consider a certain regulator j that is a repressor. It is modeled by setting $c_{ij} < 0$, which along with (2.8) implies that

$$\lim_{x_j \to 0} F_i^{(lln)}(x_1, x_2, \dots) = \infty \tag{2.9}$$

That is, when c_{ij} is significantly smaller than 0 and $x_j \approx 0$, $F_i^{(lln)}(x_1, x_2, \dots)$ will be a huge number and would introduce numerical instability. Alternatively, if $c_{ij} < 0$ but $|c_{ij}| \ll 1$, $F_i^{(lln)}(x_1, x_2, \dots)$ may behave well around $x_j \approx 0$, but $x_j^{c_{ij}} \approx 1$ for even slightly larger x_j, implying that the repression effect of regulator j has to be modeled to saturate at a very low concentration.

This particular limitation can be avoided by modeling the logarithm of the target level as the linear combination of regulator levels. This model will be referred to as the *exponential model*:

$$F_i^{(exp)}(x_1, x_2, \dots) = b_i \exp\left(\sum_j c_{ij} x_j\right) \tag{2.10}$$

However, while not suffering from the limitation of the log-linear model, the exponential model introduces another instability: when $c_{ij} > 0$, $F_i^{(exp)}$ would increase exponentially as x_j increases.

It has been suggested that in *Saccharomyces cerevisiae* the *in vivo* basal transcription level of a gene in absence of regulators is close to zero due to repression by histones and that the presence of activators is required to release such repression and to allow transcription [7, 19]. In the exponential model, this requires $b_i \approx 0$ but $b_i \exp(c_{ij}) \gg 0$ for

certain i, j, implying that $c_{ij} \gg 0$ and further contributing to the numerical instability as x_j increases.

To avoid the numerical limitation of both the log-linear model and the exponential model, a combination of the two is proposed here:

$$F_i^{(lle)}(x_1, x_2, \dots) = b_i \exp\left(\sum_j \left(c_{ij} \log(x_j) + d_{ij} x_j\right)\right) \tag{2.11}$$

This model will be referred to as the *log-linear-exponential model*. Note that this model has a larger number of parameters than the log-linear model and the exponential model. In this model, the repression effects have the option of been modeled exponentially while the activation effects have the option of been modeled log-linearly. We do not enforce a preference for these options *a prior* but leave the choice to the training process. In fact, we later confirmed in the trained models that log-linear coefficients c_{ij} has a slightly yet significantly higher chance of been positive than exponential coefficients d_{ij}, with p-value $< 2.2^{-16}$ for *C. neoformans* and p-value < 0.03 for *S. cerevisiae*. Details can be found in Table 2.5 in the result section.

Inspired by the thermodynamic understanding of gene-protein interactions, it is possible to define a model enforcing a maximum expression level for each gene:

$$F_i^{(th)}(x_1, x_2, \dots) = \frac{M_i}{1 + \frac{1}{r_i \prod_j \frac{1+\omega_{ij}\theta_j x_j}{1+\omega_{ij} x_j}}} \tag{2.12}$$

where M_i is the maximum expression level of gene i, ω_{ij} describes the affinity between gene i and regulator j, and θ_j describes the affinity between RNA polymerase (RNAP) and regulator j: $\theta_j < 1$ implies that the regulator is a repressor and $\theta_j > 1$ implies that the regulator is a activator. This model will be referred to as the *thermodynamic model*. The training of this model would be challenged by the larger number of parameters, the coupling between the effect of certain parameters, and difficulty to fit M_i when the genes are seldom expressed close to the maximum expression level. Moreover, since the affinity between a regulator j and RNAP is described only by one parameter θ_j, it is impossible to model a regulator as both activator and repressor.

Five different parametric models have been introduced in this subsection: log-linear model (2.8), additive model with cutoff (2.6), exponential model (2.10), log-linear-exponential model (2.11), and thermodynamic model (2.12).

2.5 Optimization of Parametric Models

2.5.1 Formulation as Lasso-regression

The coefficient parameters in all three of the log-linear model (2.8), the exponential model (2.10), and the log-linear-exponential model (2.11) can be fit as linear regression coefficients. The number of coefficients per target is equal to the number of regulators for the former two and twice the number of regulators for log-linear-exponential model. The number of regulators can be more than 200 in *Cryptococcus neoformans* and in *Saccharomyces cerevisiae* and poses a potential overfitting problem.

The method lasso [18] is intended to guard against overfitting in linear regression by enforcing a restriction on the maximum L^1 norm of the coefficient vector:

$$loss(\alpha, \vec{\beta}; \vec{y}, x) = \sum_k \left(y_k - (\alpha + \vec{\beta} \cdot \vec{x}_k) \right)^2 ; \tag{2.13}$$

$$\left(\hat{\alpha}_{(ls)}, \hat{\vec{\beta}}_{(ls)} \mid \vec{y}, x \right) = \underset{\alpha, \vec{\beta}}{\operatorname{argmin}} \left(loss(\alpha, \vec{\beta}; \vec{y}, x) \right) ; \tag{2.14}$$

$$\left(\hat{\alpha}_{(lasso)}, \hat{\vec{\beta}}_{(lasso)} \mid \vec{y}, x, \tau \right) = \underset{\alpha, \vec{\beta}: \|\vec{\beta}\|_1 \leq \tau \|\hat{\vec{\beta}}_{(ls)}\|_1}{\operatorname{argmin}} \left(loss(\alpha, \vec{\beta}; \vec{y}, x) \right) ; \tag{2.15}$$

where $\left(\hat{\alpha}_{(ls)}, \hat{\vec{\beta}}_{(ls)} \right)$ is the ordinary least square fit, $\left(\hat{\alpha}_{(lasso)}, \hat{\vec{\beta}}_{(lasso)} \right)$ is the lasso fit, $\tau \in [0, 1]$ is a tuning parameter, and $\|\vec{\beta}\|_1 = \sum_j |\beta_j|$ is the L^1 norm of $\vec{\beta}$.

The tuning parameter τ is referred to as L^1-shrinkage and its choice is vital for the predictive power of the lasso. It is determined by minimizing the mean squared error in cross-validation on the training set.

It is typical to use a scaling factor (normalizer) between variables in a practical problem and the lasso variables. For example, to formulate the log-linear model (2.8) as a lasso regression problem, we reparameterize it as

$$\frac{\log(Y_{ik})}{\upsilon_i} = \alpha_i + \sum_j \beta_{ij} \frac{\log(x_{jk})}{\xi_j} \tag{2.16}$$

where ξ_j and υ_i are scalers or normalizers. In Inferelator for example, ξ_j is taken as the sample standard deviation of $\log(x_j)$ and υ_i the sample standard deviation of $\log(Y_i)$ in

the training data. A lasso regression problem is defined for each gene i with observations of the response variable being

$$\frac{\log(Y_{i1})}{\upsilon_i}, \frac{\log(Y_{i2})}{\upsilon_i}, \dots \tag{2.17}$$

and $\log(x_{jk})/\xi_j$ is the kth observation of the jth lasso explanatory variable.

The relation between the lasso parameters (α_i, β_{ij}) and parameters in (2.8) is

$$\begin{cases} b_i = \exp(\upsilon_i \alpha_i) \\ c_{ij} = \frac{\upsilon_i}{\xi_j} \beta_{ij} \end{cases} \tag{2.18}$$

2.5.2 Normalization of Lasso Variables

The most common lasso normalizer is standard deviation in the training data, as described in the previous subsection. We explore two other options here: 1) no normalizing - i. e. all normalizers equal to one and 2) the normalizing method used in NetProphet [4].

NetProphet enforces two "noise floors" ξ_0, υ_0. When the sample standard deviation is bigger than the noise floor, the sample standard deviation is used as the normalizer (as in Inferelator); otherwise, the noise floor is used as the normalizer:

$$\xi_j = \max\left\{s_{\log(x_j)}, \xi_0\right\} \tag{2.19}$$

$$\upsilon_i = \max\left\{s_{\log(Y_i)}, \upsilon_0\right\} \tag{2.20}$$

where $s_{\log(x_j)}$ is the sample standard deviation of $(\log(x_{j1}), \log(x_{j2}), \ldots)$ and $s_{\log(Y_i)}$ is the sample standard deviation of $(\log(Y_{i1}), \log(Y_{i2}), \ldots)$.

The noise floors are calculated from average sample standard deviations and deviation of sample standard deviations. Denoting the index set of investigated genes as I,

$$\upsilon_0 = \frac{1}{|I|}\left(\sum_{i\in I} s_{\log(Y_i)}\right) + \sqrt{\frac{1}{|I|-1}\left(\sum_{i\in I} s^2_{\log(Y_i)}\right) - \frac{1}{|I|(|I|-1)}\left(\sum_{i\in I} s_{\log(Y_i)}\right)^2} \quad (2.21)$$

Similarly, denoting the index set of regulators as J,

$$\xi_j = \frac{1}{|J|}\left(\sum_{j\in J} s_{\log(x_j)}\right) + \sqrt{\frac{1}{|J|-1}\left(\sum_{j\in J} s^2_{\log(x_j)}\right) - \frac{1}{|J|(|J|-1)}\left(\sum_{j\in J} s_{\log(x_j)}\right)^2} \quad (2.22)$$

Tested on *C. neoformans* data, the standard deviation normalizer provided slightly better prediction ($R^2 \approx 0.51$) than NetProphet normalizer ($R^2 \approx 0.47$), both better than the unnormalized lasso ($R^2 \approx 0.37$) (see the result section).

2.5.3 Determination of L^1-Shrinkage Parameter

The fitting of parameters pertaining to each target gene can be considered as an individual lasso regression problem, therefore there is allowed to be one L^1- shrinkage parameter τ_i for each target gene. In this setup, each τ_i is determined separately by cross-validation and is referred to as *local shrinkage parameters*.

In NetProphet [4], it has been shown that using a global shrinkage parameter, i.e requiring $\tau_1 = \tau_2 = \tau_3 = \ldots$ enhances the power of the model for recovering the edges of

the gene regulatory network of *Saccharomyces cerevisiae*. The global shrinkage parameter can be determined by minimizing the mean squared error of all response variables in cross-validation.

Tested on *C. neoformans* data, local shrinkage provided better prediction ($R^2 \approx 0.51$) than global shrinkage ($R^2 \approx 0.42$) (see the result section).

2.6 Semi-parametric Bayesian Model

2.6.1 Modeling Noise

Besides the lasso-based approach that minimizes mean square error of log-expression level prediction, alternatively the expression prediction problem can be tackled using a Bayesian approach.

As biological and technical noises are prevalent in expression array data, it is necessary to include noise modeling in expression prediction. For micro-array data, expression levels are modeled to follow the log-normal distribution when parameters are fixed. When being extended to RNA-Seq data, the log-normal noise model exhibited incompetency as the read counts can be equal or close to zero yet log-normal distributed variables only take positive values. Although the pseudocount used in normalization process (2.2) may guarantee that Y_{ik} never reaches zero, small variations in the Y_{ik} when Y_{ik} is close to zero will be exaggerated by $\log(Y_{ik})$ and a log-normal noise distribution might strive to train the model to explain these variations beyond their biological significance.

Here we aim to design a noise model that behaves like the log-normal distribution when $Y_{ik} \gg 0$ yet is not oversensitive when $Y_{ik} \approx 0$.

The adoption of the log-normal noise model would imply that $\log(Y_i)$ follows a normal distribution. This inspires us to design a smooth transformation u such that $u(0) > 0$ but $u(y) \approx \ln(y)$ for $y \gg 0$ and to model $u(Y_{ik})$ as a normal distributed variable. This will allow us to take advantage of many convenient qualities of the normal distribution.

The proper design of transformation u depends on the behavior of the expression noise when the expression is close to zero or at a low level. Literature discussed the relation between noise level and expression level of genes measured in RNA-Seq experiments [3,6]. Measured in counts per million reads (cpm), the variance and expectation of expression levels have been suggested to approximate a quadratic relation. Specifically, denoting the expected level of cpm of gene i in sample k as λ_{ik},

$$\mathrm{Var}(CPM_{ik}) \approx \lambda_{ik} + \phi_i \lambda_{ik}^2 \tag{2.23}$$

where the first term accounts for technical noise and the second term for biological noise; ϕ_i defines the magnitude of biological noise and does not depend on the condition.

In practice cpm might not always be the unit of expression levels. Thus the variance instead could be approximated by with an additional scaling factor ψ_i for each gene:

$$\mathrm{Var}(Y_{ik}) \approx \psi_i \,\mathrm{E}(Y_{ik}) + \phi_i \,\mathrm{E}^2(Y_{ik}) \tag{2.24}$$

Moreover, literature [4] suggests a noise floor - i. e. a minimum nonzero noise level that is present even when then the expression level is close to zero, which can be incorporated as:

$$\mathrm{Var}(Y_{ik}) \approx \theta_i + \psi_i \,\mathrm{E}(Y_{ik}) + \phi_i \,\mathrm{E}^2(Y_{ik}) \tag{2.25}$$

Above is an empirical statement about the dependency of the noise level of Y_{ik} on $\mathrm{E}(Y_{ik})$. It remains to define a specific statistical model that caters to this empirical observation.

Note that if we let the transformed variable $U_{ik} = u_i(Y_{ik})$ and assume that $\sqrt{\mathrm{Var}(Y_{ik})} \ll \mathrm{E}(Y_{ik})$, then

$$\mathrm{Var}(Y_{ik}) \approx \mathrm{Var}(U_{ik}) u_i' \left(\mathrm{E}(Y_{ik})\right)^{-2} \tag{2.26}$$

If the transformation $u_i(\cdot)$ satisfies $u_i' \left(y\right)^{-2} = 1 + \theta_i^{-1}\psi_i y + \theta_i^{-1}\phi_i y^2$ and U_{ik} is modeled as a normally distributed variable of unknown variance $\sigma_i^2 = \theta_i$ and unknown mean, (2.25) will naturally follow. Hence we denote $p_i = \theta_i^{-1}\psi_i, q_i = \theta_i^{-1}\phi_i$ and solve for the following differential equation for u:

$$\begin{cases} u_i' \left(y\right)^{-2} = 1 + p_i y + q_i y^2 \\ u_i(1) = 0 \\ y > 0 \end{cases} \tag{2.27}$$

The solution is:

$$u_i(y) = \sqrt{1 + p_i + q_i} \ln \left(\frac{y + p_i/2 + \sqrt{y^2 + p_i y + q_i}}{1 + p_i/2 + \sqrt{1 + p_i + q_i}} \right) \tag{2.28}$$

Therefore the noise model is defined, with additional parameters p_i, q_i, and σ_i^2:

$$\begin{cases} U_{ik} \mid p_i, q_i, \sigma_i^2, F_i \sim \mathcal{N}\left(u\big(F_i(x_{1k}, x_{2k}, \ldots); p_i, q_i\big), \sigma_i^2\right) \\ Y_{ik} = u^{-1}\left(U_{ik}; p_i, q_i\right) \end{cases} \tag{2.29}$$

where $x_{1k}, x_{2k}, \ldots$ are transcription factor levels in sample k, F_i is the function that determines gene i level from regulator levels without considering noise, Y_{ik} is the observed expression level of gene i in sample k, and

$$\begin{cases} u(y; p, q) := \sqrt{1 + p + q} \ln \left(\frac{y + p/2 + \sqrt{y^2 + py + q}}{1 + p/2 + \sqrt{1 + p + q}} \right) \\ u^{-1}(\xi; p, q) := \left(1 + \frac{p}{2}\right) \cosh\left(\frac{\xi}{\sqrt{1+p+q}}\right) + \left(\sqrt{1 + p + q}\right) \sinh\left(\frac{\xi}{\sqrt{1+p+q}}\right) - \frac{p}{2} \end{cases} \tag{2.30}$$

Note that $u'(y; p, q) = (y^2 + py + q)^{-1/2}$, $u'(1; p, q) = 1$, and $u(1; p, q) = 0$.

Calculations show that:

$$\begin{cases} \mathrm{E}(Y_{ik} \mid p_i, q_i, \sigma_i^2, F_i) = \sqrt{1 + \tilde{\sigma}_i^2} F_i(x_{1k}, x_{2k}, \ldots) + (\sqrt{1 + \tilde{\sigma}_i^2} - 1) p_i/2 \\ \mathrm{Var}(Y_{ik} \mid p_i, q_i, \sigma_i^2, F_i) = \tilde{\sigma}_i^2 \, \mathrm{E}^2(Y_{ik}) + \tilde{\sigma}_i^2 p_i \, \mathrm{E}(Y_{ik}) + \tilde{\sigma}_i^2 q + \tilde{\sigma}_i^4 (q_i/2 - p_i^2/8) \end{cases} \tag{2.31}$$

where

$$\tilde{\sigma}_i^2 = \exp(\frac{\sigma_i^2}{1 + p_i + q_i}) - 1 \tag{2.32}$$

Note that $\mathrm{Var}(Y_{ik})$ is a quadratic form of $\mathrm{E}(Y_{ik})$, which is in agreement with the heuristic noise model (2.25). Specifically, when $\sigma_i^2 \ll 1$,

$$\tilde{\sigma}_i^2 \approx \sigma_i^2 \tag{2.33}$$

$$\mathrm{Var}(Y_{ik} \mid p_i, q_i, \sigma_i^2, F_i) \approx \sigma_i^2 \,\mathrm{E}^2(Y_{ik}) + \sigma_i^2 p_i \,\mathrm{E}(Y_{ik}) + \sigma_i^2 q + \sigma_i^4 (q_i/2 - p_i^2/8) \tag{2.34}$$

$$= \sigma_i^2 \,\mathrm{E}^2(Y_{ik}) + \sigma_i^2 p_i \,\mathrm{E}(Y_{ik}) + \sigma_i^2 q (1 + \sigma_i^2 (1/2 - p_i^2 q_i^{-1}/8)) \tag{2.35}$$

$$\approx \sigma_i^2 \,\mathrm{E}^2(Y_{ik}) + \sigma_i^2 p_i \,\mathrm{E}(Y_{ik}) + \sigma_i^2 q \tag{2.36}$$

$$= \theta_i + \psi_i \,\mathrm{E}(Y_{ik}) + \phi_i \,\mathrm{E}^2(Y_{ik}) \tag{2.37}$$

with the last step almost identical to (2.25).

One may consider (2.29) as a generalization of log-normal noise model: if $p_i = q_i = 0$, $u(\cdot; p_i, q_i) = \ln$.

Later we did normality test on the transformed variable U_{ik} and saw its consistently higher similarity with the normal distribution than that of $\log(Y_{ik})$ (see Table 2.2 and Table 2.3).

2.6.2 Bayesian Prior of Noise Parameter σ^2

Parameters p_i, q_i are tuning parameters and are not modeled with a prior. The tuning process is described in Subsection 2.7.1.

When other parameters are fixed, the estimation of each σ_i^2 is simply a normal estimation problem with known mean (0) and unknown variance: denote $Z_{ik} = U_{ik} - u(F_i(x_{1k}, x_{2k}, \ldots); p_i, q_i)$, the noise model (2.29) can be rewritten as

$$Z_{i1}, Z_{i2}, \ldots \mid \sigma_i^2, F_i, p_i, q_i \text{ i.i.d.} \sim \mathcal{N}(0, \sigma_i^2) \tag{2.38}$$

This motivates us to assign an inverse-gamma distribution to $\sigma_i^2 \mid F_i, p_i, q_i$, which is the conjugate prior of variance in Bayesian normal parameter estimation.

The Bayesian estimation problem of $\sigma_i^2 \mid F_i, p_i, q_i$ for each i is then formulated as:

$$\sigma_i^2 \mid F_i, p_i, q_i \sim \text{Inv}\Gamma\left(\frac{df_i}{2}, \frac{df\, s_{i0}^2}{2}\right) \tag{2.39}$$

$$Z_{i1}, Z_{i2}, \ldots \mid \sigma_i^2, F_i, p_i, q_i \text{ i.i.d.} \sim \mathcal{N}(0, \sigma_i^2) \tag{2.40}$$

$$Z_{ik} = U_{ik} - u(F_i(\vec{x}_k); p_i, q_i) \tag{2.41}$$

where df_i (degree of freedom) and s_{i0}^2 (prior suggested value of σ_i^2) are super parameters.

The criterion of choosing df_i and s_{i0}^2 and the estimation method of σ_i^2 given F_i, p_i, q_i can be found in Subsection 2.7.2.

2.6.3 Modeling Regulator-target Interaction with Regression Trees

Regulator-target interactions may be complicated and involve various molecular mechanisms such as DNA-reshaping, histone relocation, etc. Aside from modeling the details of all known mechanisms or attempting to simplify the interactions into simple parametric regression models, one can invoke non-parametric models to describe the relation between regulator and target levels.

Bayesian Additive Regression Tree (BART) [9] proves to be an effective method for predicting target gene expression level from regulator levels. BART is an ensemble Bayesian regression model, with the predictive function being a sum of small decision trees, referred to as "weak learners". One significant feature that distinguishes BART from other ensemble methods is that it penalizes the complexity of each member tree, motivated by the observation that an ensemble of weak learners sharing well distributed weights is more robust than a model than relies on a small number of strong learners.

Copying the noise model (2.29), the full BART-based expression model is defined as:

$$\begin{cases} \sigma_1^2, \sigma_2^2, \ldots \mid B_i, p_i, q_i \text{ i.i.d.} \sim \text{Inv}\Gamma\left(\frac{df_i}{2}, \frac{df_i s_{i0}^2}{2}\right) \\ B_i \mid p_i, q_i \sim \text{BART Prior Distribution} \\ U_{ik} \mid p_i, q_i, \sigma_i^2, B_i \sim \mathcal{N}\left(B_i(\log(x_{1k}), \log(x_{2k}), \ldots), \sigma_i^2\right) \\ Y_{ik} = u^{-1}\left(U_{ik}; p_i, q_i\right) \\ u(y; p, q) := \sqrt{1+p+q}\ln\left(\frac{y+p/2+\sqrt{y^2+py+q}}{1+p/2+\sqrt{1+p+q}}\right) \\ u^{-1}(\xi; p, q) := \left(1+\frac{p}{2}\right)\cosh\left(\frac{\xi}{\sqrt{1+p+q}}\right) + \left(\sqrt{1+p+q}\right)\sinh\left(\frac{\xi}{\sqrt{1+p+q}}\right) - \frac{p}{2} \end{cases} \tag{2.42}$$

With p_i and q_i fixed, the non-parametric $B_i(\cdot)$ is a BART predictive function, which is an ensemble of regression trees whose form and prior distribution follows the definition in [9]. Regulator levels in sample k are denoted as $x_{1k}, x_{2k}, \ldots$ and target levels as $Y_{1k}, Y_{2k}, \ldots$

2.7 Optimization of Semi-Parametric Model

2.7.1 Choosing Tuning Parameters

Parameters p_i and q_i affect the form of the loss function hence we do not attempt to modify them during the process of optimizing the model. It is necessary to determine these parameters prior to model optimization.

For micro-array data, p_i and q_i are set to zero. Note that expression level measurements in micro-array are usually non-zero, therefore $u(Y_{ik}; 0, 0)$, which is equal to $\ln(Y_{ik})$, is guaranteed to take finite values only.

For RNA-Seq data, an approach inspired by that in the voom method [3] is adopted to determine p_i and q_i. We calculate cross-replicates variance for each gene in each strain and tune p_i, q_i and σ_i^2 to fit the quadratic relation between $\mathrm{Var}(Y_{ik})$ and $\mathrm{E}(Y_{ik})$.

Denoting the level of gene i in sample k measured in count per million reads (cpm) as CPM_{ik}, we assume that there are universal parameters $\sigma^2, \tilde{p}, \tilde{q}$ for all genes in all conditions such that

$$\mathrm{Var}(CPM_{ik} + \nu_i) \approx \sigma^2 \, \mathrm{E}^2(CPM_{ik} + \nu_i) + \sigma^2 \tilde{p} \, \mathrm{E}(CPM_{ik} + \nu_i) + \sigma^2 \tilde{q} \tag{2.43}$$

where $\nu_i \ll \mathrm{E}(CPM_i)$ is the pre-determined pseudocount in cpm. ν_i is equivalent to the pseudocount ν in the data normalization process (2.2), but measured in cpm instead of FPKM.

The tuning parameters p_i and q_i are related to $\sigma^2, \tilde{p}, \tilde{q}$ and latter ones are estimates from the data in the following steps.

Assuming that $\mathrm{Var}(CPM_{ik}) \ll \mathrm{E}^2(CPM_{ik})$, by straight forward calculation similar to that is shown in [3],

$$\mathrm{Var}(\log(CPM_{ik} + \nu_i)) \approx \sigma^2 + \sigma^2 \tilde{p} (\mathrm{E}(CPM_{ik} + \nu_i))^{-1} + \sigma^2 \tilde{q} (\mathrm{E}(CPM_{ik} + \nu_i))^{-2} \tag{2.44}$$

This quadratic relation is fitted in the following procedure.

First, sample variance $s^2_{ij(\log)}$ of log-cpm of gene i in strain j is calculated, as well as the sample mean μ_{ij} of cpm:

$$\mu_{ij} = \frac{1}{|K_j|} \sum_{k \in K_j} CPM_{ik} + \nu_i \tag{2.45}$$

$$\mu_{ij(\log)} = \frac{1}{|K_j|} \sum_{k \in K_j} \log(CPM_{ik} + \nu_i) \tag{2.46}$$

$$s^2_{ij(\log)} = \frac{1}{|K_j| - 1} \sum_{k \in K_j} (\log(CPM_{ik} + \nu_i) - \mu_{ij(\log)})^2 \tag{2.47}$$

where K_j is the index set of samples from strain j.

Linear regression is performed to solve for coefficients c_0, c_1, c_2 in:

$$s^2_{ij(\log)} \sim c_0 + c_1 \mu_{ij}^{-1} + c_2 \mu_{ij}^{-2} \tag{2.48}$$

We define $p = c_1/c_0$ and $q = c_2/c_0$. Further, another linear fit is solved to determine the proper conversion coefficient that relate expression levels measured in cpm and in fold change Y_{ik}:

$$CPM_{ik} + \nu_i \sim w_i Y_{ik} \tag{2.49}$$

Converting the unit from cpm to fold change, the tuning parameters p_i and q_i are determined as

$$\begin{cases} p_i &= \tilde{p}/w_i \\ q_i &= \tilde{q}/w_i^2 \end{cases} \tag{2.50}$$

2.7.2 Marginalizing Noise Parameters

When all other parameters are fixed, the noise parameters σ_i^2 and the observation forms a Bayesian normal estimation problem with known mean:

$$\begin{cases} \sigma_i^2 \mid p_i, q_i, F_i \sim \text{Inv}\Gamma\left(\frac{df_i}{2}, \frac{df\, s_{i0}^2}{2}\right) \\ u(Y_{ik}; p_i, q_i) - u(F_i(\vec{x}_k); F_i, p_i, q_i) \mid p_i, q_i, F_i \text{ independently} \sim \mathcal{N}(0, \sigma_i^2); \end{cases} \tag{2.51}$$

where $\vec{x}_k = (x_{1k}, x_{2k}, \dots,)^T$ is the regulator expression profile of sample k.

In the implementation of the method, the super parameters df_i and s_{i0}^2 may be user specified. By default, they are determined using linear model estimation in R package *BayesTree*, where a linear regression is performed between the U_{ik} and $(\log(x_{1k}), \log(x_{2k}), \dots)$ and df_i and s_{i0}^2 are chosen to make the prior distribution $\text{Inv}\Gamma(df_i/2, df\, s_{i0}^2/2)$ of σ_i^2 emulate the distribution of the squared error of the linear model [9].

Since the conjugate prior is used, the posterior distribution of $\sigma_1^2, \sigma_2^2, \dots$ conditioning on other parameters is, by calculation in [12]:

$$\sigma_i^2 \mid p_i, q_i, F_i, Y_{i1}, Y_{i2}, \dots \sim \text{Inv}\Gamma\left(\frac{df}{2} + \frac{n}{2}, \frac{df\, s_0^2}{2} + \frac{\sum_{k=1}^n \left(u(Y_{ik}; p_i, q_i) - u(F_i(\vec{x}_k); p_i, q_i)\right)^2}{2}\right) \tag{2.52}$$

where $k = 1, 2, \dots, n$ are the indices of the training samples.

The model evidence of this Bayesian problem is by definition equal to $p(Y_{i1}, Y_{i2}, \ldots \mid F_i, p_i, q_i)$. By calculation in [12], the model evidence as a conditional probability density function is

$$p(Y_{i1}, Y_{i2}, \ldots \mid p_i, q_i, F_i) = \frac{p(Y_{i1}, Y_{i2}, \ldots \mid \sigma_i^2, p_i, q_i, F_i) p(\sigma_i^2 \mid p_i, q_i, F_i)}{p(\sigma_i^2 \mid p_i, q_i, F_i, Y_{i1}, Y_{i2}, \ldots)} \tag{2.53}$$

$$= \frac{1}{(2\pi)^{\frac{n}{2}}} \frac{\beta^{\alpha}}{\left(\beta + \frac{\sum_{k=1}^{n} (u(Y_{ik}; p_i, q_i) - u(F_i(\vec{x}_k); p_i, q_i))^2}{2}\right)^{\alpha + \frac{n}{2}}} \frac{\Gamma(\alpha + n/2)}{\Gamma(\alpha)} \tag{2.54}$$

where

$$\alpha = \frac{df}{2} \tag{2.55}$$

$$\beta = \frac{df s_0^2}{2} \tag{2.56}$$

The easiness of the marginalization of the noise parameters $\sigma_1^2, \sigma_2^2, \ldots$ allows us to fit other parameters alone without the fitting noise parameters.

2.7.3 Fitting Bayesian Additive Regression Trees

After p_i and q_i are determined in the way described in Subsection 2.7.1, the BART predictive function B_i in model (2.42) is optimized using a backfitting MCMC algorithm described in [9]. The R package *BayesTree* is applied to carry out the optimization process.

2.8 Results

2.8.1 In *Cryptococcus neoformans*

RNA-Seq data of *Cryptococcus neoformans* produced in Brent Lab and Doering Lab is used to measure the performance of the methods.

For data normalization, the pseudocount in (2.2) is set as $\nu = 5249$, which is the 0.5 percentile of nonzero fpkm's from all single deletion and wild type strains.

The explanatory variables here are the normalized expression levels of 224 regulators of *Cryptococcus neoformans*, as listed in Appendix E. The normalization follows the procedure described in Subsection 2.3.3.

Aside from these 224 regulator genes, the normalized (Subsection 2.3.3) expression levels of the other 6756 genes among the 6980 *Cryptococcus neoformans* genes studied are considered response variables.

Models were trained on the single deletion strain samples and the wild type samples and tested on the double deletion samples. Benchmarks of quantitative predictions include R^2, rmse, and correlation of $\log_2$ expression as defined in (2.4).

We first examined the performance of log-linear model (2.8) with lasso L^1 shrinkage determined by cross-validation. Logarithms of regulator levels and target levels are scaled to have standard deviation of 1 (i. e. using standard deviations as lasso normalizers). The prediction of $\log_2$-fold-change (with respect to wild type) reached accuracy of $R^2 \approx 0.51$.

To determine the importance of normalization of lasso variables, we also examined the performance of log-linear model without variable scaling and log-linear model with

NetProphet-type variable normalization (2.20) (2.22). Normalization seems to provide a great improvement in the predictive power (see Table 2.1).

Also tested is the log-linear model with global shrinkage parameter (introduced in [4], also see Subsection 2.5.3). Contrary to the improvement it provided on network edge recovery in the data used in [4], switching from local shrinkage to global shrinkage appeared to be counterproductive to the performance in expression prediction in this data:

Table 2.1: Influence of nuances in log-linear model on the predictive power

Normalizer	Standard deviation	NetProphet	None	Standard deviation
Shrinkage type	Local	Local	Local	Global
R^2	0.51	0.47	0.37	0.42

Aside from log-linear model, we tested the exponential model (2.10) and the log-linear-exponential model (2.11). Both are implemented with standard deviations as normalizers of lasso variables and with local L^1-shrinkage. The comparison between the performance of the log-linear model, the exponential model, and the log-linear exponential model is illustrated and discussed in Subsection 2.8.3.

Next, we applied BART-based semi-parametric model (2.42) and tested its performance. The tuning parameters p_i, q_i were fitted using steps in Subsection 2.7.1 and used to define the transformed variables U_{ik} in (2.29). Since the noises in the transformed expression levels U_{ik} are modeled as normal variables, we examined the similarity between

the distribution of centralized U_{ik} and the normal distribution. Define $\overline{U}_{ik}$ as the strain average U_{ik}: e.g., for the index set $strain_j$ of samples in the jth strain,

$$\forall k \in strain_j, \overline{U}_{ik} := \frac{1}{|strain_j|} \sum_{l \in strain_j} U_{il} \tag{2.57}$$

We compared the sample distribution of $U_{ik} - \overline{U}_{ik}$ and the normal distribution. Similarly, we can define the strain average $\overline{Y_{ik}}$ of Y_{ik} and the strain average $\overline{\log(Y_{ik})}$ of $\log(Y_{ik})$. Shapiro-Wilk normality test [20] was performed on $U_{ik} - \overline{U}_{ik}$, $Y_{ik} - \overline{Y}_{ik}$, and $\log(Y_{ik}) - \overline{\log(Y_{ik})}$ for each gene across strains in the training data and Lilliefors test [21] was performed on them across all genes and strains in the training data.

The distribution of $U_{ik} - \overline{U}_{ik}$ seems to show highest similarity to the normal distribution, slightly yet significantly (p-value $< 2.2 \times 10^{-16}$) higher than that of $\log(Y_{ik}) - \overline{\log(Y_{ik})}$ and much higher than that of $Y_{ik} - \overline{Y}_{ik}$ and $FPKM_{ik} - \overline{FPKM}_{ik}$

Table 2.2: Normality test on transformed expression levels

Variable	U	$\log(Y)$	FPKM	Y
Average S-W correlation per gene	0.928	0.926	0.870	0.857
Median S-W correlation per gene	0.943	0.941	0.930	0.913
Overall Lilliefors statistic	0.1264	0.1358	0.4249	0.3769

Since each gene is assigned with a Shapiro-Wilk correlation for each transformation, we can perform paired t-test between the Shapiro-Wilk correlation vectors of different transformations:

Table 2.3: Comparison of S-W correlation per gene

Candidates	U vs $\log(Y)$	$\log(Y)$ vs FPKM	FPKM vs Y
p-value of $A > B$	$< 2.2 \times 10^{-16}$	$< 2.2 \times 10^{-16}$	9.251×10^{-16}

We conclude that log-fold-change alone is capable of greatly increasing similarity with the normal distribution over FPKM and fold-change while $u(\cdot; p_i, q_i)$ provides even higher similarity than log-fold-change.

BART-based semi-parametric model showed robust and competitive performance. Its R^2, mse, $corr$, and comparison with other methods are illustrated and discussed in Subsection 2.8.3.

We also tested the performance of kNN-based expression prediction published in [10]. As the performance kNN depends on the parameter k, we explored options of $k = 1, 2, \ldots, 30$ and the correspondent R^2 at different k are listed in appendix B. The best performance is achieved at $k = 5$ applying kNN on log-fold-change (as opposed to fold change) of expression levels ($R \approx 0.26$).

2.8.2 In *Saccharomyces cerevisiae*

Micro-array data of *Saccharomyces cerevisiae* published in [2] is used to measure the performance of the methods. The published fold-change with respect to wild type is used as the raw data for this chapter, with an artificial vector included to represent wild-type expression profile, all of whose entries are 1.

We consider 320 *S. cerevisiae* genes as regulators (Appendix F). The majority of these regulators are transcription factors. The data set consists of 283 single deletion strains of regulator genes and 1201 single deletion strains of non-regulator genes. To test the contribution of regulator deletion strains to the predictive power of the model, we

performed 2-fold cross-validation between the set of regulator deletion strains and the set of non-regulation strains for each model.

We tested the performance of the log-linear model (2.8), the exponential model (2.10), and the log-linear-exponential model (2.11), all with standard deviations as lasso variable normalizers. Local L^1-shrinkage was used (see Subsection 2.5.3).

The BART-based semi-parametric model (2.42) was also tested. Note that for microarray data we set p_i, q_i in the noise model (2.29) to zero, i.e. modeling the noise of expression level with a log-normal distribution.

For comparison purposes, the kNN-based method published in [10] was also tested for performance. The parameter k of number of neighbors that we tried ranges from 1 to 30. Detailed R^2 information for different values of k is listed in Appendix B. Trained on regulator deletion strains and tested on non-regulator deletion strains, the best performance ($R^2 \approx 0.22$) is achieved applying kNN on log-fold-change with $k = 5$. Trained on non-regulator deletion strains and tested on regulator deletion strains, the best performance ($R^2 \approx 0.27$) is achieved applying kNN on fold change with $k = 6$.

The performance of these methods and their comparison can be found in the following subsection.

2.8.3 Comparison

We present the performance of the methods tested on *C. neoformans* and *S. cerevisiae*. In addition, we have taken the average of BART-based semi-parametric prediction and lasso-based log-linear prediction in hope to combined the strength of the two methods.

Table 2.4: Prediction accuracy of $\log_2$ target levels from regulator levels

	C. neoformans			*S. cerevisiae*, #1			*S. cerevisiae*, #2		
Training set	WT and singleΔ			RegulatorΔ			Non-regulatorΔ		
Test set	DoubleΔ			Non-regulatorΔ			RegulatorΔ		
	R^2	rmse	corr	R^2	rmse	corr	R^2	rmse	corr
log.kNN	0.26	0.88	0.54	0.22	0.15	0.48	0.24	0.17	0.50
linear.kNN	0.20	0.91	0.48	0.22	0.15	0.47	0.27	0.17	0.52
log.linear	0.51	0.71	0.72	0.51	0.12	0.71	0.37	0.16	0.69
exponential	0.40	0.79	0.63	0.53	0.11	0.73	0.42	0.15	0.71
log.linear.exp	0.44	0.76	0.69	0.54	0.11	0.74	0.44	0.15	0.67
BART	0.47	0.74	0.69	0.48	0.12	0.69	0.59	0.13	0.77
BART+log.linear	0.52	0.71	0.73	0.52	0.12	0.72	0.60	0.13	0.78

In contrary to the frustration that the developers of the kNN-based method [10] expressed about the failure of most model-based approaches in surpassing the performance of kNN in the DREAM3 challenge, we observe here that both the lasso-based models and the BART-based semi-parametric model surpass the performance of kNN by a great margin.

Log-linear model is most similar to Inferelator [15]. Although it is typical in many different methods and tasks to work with log-transformed expression levels, we find it inconclusive which one among the log-linear model (2.8), the exponential model (2.10), and the log-linear-exponential model (2.11) is the most desirable.

The BART-based model stands out to be the most robust model across all cases, although not always the one with best performance in every case. In *S. cerevisiae*, it appears that the performance of lasso-based models is hindered when being trained on the expression profiles of non-regulator deletion strains, while BART-based semiparametric model benefits from the abundance of data of non-regulator deletion strains (1201 strains, compared to 283 regulator deletion strains).

Finally, the combination of BART-based prediction and lasso-based log-linear prediction indeed provides a robust improvement.

The initial motivation in proposing the log-linear-exponential model (2.11) is the observation of the limitation of the log-linear model (2.8) in modeling repression and the numerical instability of the exponential model (2.10) in modeling activation (see Subsection 2.4). The log-linear-exponential model gives the option of circumventing such limitation by modeling repression using its exponential module and modeling activation using its log-linear module. This distribution of functionality is not *a priori* enforced or encouraged. Instead, we trained the model and looked into the learned parameters and confirmed that the log-linear module tends to have more positive coefficients (representing activation) and that the exponential module tends to have more negative coefficients (representing repression).

For each target gene i, we extract all its nonzero coefficients in the log-linear module, calculating in which the fractions of positive and negative coefficients:

$$P_i^{(lln)+} = \frac{|\{j : c_{ij} > 0\}|}{|\{j : c_{ij} \neq 0\}|} \tag{2.58}$$

$$P_i^{(lln)-} = \frac{|\{j : c_{ij} < 0\}|}{|\{j : c_{ij} \neq 0\}|} \tag{2.59}$$

where the notation of parameters follow that of (2.11). Similarly, we define the fractions of positive and negative coefficients among the nonzero coefficients of the exponential module for each target gene:

$$P_i^{(exp)+} = \frac{|\{j : d_{ij} > 0\}|}{|\{j : d_{ij} \neq 0\}|} \tag{2.60}$$

$$P_i^{(exp)-} = \frac{|\{j : d_{ij} < 0\}|}{|\{j : d_{ij} \neq 0\}|} \tag{2.61}$$

Also, we calculated the overall fraction of positive (and negative) coefficients of each module across all genes:

$$Q^{(lln)+} = \frac{|\{(i,j) : c_{ij} > 0\}|}{|\{(i,j) : c_{ij} \neq 0\}|} \tag{2.62}$$

$$Q^{(exp)+} = \frac{|\{(i,j) : d_{ij} > 0\}|}{|\{(i,j) : d_{ij} \neq 0\}|} \tag{2.63}$$

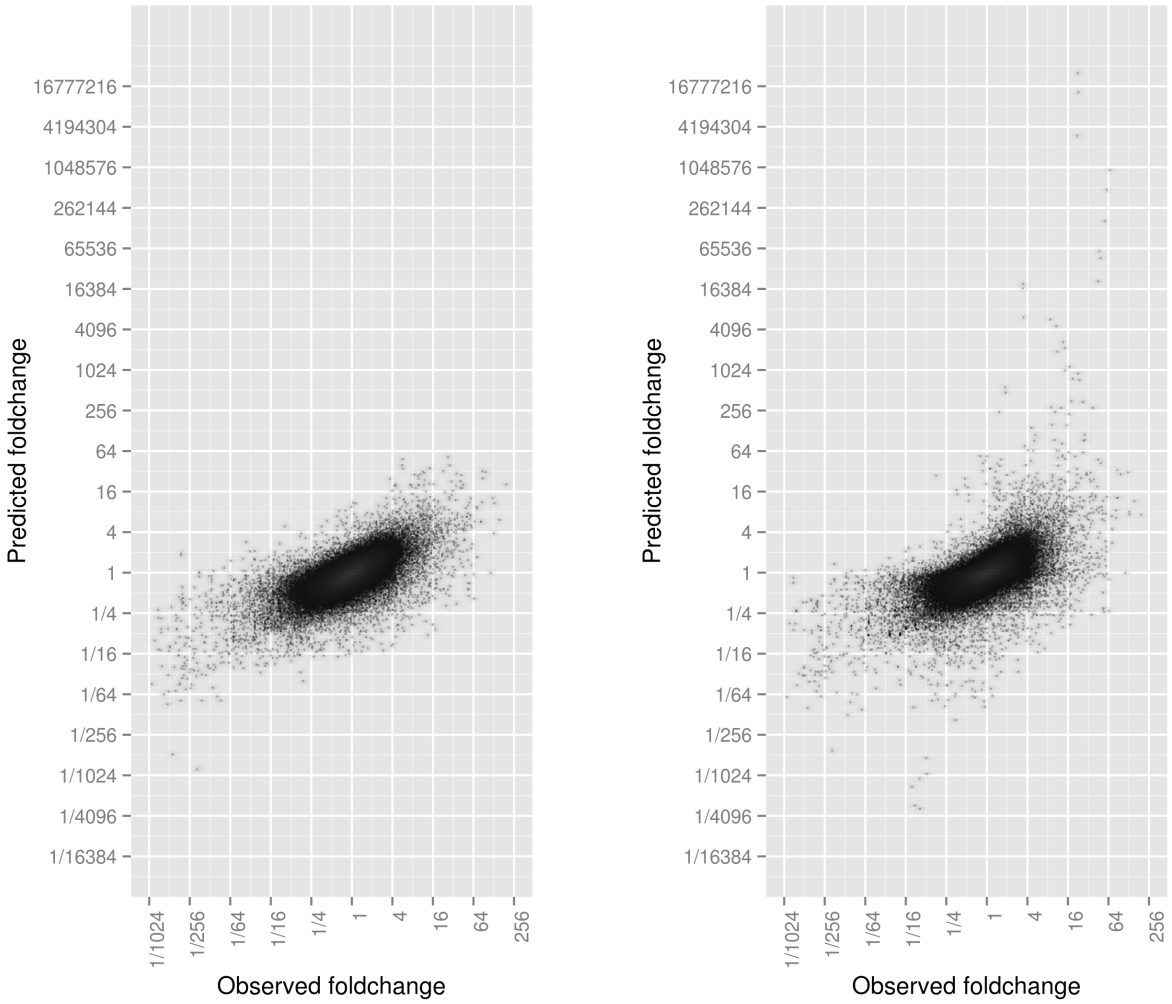
By directly comparing $Q^{(lln+)}$ and $Q^{(exp+)}$ and by performing a paired t-test between $P^{(lln)+}$ and $P^{(exp)+}$, we found that the log-linear module has slightly yet significantly stronger tendency to have positive coefficients than the exponential module:

Table 2.5: Distribution of coefficient signs in log-linear-exponential model

Training data	*C. neoformans*	*S. cerevisiae*, #1 RegulatorΔ	*S. cerevisiae*, #2 Non-regulatorΔ
Overall % of positive log-lin coeff	56.28%	50.41%	55.83%
Overall % of positive exp coeff	47.59%	49.72%	50.28%
p-value of difference	$< 2.2 \times 10^{-16}$	0.005696	0.02281

We observed a tendency of over-extrapolation in the prediction produced by exponential models (and also, but less severely, by log-linear exponential models) for certain expression levels higher than the wild type. In contrast, these over-extrapolation points were not observed in BART-based prediction log-linear model prediction. In *C. neoformans*, for example, we notice these points in the upper-right corner of the prediction-observation plot, where the predicted levels are much higher than the observed levels in the test data:

Figure 2.1.: Comparison of BART-based prediction and exponential model prediction in *C. neoformans*



(a) BART-based semi-parametric model

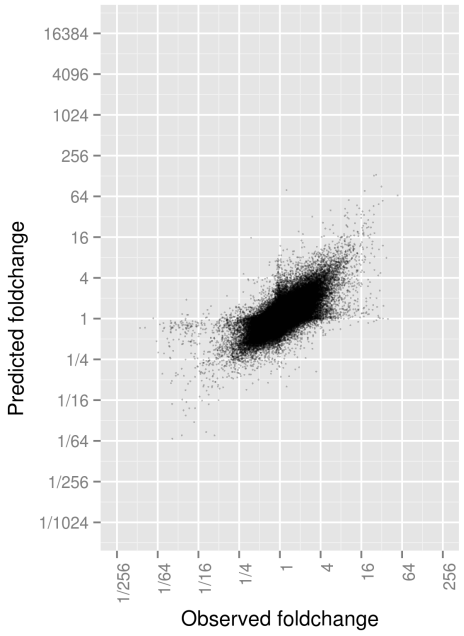
(b) Exponential model

This tendency of over-extrapolation does not fall out from our expectation, for 1) the exponential model is transformable to a linear model, which inherently extrapolates unboundly whereas actual gene expression levels may have a maximum and 2) we have

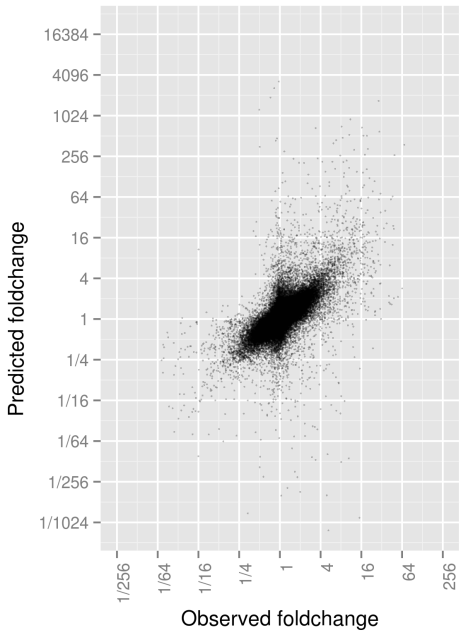
observed in Subsection 2.4 that exponential modeling of activation induces instability. On the other hand, BART is an ensemble of decision trees, which do not attempt to extrapolate when encountered with a query value of an explanatory variable (regulator level) outside its range observed in the training data.

In *S. cerevisiae*, over-extrapolation is more common in models trained on the non-regulator deletion strains than in those trained on regulation deletion strains:

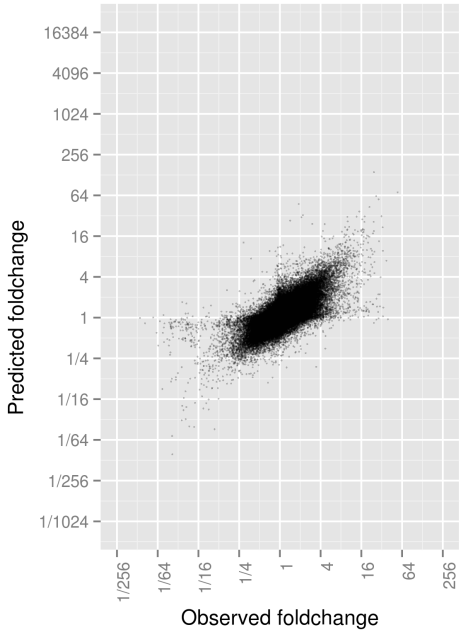
Figure 2.2.: Comparison of *S. cerevisiae* target levels predicted from different data sets by the exponential model and the log-linear-exponential model



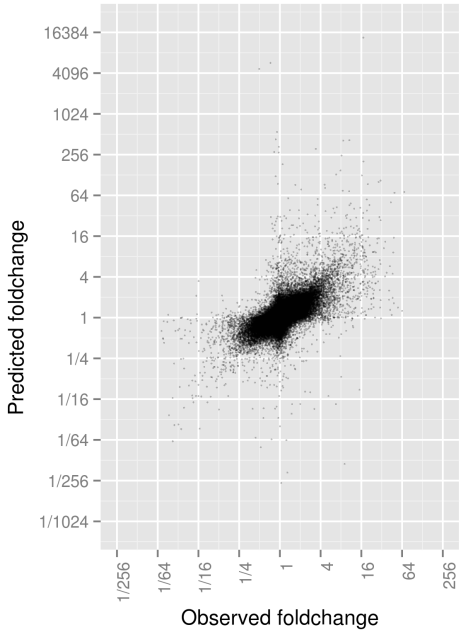
(a) Exponential model trained on regulatorΔ, tested non-regulatorΔ



(b) Exponential model trained on non-regulatorΔ, tested regulatorΔ



(c) Log-linear-exponential model trained on regulatorΔ, tested non-regulatorΔ



(d) Log-linear-exponential model trained on non-regulatorΔ, tested regulatorΔ

Direct artificial perturbation of regulators provides a wider variation range of regulator levels, which might account for the absence of over-extrapolation of models trained on the regulator deletion strains.

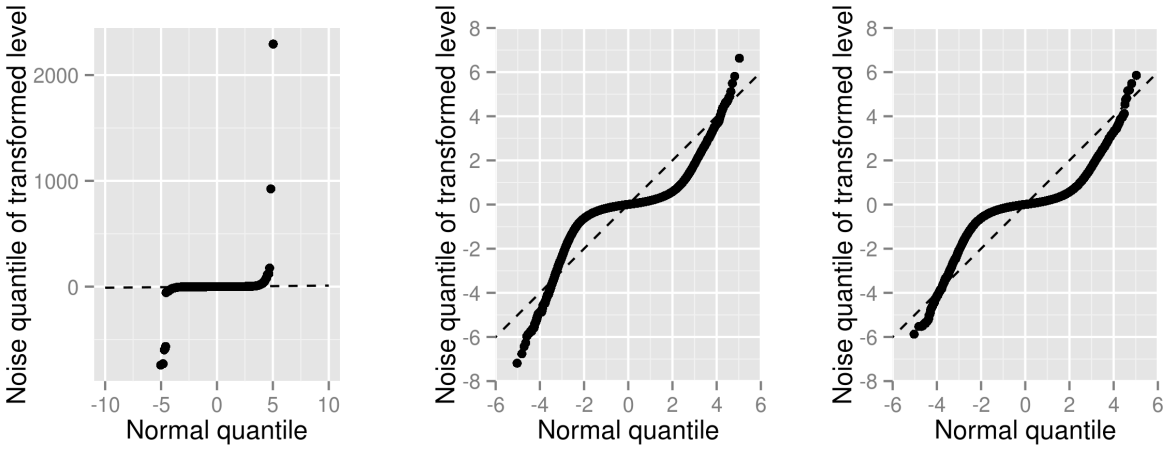
In contrast to the simple parametric methods, BART-based semi-parametric models are proof from over-extrapolation whether trained on regulator deletion strains or on non-regulator strains.

2.9 Discussion

2.9.1 Further Improvement of the Noise Model

In Subsection 2.29, we have successfully developed a transformation function (2.29) that converts expression levels closer to a normal variable than a logarithm transformation is capable of. But we find the similarity between the distribution of noises of the transformed expression levels and the normal distribution not yet satisfying, as shown by the normal quantile-quantile plot of noises in the training data of *C. neoformans*:

Figure 2.3.: Normal Q-Q plot of transformed expression noises



(a) Fold-change (b) Log-fold-change (c) Novel transform (2.29)

We propose some potential improvements in the transformation function: 1) Instead of the linear fitting method for tuning transformation parameters p_i and q_i in Subsection 2.7.1, p_i and q_i maybe tuned to optimize certain indicators of normality of the transformed noise, e. g. Shapiro-Wilk correlation [20]; 2) in addition to the transformation $u(\cdot; p_i, q_i)$ in (2.29) whose shape is similar to logarithm transformation, transformations with vastly different shapes may be tried; 3) an empirical transformation may be learned from the relation between expression levels and noise levels.

2.9.2 Non-lasso Type Parametric Models

As indicated by Table 2.5, log-linear modeling is more capable of depicting activation effects and exponential modeling more capable of depicting repression effects. A modi-

fication of the log-linear-expression model (2.11) that reduces the number of parameters may be considered:

$$F_i^{(llaer)}(x_1, x_2, \ldots; \theta) = b_i \left(\prod_{j:c_{ij}>0} (x_j)^{c_{ij}} \right) e^{\left(\sum_{j:c_{ij}<0} c_{ij} x_j \right)} \tag{2.64}$$

We refer to this modified model as *log-linear-activation-exponential-repression* (LLAER). It enforces all activation effects to be modeled log-linearly and all repression effects exponentially. With these restrictions, the optimization of this model can no longer be formulated as a lasso regression problem. However, this model has fewer parameters than the log-linear-exponential model and we consider it a possibility that this model might provide more robustness than the log-linear-exponential model.

It has been observed [7, 19] that the basal transcription level of a gene in absence of regulators is close to zero due to repression by histones and that the presence of activators is required to release such repression and to allow transcription. In Section 2.4, we have reasoned that the exponential model (2.10) would introduce numerical instability when describing this observation. The Log-linear-exponential model and the LLAER model however also have their share of limitation in modeling this observation, for they require all activators of a gene to be nonzero to allow transcription. We consider another modeling approach that allows transcription of a gene in the presence of some but not necessarily all of its activators:

$$F_i^{(aaer)}(x_1, x_2, \ldots; \theta) = b_i \left(\sum_{j:c_{ij}>0} (c_{ij} x_j) \right) e^{\left(\sum_{j:c_{ij}<0} (c_{ij} x_j) \right)} \tag{2.65}$$

We refer to this model as *additive-activation-exponential-repression* (AAER). This optimization of this model cannot be formulated as a lasso regression problem either and requires development of *ad hoc* preventive measures against overfitting. This model by definition implies a strong restriction on the sensitivity of a gene to the levels of its activators. Note that

$$\forall j : c_{ij} > 0, \frac{\partial \ln\left(F_i^{(aaer)}(x_1, x_2, \ldots; \theta)\right)}{\partial \ln(x_j)} = \frac{c_{ij}x_j}{\sum\limits_{j:c_{ij}>0}(c_{ij}x_j)} \tag{2.66}$$

which implies that

$$\sum_{j:c_{ij}>0} \frac{\partial \ln\left(F_i^{(aaer)}\right)}{\partial \ln(x_j)} = 1 \tag{2.67}$$

$$\forall j : c_{ij} > 0, \frac{\partial \ln\left(F_i^{(aaer)}\right)}{\partial \ln(x_j)} \leq 1 \tag{2.68}$$

The log-scale sensitivity of target level to activator levels is implied to be less than or equal to 1 and should add up to one. It would be interesting to investigate whether this strong consequential restriction is supported or undermined by experimental data.

2.9.3 Relating Sequence Affinity to Gene Expression

Successful efforts have been seen in predicting expression levels of transcription factor levels with the aid of knowledge of protein-DNA binding affinity. In [11], protein-DNA binding affinity were calculated through sequence analysis and used for modeling protein-DNA interaction mechanism and expression regulation. With sequence affinity in-

formation and transcription factor (TF) levels as inputs, [11] was able to correctly predict the expression patterns of certain genes in the segmentation gene network of *Drosophila melanogaster*.

We propose here a modification of the model in [11] that greatly reduces the time complexity of calculation.

The sequence selectivity of a TF is conventionally represented by position-specific weight matrix (PWM), position-specific probability matrix (PPM), position-specific scoring matrix (PSSM), et cetera introduced in [17]. Over the decade, these representations have been shown to accurately depict the affinity between TFs and DNA sequences.

A PPM of a TF takes the form a $4 \times L$ matrix:

$$PPM = \begin{pmatrix} PPM_{11} & PPM_{12} & \dots \\ PPM_{21} & PPM_{22} & \dots \\ PPM_{31} & PPM_{32} & \dots \\ PPM_{41} & PPM_{42} & \dots \end{pmatrix} \begin{matrix} A \\ C \\ G \\ T \end{matrix} \tag{2.69}$$

All entries are non-negative and the column sums of the matrix are 1; PPM_{ij} can be understood as the proportion of nucleobase i (A, C, G, T) appearing at the jth position of the binding site of the concerned TF. PPM is intended for the calculation of the affinity between the TF and an arbitrary sequence s of length L on the DNA by the product:

$$\omega(PPM, s) = \prod_{i=1}^{L} PPM_{s_i i} \tag{2.70}$$

where s_i is represents the base on the ith location of the sequence, with the code

$$\begin{cases} 1 & A \\ 2 & C \\ 3 & G \\ 4 & T \end{cases} \tag{2.71}$$

since each sequence s on a DNA strand has its reverse complement on the other strand $\tilde{s}$, the affinity between the TF and the location of s may be represented by:

$$P_{TF}(s) = \frac{\omega(PPM^{(TF)}, s) + \omega(PPM^{(TF)}, \tilde{s})}{2} \tag{2.72}$$

We refer to this term as the PPM score of the given position and the given TF.

In addition to PPMs of TFs, a background PPM is defined to represent the random chance of bases appearing in the DNA sequence. Denote the number of bases A, C, G, T in the DNA of a certain species as n_A, n_C, n_G, n_T respectively,

$$PPM^{(bg)} = \frac{1}{n_A + n_C + n_G + n_T} \begin{pmatrix} n_A \\ n_C \\ n_G \\ n_T \end{pmatrix} \tag{2.73}$$

The background PPM score of a position on the DNA provide a baseline that one may compare TF PPM scores with.

PWM is defined as

$$PWM = \begin{pmatrix} \log\left(\frac{PPM_{11}}{PPM_{11}^{(bg)}}\right) & \log\left(\frac{PPM_{12}}{PPM_{11}^{(bg)}}\right) & \dots \\ \log\left(\frac{PPM_{21}}{PPM_{21}^{(bg)}}\right) & \log\left(\frac{PPM_{22}}{PPM_{21}^{(bg)}}\right) & \dots \\ \log\left(\frac{PPM_{31}}{PPM_{31}^{(bg)}}\right) & \log\left(\frac{PPM_{32}}{PPM_{31}^{(bg)}}\right) & \dots \\ \log\left(\frac{PPM_{41}}{PPM_{41}^{(bg)}}\right) & \log\left(\frac{PPM_{42}}{PPM_{41}^{(bg)}}\right) & \dots \end{pmatrix} \begin{matrix} A \\ C \\ G \\ T \end{matrix} \tag{2.74}$$

and the PWM score of a sequence s is defined as

$$\phi(PWM, s) = \sum_{i=1}^{L} PWM_{s_i i} \tag{2.75}$$

The relation between the PWM score and the PPM score (2.70) is

$$\phi(PWM, s) = \log(\omega(PPM, s)) - L\log(\omega(PPM^{(bg)}, s)) \tag{2.76}$$

The PWM and the PPM of a TF can be statistically inferred from experimental data of protein-DNA binding locations. These matrices have been calculated for many TFs in various model species (especially *S. cerevisiae*) and are available in the literature.

To relate binding affinity to gene expression, the work in [11] simplifies the interaction between transcription factors (TF) and promoter sequences of genes with several assumptions:

1) When a TF is binding to a location on the DNA, the recognition site, or the span of its PPM aligned on the sequence, is considered identical to the physical area that the TF occupies on the sequence.

2) In general, two TFs cannot occupy overlapping areas on the sequence at the same time (the authors also introduced a more complicated model involving cooperative binding which may or may not make an exception to this assumption; modeling cooperative binding will not be in the scope of this discussion).

3) The Boltzmann weight of an TF binding to a certain position on a sequence is proportional to the product of the amount of that TF in the cell and to the exponential of the PWM score of that position:

$$W(\,TF\text{-}s) \propto [\,TF] \exp(\phi(PWM^{(TF)}, s)) \tag{2.77}$$

4) Given the number of molecules of each TF species bound on a promoter sequence of a gene, the transcription rate of the gene is determined by a logistic transformation of a linear combination of the numbers of the TFs, with each coefficient representing activation (positive signed) or repression (negative signed) and depending on TF species but not on the identity of the gene:

$$\frac{1}{1 + \exp\left(-w_0 - \sum_j w_j n_{TF_j}\right)} \tag{2.78}$$

where n_{TF_j} is the number of molecules of TF species j bound on the promoter.

With these assumptions, the authors refer to the locations of all TFs bound to a promoter sequence as a "configuration" and are able to represent The Boltzmann weight of each configuration with an expression involving TF levels, unknown parameters, and PWM scores. Denoting the set of the binding locations of TF species j in configuration *config* as $S_j(\textit{config})$, the Boltzmann weight of *config* given TF levels $\vec{x} = (x_1, x_2, \ldots)^T$ is

$$W(\textit{config} \mid \vec{x}) = \begin{cases} 0 \text{ - if the binding areas of any two TFs overlap in } \textit{config} \\ \prod_j \prod_{s \in S_j(\textit{config})} \tau_j x_j \exp(\phi(PWM^{(TF_j)}, s)) & \text{otherwise} \end{cases} \tag{2.79}$$

where τ_j is a proportionality parameter.

In thermodynamic modeling, the Boltzmann weight of a configuration is proportional to its probability of occurring. For a given profile of TF levels and values of parameters, the distribution of configurations may be numerically estimated by (2.79). Given a configuration (and the choice of parameter values), the transcription rate can be determined; the distribution of configurations and the values of parameters will determine the expectation of transcription rate, which is proportional to the expression level of the gene.

The estimation of configuration distribution from Boltzmann weights (or the expectation of a variable in this probability space - e.g. transcription rate) involves calculating the sum of Boltzmann weights of all configurations and sampling from the universe of configurations. The authors discovered that the sum of Boltzmann weights can be calcu-

lated efficiently by formulating the problem as a generalized Hidden Markov-chain Model (gHMM) and invoking a dynamic programming algorithm of time complexity $\mathcal{O}$((number of TF species) $\times$ (length of promoter sequence)). However, the calculation of expected transcription rate requires sampling from the probability space of configurations and calculating the amount (2.78) and may cause a time complexity of $\gg \mathcal{O}$((number of TF species) $\times$ (length of promoter sequence)).

Here, we propose a modification to the assumptions in [11] and simplify the calculation of the expression level for given values of parameters and TF levels to $\mathcal{O}$((number of TF species) $\times$ (length of promoter sequence)) while completely avoiding the cost of sampling from configurations, providing a significant speed up.

We include RNA polymerase (RNAP) in the picture of thermodynamic modeling, considering RNAP along with all TFs as a DNA-binding protein (or protein complex). The Boltzmann weight of a configuration with RNAP bound and one with RNAP unbound can both be represented. In addition, we assume that the expression level is proportional to the probability of RNAP bound to the sequence:

$$W(\textit{config} \mid \vec{x}) = \tag{2.80}$$

$$\begin{cases} 0 \text{ - if the binding areas of any two TFs overlap in } \textit{config} \\ \prod_j \prod_{s \in S_j(\textit{config})} \tau_j x_j \exp(\phi(PWM^{(TF_j)}, s)) \text{ - no overlap, RNAP not bound in } \textit{config} \\ \rho_i \prod_j \prod_{s \in S_j(\textit{config})} \omega_j \tau_j x_j \exp(\phi(PWM^{(TF_j)}, s)) \text{ - no overlap, RNAP bound in } \textit{config} \end{cases} \tag{2.81}$$

where parameter ω_j describes the affinity between TF_j and RNAP and parameter ρ_i describes the affinity between the promoter of gene i and RNAP. ω_j only depends on the identity j of the TF and ρ_i only depends on the identity i of the target gene. A value of ω_j smaller than 1 represents repression and a value larger than 1 represents activation.

We define Z^{on} as the sum of the Boltzmann weights of all the configurations with RNAP bound to the promoter and Z^{off} as the sum of the Boltzmann weights of the configurations without RNAP bound. Hence the expression level is

$$F_i(x_1, x_2, \ldots) = \frac{\alpha_i Z^{on}}{Z^{on} + Z^{off}} \tag{2.82}$$

where α_i is a proportionality parameter.

Both Z^{on} and Z^{off} can be calculated using the gHMM formulation and dynamic programming approach discussed in [11] with time complexity of $\mathcal{O}$((number of TF species) $\times$ (length of promoter sequence)).

If we define a function

$$Q_{config}(\xi_1, \xi_2, \ldots) = \begin{cases} 0 \text{ - if the binding areas of any two TFs overlap in } \textit{config} \\ \prod_j \prod_{s \in S_j(config)} \xi_j \exp(\phi(PWM^{(TF_j)}, s)) \text{ - otherwise} \end{cases} \tag{2.83}$$

then by (2.78), the expression level in the original model [11] can be written as

$$\frac{\alpha_i \sum_{config} \frac{Q_{config}(\tau_1 x_1, \tau_2 x_2, \ldots)}{1+\exp\left(-w_0 - \sum_j w_j |S_j(config)|\right)}}{\sum_{config} Q_{config}(\tau_1 x_1, \tau_2 x_2, \ldots)} \tag{2.84}$$

while the expression level in our modified model is

$$\frac{\alpha_i}{1+\frac{\sum\limits_{config} Q_{config}(\tau_1 x_1,\tau_2 x_2,\ldots)}{\rho_i \sum\limits_{config} Q_{config}(\omega_1\tau_1 x_1,\omega_2\tau_2 x_2,\ldots)}} \tag{2.85}$$

Our model completely avoids sampling from the the probability space of configurations, yet does not add significantly to the number of parameters in [11].

Neither the sequence affinity model [11] nor our modification of it in (2.81) considers the effect of histones on gene transcription. It has been observed [7, 19] that the basal transcription level of a gene in absence of regulators is close to zero due to repression by histone proteins in nucleosomes. But on the contrary, [8] observed that the inclusion of histones in the thermodynamic picture of sequence affinity analysis is counter-productive for the task of predicting ChIP-evidenced binding sites. It is questionable whether the repression by histones has the same quantitative features as the repression by a transcription factor.

A simple way to include the repression effect of histones in the model based on sequence affinity is to make an exception to the Boltzmann weight definition (2.81) by stating that

the weight of an TF-less promoter-RNAP complex is zero or an unknown parameter $\tilde{\rho}_i$ close to zero:

$$W(\mathit{config}, \vec{x}) = \begin{cases} 0 \text{ - if the binding areas of any two TFs overlap in } \mathit{config} \\ \tilde{\rho}_i (\approx 0) \text{ - RNAP bound, but no TF bound in } \mathit{config} \\ \prod_j \prod_{s \in S_j(\mathit{config})} \tau_j x_j \exp(\phi(PWM^{(TF_j)}, s)) \\ \text{ - no overlap, RNAP not bound in } \mathit{config} \\ \rho_i \prod_j \prod_{s \in S_j(\mathit{config})} \omega_j \tau_j x_j \exp(\phi(PWM^{(TF_j)}, s)) \\ \text{ - no overlap, RNAP bound, at least one TF bound in } \mathit{config} \end{cases} \tag{2.86}$$

This modification do not add to the time complexity of (2.81). It would be interesting to test whether (2.86) or the histone-free model (2.81) better explains the variance in the experimental data.

2.9.4 Spectrum between Simple Regression Models and Sequence-based Models

In (2.81), we have made a modification to the sequence-based model (2.79) defined in [11] which would greatly reduce time complexity. Yet, the resulting time complexity is proportional to the length of promoter sequences and several magnitudes higher than that of simple regression models (2.8), (2.10), and (2.11). These highly detailed sequence-

thermodynamic models are viable for sub-networks involving only a small number (e.g. < 10) of regulators, but their extension to genome-wide analysis might be difficult under current technology even with the aid of parallel computing.

Some intermediates between simple regression models and sequence based models are possible and may provide both frugal time complexity and the ability to borrow from the knowledge of sequence affinity.

In [8], it has been observed that the modeling of TF-TF interactions such as competition and cooperativity or even simply disallowing overlapping binding is generally counter-productive to the task of predicting ChIP evidenced binding sites. If we infer from this the hypothesis that a location on a promoter sequence has much larger chance of not being bound by any TF than otherwise (sparse binding), the interaction between TFs is omittable and the Boltzmann weight (2.81) can be approximated by a more simple form

$$W(\mathit{config}, \vec{x}) = \begin{cases} \prod_j \prod_{j \in S_j(\mathit{config})} \frac{\tau_j x_j P_{TF_j}(s)}{\left(P_{bg}(s)\right)^L} & \text{RNAP not bound in } \mathit{config} \\ \rho_i \prod_j \prod_{j \in S_j(\mathit{config})} \frac{\omega_j \tau_j x_j P_{TF_j}(s)}{\left(P_{bg}(s)\right)^L} & \text{RNAP bound in } \mathit{config} \end{cases} \tag{2.87}$$

where $S_j(\mathit{config})$ is the set all locations on the promoter bound by TF species j. The weight sum of all RNAP-bound configurations as well as that of RNAP-unbound configurations is simplified to, by straightforward factoring:

$$Z^{(\mathit{off})} = \prod_{j,s} \left(1 + \frac{\tau_j x_j P_{TF_j}(s)}{(P_{bg}(s))^L}\right) \tag{2.88}$$

$$Z^{(\mathit{on})} = \rho_i \prod_{j,s} \left(1 + \frac{\omega_j \tau_j x_j P_{TF_j}(s)}{(P_{bg}(s))^L}\right) \tag{2.89}$$

We may define a polynomial for each target i and TF j:

$$P_{ij}(\xi) = \prod_{s\text{: a location on promoter } i} \left(1 + \frac{\xi P_{TF_j}(s)}{(P_{bg}(s))^L}\right) \tag{2.90}$$

The coefficients of these polynomials can be pre-calculated from known sequence affinity information without dependence on parameters or regulator levels. Applying these polynomials,

$$Z^{(\mathit{off})} = \prod_j P_{ij}(\tau_j x_j) \tag{2.91}$$

$$Z^{(\mathit{on})} = \rho_i \prod_j P_{ij}(\omega_j \tau_j x_j) \tag{2.92}$$

$$\tag{2.93}$$

and the expression level of target i is

$$F^{(simp)}(x_1, x_2, \ldots; \text{parameters}) = \frac{\alpha_i Z^{on}}{Z^{on} + Z^{off}} \tag{2.94}$$

$$= \frac{\alpha_i}{1 + \frac{1}{\rho_i \prod_j \frac{P_{ij}(\omega_j \tau_j x_j)}{P_{ij}(\tau_j x_j)}}} \tag{2.95}$$

Similarly, the model with histone consideration (2.86) can be simplified to

$$Z^{(off)} = \prod_j P_{ij}(\tau_j x_j) \tag{2.96}$$

$$Z^{(on)} = \rho_i \left(\left(\prod_j P_{ij}(\omega_j \tau_j x_j) \right) - 1 \right) + \tilde{\rho}_i \tag{2.97}$$

$$F^{(simp.hist)}(x_1, x_2, \ldots; \text{parameters}) = \frac{\alpha_i}{1 + \frac{\prod_j P_{ij}(\tau_j x_j)}{\left(\rho_i \prod_j P_{ij}(\omega_j \tau_j x_j)\right) - \rho_i + \tilde{\rho}_i}} \tag{2.98}$$

If we approximate P_{ij} by only the first D terms in its Taylor expansion, then $Z^{(on)}$, $Z^{(off)}$, $F^{(simp)}$, and $F^{(simp.hist)}$ can be calculated with time complexity of $\mathcal{O}((\text{number of TF species}) \times D)$ for given values of parameters and TF levels. In particular if $D = 2$, the approximated form of $F^{(simp)}$ simplifies to a special case of the thermodynamic model (2.12).

3. Predicting Gene Expression from Genotype

3.1 Background

The goal of this chapter is to develop a method to predict steady state expression levels of genes from genetic perturbation information.

This goal is comparable to, yet in contrast with, that of Chapter 2: in Chapter 2, regulator levels are treated as explanatory variables and are revealed in both training and query data, while target gene levels are treated as response variables with random noise. In this chapter, only perturbation information is treated as explanatory variables and we aim to predict the expression levels of all genes, including regulators and non-regulators, from perturbation information.

The ability to predict expression levels from perturbation information can give insight to the functional dependence and logic between the regulatory roles of genes, aid the prediction of phenotypes from genotypes, and provide suggestions to genetic engineering tasks that aim to shift gene expressions and metabolism to a target state.

3.2 Previous Works

Prediction of gene expressions from genotype is a less charted territory in bioinformatics.

In [13], a model based on linear decomposition was used to predict *Saccharomyces cerevisiae* expression levels from gene deletion information. The training data there is composed of the single deletion strains of five regulator genes (*TEC1*, *CUP9*, *SFL1*, *SOK2*, *SKN7*) and all 10 possible double deletion strains of these genes. These five genes were chosen for their involvement in the regulation of filamentous growth. The test data contains the the single deletions train *yap6*Δ and double deletion strains *yap6*Δ*tec1*Δ, *yap6*Δ*cup9*Δ, *yap6*Δ*sfl1*Δ, *yap6*Δ*sok2*Δ, and *yap6*Δ*skn7*Δ.

The accuracy in R^2 or MSE was not given; the method was able to predict the direction of expression change in certain cases in the test data. However, the model is difficult to be generalized to cases where the training data contains few or no double deletion strains (see equation (A.3)) or if the knowledge of the regulatory pathways of the studied species is not comprehensive.

3.3 Data and Evaluation Criteria

3.3.1 Data Type and Interpretation

The Same as in Chapter 2, we work with high-throughput expression data such as RNA-Seq and micro-array. Yet different from Chapter 2, here we only use perturbation information as explanatory variables. Perturbation information consists of two aspects: 1) whether a gene is artificially perturbed (e.g. deleted, underexpressed, or overexpressed)

in a certain sample and 2) the perturbed expression level of a gene in a certain sample where it is artificially perturbed (e.g. 0 if deleted). To formalize, denote:

$$P_{train} = \{(i,k) \in G \times T : \text{gene } i \text{ is artifcially perturbed in sample } k \} \tag{3.1}$$

$$P_{test} = \{(i,k) \in G \times S : \text{gene } i \text{ is artifcially perturbed in sample } k \} \tag{3.2}$$

$$P = P_{train} \cup P_{test} \tag{3.3}$$

$$y_{ik}^{(pert)} = \begin{cases} Y_{ik} & \text{if } (i,k) \in P \\ 1 & \text{otherwise} \end{cases} \tag{3.4}$$

where G is the index set of investigated genes, T is the index set of training samples, S is the index set of test samples, and Y_{ik} is the expression level of gene i in sample k.

Since the artificial perturbation of a transcription factor might affect not only its direct target genes but also the downstream targets of its targets, the prediction problem in this chapter requires a model that captures the propagation of the effect of genetic perturbation in the network.

3.3.2 Training and Test Data

The method is trained and tested on RNA-Seq data of *Cryptococcus neoformans* produced in Brent Lab (http://mblab.wustl.edu/) and Doering Lab (http://www.crypto.wustl.edu/). The expression profiles of all single-gene deletion samples and wild type samples are used as training data and the expression profiles of all double-gene deletion strains as test data.

3.3.3 Normalization of RNA-Seq Data

The RNA-Seq data used in this chapter is normalized in the same manner described in Subsection 2.3.3. In particular, the normalized expression level gene i in sample k from batch j is defined as in (2.2):

$$\forall k \in batch_j, Y_{ik} = \frac{FPKM_{ik} + \nu}{\frac{\sum_{l \in batch_j \cap WT} FPKM_{il}}{|batch_j \cap WT|} + \nu} \tag{2.2}$$

where $batch_j$ is the index set of samples in the jth batch, WT is the index set of wild type samples, $FPKM_{il}$ is the number of fragments per kilobase per million reads of gene i in sample l, and ν is a small pseudocount to ensure that the fraction does not evaluate to ∞ or $0/0$.

This definition uses wild type expression levels as normalizers and Y_{ik} can also be interpreted as the fold change of expression level with respect to wild type.

3.3.4 Assessment of Quantitative Prediction

The quantitative precision of the predictions is measured in the same way as in subsections 2.3.4. However, if a gene is artificially perturbed in a sample (deleted, overexpressed, etc), it is not the task of this chapter to predict its level there.

Using the notations in (3.3), the performance measures of quantitative predictions are:

$$\begin{cases} MSE = \frac{1}{|G\times S-P|} \sum\limits_{(i,k)\in G\times S-P} \left(\log_2 \hat{Y}_{ik} - \log_2 Y_{ik}\right)^2 \\ rmse = \sqrt{MSE} \\ R^2 = 1 - \frac{MSE}{s^2_{G\times S-P}(\log_2 Y)} \\ Corr = \frac{\sum\limits_{(i,k)\in G\times S-P} \left((\log_2 \hat{Y}_{ik})(\log_2 Y_{ik})\right)}{\sqrt{\sum\limits_{(i,k)\in G\times S-P} \left(\log_2 \hat{Y}_{ik}\right)^2 \sum\limits_{(i,k)\in G\times S-P} (\log_2 Y_{ik})^2}} \end{cases} \tag{3.5}$$

where

$$s^2_{G\times S-P}(\log_2 Y) = \frac{\sum\limits_{(i,k)\in G\times S-P} (\log_2 Y_{ik})^2}{|G\times S-P|-1} - \frac{\left(\sum\limits_{(i,k)\in G\times S-P} \log_2 Y_{ik}\right)^2}{(|G\times S-P|)(|G\times S-P|-1)} \tag{3.6}$$

is the sample variance of $\log_2(Y)$.

3.3.5 Assessment of Qualitative Prediction

Another interesting question is how well the significant differential expressions in the novel genotypes can be revealed by the prediction. Here, differential expression (DE) of a gene is defined as a level significantly different from its wild type level.

The assessment requires a DE score that is credible enough to be used as the gold standard (true label) on the test data and another DE score drawn from the prediction of the model trained on the training data. In addition, we are interested in how well the direction of differential expression (higher or lower than wild type) can be predicted.

The discovery of differential expressions can be formulated as two binary classification problems: 1) discovery of upward differential expression, with prediction score s^+ as a continuous classifier with variable discrimination threshold and L^+ as the true labels describing whether the genes are expressed significantly higher than in wild type; 2) discovery of downward differential expression, with prediction score s^- as a continuous classifier with variable discrimination threshold and L^- as the true labels describing whether the genes are expressed significantly lower than in wild type.

The gold standard L^+ and L^- is obtained by applying certain published DE evaluation algorithms on the test data. E.g. voom [3] for RNA-Seq and LIMMA [16] for micro-array are both well accepted methods for evaluating the significance of differential expressions. These methods calculate for each gene in each sample a p-value representing the statistical significance of differential expression and a number representing estimated fold change with respect to wild type. Denote the p-value assigned to gene i in sample k as p_{ik} and the estimated fold change as m_{ik}. A reasonable significance threshold α_0 (e.g. 0.001) is chosen and the true labels are defined as:

$$L^+_{ik} = \begin{cases} 1 & p_{ik} < \alpha_0 \wedge m_{ik} > 1 \\ 0 & \text{otherwise} \end{cases} \tag{3.7}$$

$$L^-_{ik} = \begin{cases} 1 & p_{ik} < \alpha_0 \wedge m_{ik} < 1 \\ 0 & \text{otherwise} \end{cases} \tag{3.8}$$

The continuous classifiers s^+ and s^- are drawn from the prediction.

For Bayesian prediction methods, they are the posterior probability of the predicative being larger (or conversely, smaller) than the ideal wild type level 1:

$$\begin{cases} s_{ik}^{+} = P(Y_{ik} > 1 \mid training.data, model) \\ \\ s_{ik}^{-} = P(Y_{ik} < 1 \mid training.data, model) \end{cases} \tag{3.9}$$

For maximum likelihood prediction without Bayesian interpretation, the predicted log-fold-change is used as the score:

$$\begin{cases} s_{ik}^{+} = \log(\hat{Y}_{ik}) \\ \\ s_{ik}^{-} = -\log(\hat{Y}_{ik}) \end{cases} \tag{3.10}$$

The score s^+ is then used as a binary classifier with variable threshold on the index set $G \times S - P$ with L^+ being the true labels. Similarly, s^- is used as a binary classifier with L^- being the true labels. Here S denotes the index set of test samples. We vary the threshold of the classifiers and plot precision recall curves (PRC) to visualize accuracy of classification.

3.4 Naïve Nearest-neighbor Approach

A naïve way to predict expression levels from perturbation information is simple yet effective: averaging the expression profiles of related mutant strains in the training data to infer the expression profile of a novel query strain. For a query double deletion strain $A\Delta B\Delta$ with genes A and B deleted, suppose that the single deletion strains $A\Delta$ and $B\Delta$

are both available in the training data. Denote the index set of $A\Delta$ samples as $K_{A\Delta}$ and that of $B\Delta$ samples as $K_{A\Delta}$. The geometric mean of these samples is taken to predict gene expression levels in $A\Delta B\Delta$:

$$\hat{Y}^{(geo)}_{i,A\Delta B\Delta} = \left(\prod_{k\in K_{A\Delta}} Y_{ik}\right)^{\frac{1}{2|K_{A\Delta}|}} \left(\prod_{k\in K_{B\Delta}} Y_{ik}\right)^{\frac{1}{2|K_{B\Delta}|}} \tag{3.11}$$

3.5 An Approach Based on Network Modeling and Simulation

3.5.1 Modeling regulator-target interaction

We propose a prediction model based on simulation of gene regulatory networks.

A necessary part of relating genetic perturbation to expression profile is to model regulator-target interaction. This partly overlaps with the task of Chapter 2.

Two options are explored here. The first one is to model the relation between target levels and regulator levels as log-linear, the same as in (2.8) and inferelator [15]. In particular,

$$F^{(llnp)}_{ik}(x_{1k}, x_{2k}, \ldots; \theta) = \begin{cases} y^{(pert)}_{ik} & \text{if gene } i \text{ is perturbed in sample } k \\ F^{(lln)}_{i}(x_{1k}, x_{2k}, \ldots; \theta) & \text{otherwise} \end{cases} \tag{3.12}$$

where $F^{(lln)}$ is defined as in (2.8) and $y^{(pert)}_{ik}$ is the perturbed level of gene i in sample k, e.g. zero if it is deleted. Note that $F^{(llnp)}$ differs from $F^{(lln)}$ only in considering the perturbation information.

The second option enforces a truncation on the functional range of regulator levels as well as maximum and minimum allowed levels of target genes:

$$P = \{(i,k) : \text{gene } i \text{ is perturbed in sample } k\} \tag{3.13}$$

$$F_{ik}^{(llcp)}\begin{pmatrix} x_{1k}, \\ x_{2k}, \\ \vdots \\ ;\theta \end{pmatrix} = \begin{cases} y_{ik}^{(pert)} & (i,k) \in P \\ F_i^{(lln)}\begin{pmatrix} \min\{\max\{x_{1k}, \underline{r}_1\}, \overline{r}_1\}, \\ \min\{\max\{x_{2k}, \underline{r}_2\}, \overline{r}_2\}, \\ \vdots \\ ;\theta \end{pmatrix} & (i,k) \notin P \\ & \wedge F^{(lln)}(\cdots;\theta) \in [\underline{m}_i, \overline{m}_i] \\ \underline{m}_i & (i,k) \notin P \\ & \wedge F^{(lln)}(\cdots;\theta) < \underline{m}_i \\ \overline{m}_i & (i,k) \notin P \\ & \wedge F^{(lln)}(\cdots;\theta) > \overline{m}_i \end{cases} \tag{3.14}$$

Additional tuning parameters are introduced: $\underline{m}_i$ and $\overline{m}_i$ are the minimum and maximum allowed levels of gene i, respectively; $\underline{r}_j$ and $\overline{r}_j$ are the minimum and maximum recognizable functional range of regulator j. The truncation on the functional range of regulator levels is inspired by the observation of the limitation of the log-linear model as repressor levels approach zero in Section 2.4; the minimum and maximum allowed target levels are

intended to prevent over-extrapolation as observed in Subsection 2.8.3. This model will be referred to as *log-linear-cutoff model*.

3.5.2 Modeling Transcription Factor Concentration

At steady state, equation (1.3) implies that the concentration of a certain protein is proportional to the transcription rate of its encoding gene. Fold-change with respect to wild type has been used as the measurement of gene expression levels in this thesis; the same measure will be used for the measurement of protein concentration. Therefore the notation Y_{ik} can be interpreted either as steady state gene expression level or as steady state protein level.

In case a transcription factor is composed of multiple proteins, it is assumed that the concentration of the TF is far less than the concentration of the individual species of its member proteins. Under such assumption, the concentration of the TF is nearly proportional to the product of the levels of its member proteins. Assume that TF j contains n_{ij} protein molecules of gene i for each i, then:

$$[TF_j] \propto \prod_i (transcriptionRate_i)^{n_{ij}} \tag{3.15}$$

We can therefore define a function G to describe the level of a TF as determined by the levels of its encoding genes:

$$G_j(y_1, y_2, \ldots) = \prod_i y_i^{n_{ij}} \tag{3.16}$$

The composition of transcription factors, i.e. the value of n_{ij}'s, is drawn from biological knowledge of the transcription factors in the investigated species. In absence of the knowledge of TF compositions, all TFs are modeled as monomers.

3.5.3 Modeling Propagation of the Effect of Genetic Perturbation

This subsection addresses the problem of determining gene expression levels from a given set of gene interaction parameters and genetic perturbation information.

Inheriting the notations of (3.12), (3.14), and (3.16), we further denote

$$\vec{F}_k^{(llnp)} = (F_{1k}^{(llnp)}, F_{2k}^{(llnp)}, \ldots)^T \tag{3.17}$$

$$\vec{F}_k^{(llcp)} = (F_{1k}^{(llcp)}, F_{2k}^{(llcp)}, \ldots)^T \tag{3.18}$$

$$\vec{G} = (G_1, G_2, \ldots)^T \tag{3.19}$$

A difference equation system is defined in order to simulate the propagation of perturbation effects in the gene regulatory network:

$$y_{ik}^{(0)} = \begin{cases} 1 & \text{if gene } i \text{ is not perturbed in sample } k \\ y_{ik}^{(pert)} & \text{otherwise} \end{cases} \tag{3.20}$$

$$\vec{H}_k^{(j)}(\theta) = \begin{cases} (\vec{F}_k^{(llnp)}(\cdot;\theta) \circ \vec{G})^j \begin{pmatrix} y_{1k}^{(0)} \\ y_{2k}^{(0)} \\ \vdots \end{pmatrix} & \text{if the interaction model is } F^{(llnp)} \\ (\vec{F}_k^{(llcp)}(\cdot;\theta) \circ \vec{G})^j \begin{pmatrix} y_{1k}^{(0)} \\ y_{2k}^{(0)} \\ \vdots \end{pmatrix} & \text{if the interaction model is } F^{(llcp)} \end{cases} \tag{3.21}$$

The vector $\vec{H}_k^{(0)}(\theta)$ is a trivial prediction of the expression levels: except for the directly perturbed genes, all other genes has level 1, i.e. the wild type level. $\vec{H}_k^{(1)}(\theta)$, applying the function $\vec{F}_k$ which determines target levels from regulator levels, propagate the perturbation in regulator genes to their predicted direct targets. Every $\vec{H}_k^{(j+1)}$ propagate the perturbation one step further than $\vec{H}_k^{(j)}$.

While there is no guarantee that $\vec{H}_k^{(j)}$ will converge as k goes to infinity, we choose two positive integers M (e.g. 8) and N (e.g. 16) and define the prediction as

$$\vec{H}_k(\theta) = \frac{1}{N} \sum_{j=M+1}^{M+N} \vec{H}_k^{(j)}(\theta) \tag{3.22}$$

The parameter optimization task then involves finding the θ so that $\vec{H}_k(\theta)$ approaches the observed expression level $\vec{Y}_k$.

3.5.4 Modeling Data Noise

A similar noise model to that used in Chapter 2 is adopted here, as described by (2.29). The noise model is:

$$\begin{cases} U_{ik} \text{ (independently)} \sim \mathcal{N}\left(u\big(H_{ik}(\theta); p_i, q_i\big), \sigma_i^2\right) \\ Y_{ik} = u^{-1}\left(U_{ik}; p_i, q_i\right) \\ u(y; p, q) := \sqrt{1+p+q}\ln\left(\frac{y+p/2+\sqrt{y^2+py+q}}{1+p/2+\sqrt{1+p+q}}\right) \\ u^{-1}(\xi; p, q) := \left(1+\frac{p}{2}\right)\cosh\left(\frac{\xi}{\sqrt{1+p+q}}\right) + \left(\sqrt{1+p+q}\right)\sinh\left(\frac{\xi}{\sqrt{1+p+q}}\right) - \frac{p}{2} \end{cases} \tag{3.23}$$

where p_i, q_i are tuning parameters.

3.5.5 Full Model

The full model can be summarized as:

$$y_{ik}^{(0)} = \begin{cases} y_{ik}^{(pert)} & \text{if gene } i \text{ is perturbed in sample } k \\ 1 & \text{otherwise} \end{cases} \tag{3.24}$$

$$\theta = \begin{pmatrix} b_1 & c_{11} & c_{12} & \dots \\ b_2 & c_{21} & c_{22} & \dots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix} \tag{3.25}$$

$$\eta = (\vec{p}, \vec{q}, M, N, \underline{\vec{m}}, \overline{\vec{m}}, \underline{\vec{r}}, \overline{\vec{r}}) \tag{3.26}$$

$$F = F^{(llnp)} \text{ or } F^{(llcp)} \tag{3.27}$$

$$G_j(\vec{y}) = \prod_i y_i^{n_{ij}} \tag{3.28}$$

$$\vec{H}_k^{(j)}(\theta;\eta) = (\vec{F}_k(\cdot;\theta,\eta) \circ \vec{G})^j \left(y_{1k}^{(0)}, y_{2k}^{(0)}, \dots\right) \tag{3.29}$$

$$\vec{H}_k(\theta;\eta) = \frac{1}{N} \sum_{j=M+1}^{M+N} \vec{H}_k^{(j)}(\theta;\eta) \tag{3.30}$$

$$U_{ik} \mid \sigma_i^2, \theta, \eta, F \text{ (independently)} \sim \mathcal{N}\left(u\big(H_{ik}(\theta); p_i, q_i\big), \sigma_i^2\right) \tag{3.31}$$

$$Y_{ik} = u^{-1}\left(U_{ik}; p_i, q_i\right) \tag{3.32}$$

$$u(y;p,q) = (1+p+q)\ln\left(\frac{y + p/2 + \sqrt{y^2+py+q}}{1+p/2+\sqrt{1+p+q}}\right) \tag{3.33}$$

$$u^{-1}(\xi;p,q) = \left(1+\frac{p}{2}\right)\cosh\left(\frac{\xi}{\sqrt{1+p+q}}\right) + \left(\sqrt{1+p+q}\right)\sinh\left(\frac{\xi}{\sqrt{1+p+q}}\right) - \frac{p}{2} \tag{3.34}$$

Table 3.1: List of parameters and variables in network simulation

Variables	
Y_{ik}	observed expression level of gene i in sample k
Known quantities	
n_{ij}	number of protein molecules encoded by gene i per TF complex j
$y_{ik}^{(pert)}$	perturbed level of gene i in sample k
$y_{ik}^{(0)}$	trivial prediction of Y_{ik}
Tuning parameters	
F	which regulator-target interaction model is used: see (3.12) and (3.14)
η	collective notation of all tuning parameters
$M \in \mathbb{Z}^+$	number of simulation steps discarded
$N \in \mathbb{Z}^+$	number of simulation steps to include for estimating expression levels
$p_i > 0$	noise shape tuner
$q_i > 0$	noise shape tuner
$\underline{m}_i \geq 0$	minimum allowed level of gene i
$\overline{m}_i \geq 0$	maximum allowed level of gene i
$\underline{r}_j \geq 0$	lower truncation point of the functional range of regulator j
$\overline{r}_j \geq 0$	upper truncation point of the functional range of regulator j
Optimizable parameters	
θ	matrix for collectively denoting all b_i's and c_{ij}'s
$b_i > 0$	basal transcriptional level of gene i
c_{ij}	regulative power of TF j on gene i
σ_i^2	fold noise of the level of gene i

The model defined above can be described as two layers: one to describe how transcription factor levels determine target levels and the other using the first layer recursively to describe the effect of the perturbation of transcription factors on the whole network.

3.5.6 Bayesian Prior Distribution of Parameters

The prior distribution of parameters σ_i^2 is modeled to be independent from b_i and c_{ij} as well as from each other:

$$p(\theta, \sigma_1^2, \sigma_2^2, \ldots \mid \eta, F) = p_\theta(\theta \mid \eta, F) \prod_i \left(p_{\sigma_i^2}(\sigma_i^2 \mid \eta, F) \right) \tag{3.35}$$

where θ is the collective matrix notation of parameter b_i's and c_{ij}'s:

$$\theta = \begin{pmatrix} b_1 & c_{11} & c_{12} & \ldots \\ b_2 & c_{21} & c_{22} & \ldots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix} \tag{3.36}$$

and η is the collective notation of tuning parameters.

The prior distribution of σ_i^2 is the inverse-gamma distribution, similar as defined in 2.6.2:

$$\sigma_i^2 \mid \eta, F \sim \text{Inv}\Gamma\left(\frac{df}{2}, \frac{df s_0^2}{2}\right) \tag{3.37}$$

This definition facilitates the marginalization of σ_i^2 (see 3.6.2).

The prior probability density function of θ consists of three factors:

$$p_\theta(\theta \mid \eta, F) \propto f^{(pen)}(\theta \mid \eta, F) f^{(s)}(\theta) f^{(r)}(\theta) \tag{3.38}$$

where $f^{(pen)}$ penalizes the inconsistency between $\vec{H}_k^{(M+1)}, \vec{H}_k^{(M+2)}, \ldots, \vec{H}_k^{(M+N)}$, $f^{(s)}$ carries out the task of feature selection, and $f^{(r)}(\theta)$ serves the purpose of ensuring that the product (3.38) has a finite integral so that p_θ is normalizable.

The fact that wild type and mutant strains of the studied organism can survive and each exhibits relatively a consistent gene expression profile implies that their dynamic gene regulatory networks are stabilizing systems, with steady state expression profile being the stable equilibrium. Since the later simulation steps in the system (3.21) are intended to predict the steady state of the real biological network, it is important to require (3.21) to be auto-stabilizing as well, although we do not require it be strictly convergent.

This requirement is carried out by a factor in the prior probability density of θ that penalizes the magnitude of difference between $\vec{H}_k^{(M+1)}, \vec{H}_k^{(M+2)}, \ldots, \vec{H}_k^{(M+N)}$.

Denote the index set of training samples as T, the penalizing factor is defined as

$$f^{(pen)}(\theta \mid \eta, F) = \prod_{i \in G, j \in T} e^{\left(-\frac{1}{2Ns^2_{(pen)}} \sum_{k=M+1}^{N} \left(u(H_{ik}^{(j)}(\theta;\eta);p_i,q_i) - u(H_{ik}(\theta;\eta);p_i,q_i) \right)^2 \right)} \tag{3.39}$$

where H_{ik} is the average of $(H_{ik}^{(M+1)}, H_{ik}^{(M+2)}, \ldots, H_{ik}^{(M+N)})$ as defined in (3.22) and $s^2_{(pen)}$ is a super parameter. By default, $s^2_{(pen)}$ is equal to s_0^2, the prior variance suggestion of $U_{ik}|\theta, \eta, F$. The smaller $s^2_{(pen)}$ is, the harsher the penalty on the inconsistency between $\vec{H}_k^{(M+1)}, \vec{H}_k^{(M+2)}, \ldots, \vec{H}_k^{(M+N)}$.

The second factor $f^{(s)}(\theta)$ in the prior density (3.38) of θ is enforced to carry out certain feature selection and regularization techniques.

The form of $f^{(s)}$ is designed with the aid of the lasso [18].

Note that (3.16) relates transcription factor levels with the level of their encoding genes. Now we define

$$X_{jk} = G_j(Y_{1k}, Y_{2k}, \ldots) \tag{3.40}$$

and refer to X_{jk} as observed regulator levels.

For each gene i, we perform lasso regression with $\log(Y_i)$ as the response variable and the log-level of all regulators that are not encoded by gene i as explanatory variables. This process is in its nature explaining target expression levels from regulator expression levels, which has been addressed in the previous chapter. The fitting process is described in 2.5. All lasso variables are normalized using their standard deviations and the lasso L^1-shrinkage parameter is determined by trials in cross-validation.

The lasso regression generates coefficients for a log-linear expression model:

$$\frac{\log(Y_{ik})}{\upsilon_i} \sim \alpha_{(lasso)i} + \sum_j \beta_{(lasso)ij} \frac{\log(X_{jk})}{\xi_j} \tag{3.41}$$

where ξ_j and υ_i are lasso variable normalizers, in this case calculated from standard deviation of lasso variables. The L^1 norm of row vector $\vec{\beta_i}$, i.e. $\sum_j |\beta_{ij}|$, is constrained by the L^1-shrinkage parameter (as in (2.15)).

Although this model itself is unsuitable for the predicting expression levels from perturbation information, we can take certain advice from it for the sake of feature selection and regularization.

The log-linear model (2.7) can be rewritten as:

$$\frac{\log(F_i^{(lln)})}{\upsilon_i} = \frac{\log(b_i)}{\upsilon_i} + \sum_j \left(\frac{c_{ij}\xi_j}{\upsilon_i}\right) \frac{\log(x_j)}{\xi_j} \tag{3.42}$$

Recall that the lasso uses L^1-shrinkage to guard against overfitting; we can enforce the same L^1-shrinkage on the θ for the network simulation model. Comparing (3.41) and (3.42), we require that

$$\sum_j \left|\frac{c_{ij}\xi_j}{\upsilon_i}\right| \leq \sum_j |\beta_{(lasso)ij}| \tag{3.43}$$

We define the feature selection factor $f^{(s)}$ of the prior density (3.38) as

$$f^{(s)}(\theta \mid \beta_{(lasso)}) = \begin{cases} 1 & \forall i, \sum_j \left|\frac{c_{ij}\xi_j}{\upsilon_i}\right| \leq \sum_j |\beta_{(lasso)ij}| \\ 0 & \text{otherwise} \end{cases} \tag{3.44}$$

In this definition, although we do not appoint in advance which c_{ij}'s are allowed to be nonzero, the restriction on L^1 norm will force the optimization process to set only a limited number of coefficients to be non-zero.

The factor $f^{(r)}$ in the prior density (3.38) of θ ensures that the prior density has a finite integral. The restriction on coefficient L^1 norm enforced by $f^{(s)}(\cdot)$ already ensures that

the prior density has a finite support in the coefficient space, therefore we can regularize only the basal transcription parameter distribution by simply defining

$$f^{(r)}(\theta) = \prod_i \phi(b_i; 0, s^2_{(b)}) \tag{3.45}$$

where $s^2_{(b)}$ is a super parameter and ϕ is the probability density function of the standard normal distribution.

3.6 Optimization Method

3.6.1 Choosing Tuning Parameters

Parameter M represents the number of propagation steps before simulated levels are used for predictions and is chosen empirically. Since the effect of indirect regulation would usually be insignificant when the number of intermediate regulators exceeds 3 [4], an integer larger than 3 but at the same level of magnitude will be a valid choice. The default M used in this research is 8.

Parameter N represents the number of simulation steps to include for the prediction. A larger N is preferable for the robustness of the model. Computational affordability is another concern in choosing N. The default N used in this research is 16.

Parameters p_i and q_i dictate the form of the noise model. They are determined in the same procedure as in Subsection 2.7.1.

The minimum and maximum allowed gene levels $\underline{m}_i$ and $\overline{m}_i$ are set using the minimum and maximum observed levels in the training data:

$$\underline{m}_i = \min_{k \in \text{training}} Y_{ik} \tag{3.46}$$

$$\overline{m}_i = \max_{k \in \text{training}} Y_{ik} \tag{3.47}$$

The recognizable functional range of regulators is also learned from the observations in the training data:

$$\underline{r}_j = \min_{k \in \text{training}} G_j(Y_{1k}, Y_{2k}, \ldots) \tag{3.48}$$

$$\overline{r}_j = \max_{k \in \text{training}} G_j(Y_{1k}, Y_{2k}, \ldots) \tag{3.49}$$

where G_j is defined as in (3.16) and $G_j(Y_{1k}, Y_{2k}, \ldots)$ is considered as the observed level of transcription factor j in sample k.

3.6.2 Marginalizing Noise Parameters

The noise parameters $\sigma_1^2, \sigma_2^2, \ldots$ can be marginalized out so that the explicit form of the marginal likelihood $p(Y \mid \theta, \eta, F)$ may be calculated, which does not contain any reference of σ_i^2. This will facilitate the fitting process of parameter θ.

By calculation similar to that in Subsection 2.7.2 and [12], this marginal likelihood is equal to

$$p(Y \mid \theta, \eta, F) = \frac{\left(\frac{df s_0^2}{2}\right)^{\frac{df|G|}{2}}}{\prod_{i \in G} \left(\frac{df s_0^2}{2} + \frac{\sum_{k \in T} (u(Y_{ik};\eta) - u(H_{ik}(\theta;\eta);p_i,q_i))^2}{2}\right)^{\frac{df+|T|}{2}}} \left(\frac{\Gamma\left(\frac{df+|T|}{2}\right)}{(2\pi)^{\frac{|T|}{2}}\Gamma\left(\frac{df}{2}\right)}\right)^{|G|} \tag{3.50}$$

where G is the index set of studied genes, T the set index set of training samples, and s_0^2 and df are super parameters of the prior distribution of σ_i^2.

In the implementation of the method, the super parameters s_0^2 and df may be user specified. By default, $df = 1$ and s_0^2 is the average cross-replicates sample variance of U_{ik} per gene per strain in the training data.

With the tuning parameters η fixed and σ_i^2 marginalized, the unnormalized posterior probability density function of θ is, by (2.54):

$$p(\theta, \mid Y, \eta, F) \propto p(\theta, Y \mid \eta, F) \tag{3.51}$$

$$= p(Y \mid \theta, \eta, F) p(\theta \mid \eta, F) \tag{3.52}$$

$$\propto p(Y \mid \theta, \eta, F) f^{(pen)}(\theta \mid \eta, F) f^{(s)}(\theta) f^{(r)}(\theta) \tag{3.53}$$

$$= \tag{3.54}$$

$$\frac{\left(\frac{df s_0^2}{2}\right)^{\frac{df|G|}{2}}}{\prod_{i \in G} \left(\frac{df s_0^2}{2} + \frac{\sum_{k \in T} (u(Y_{ik}; p_i, q_i) - u(H_{ik}(\theta; \eta); p_i, q_i))^2}{2}\right)^{\frac{df+|T|}{2}}} \left(\frac{\Gamma\left(\frac{df+|T|}{2}\right)}{(2\pi)^{\frac{|T|}{2}} \Gamma\left(\frac{df}{2}\right)}\right)^{|G|} \cdot \tag{3.55}$$

$$f^{(pen)}(\theta \mid \eta, F) f^{(s)}(\theta) f^{(r)}(\theta)$$

$$= \frac{f^{(pen)}(\theta \mid \eta, F) f^{(s)}(\theta) f^{(r)}(\theta) (df s_0^2)^{\frac{df|G|}{2}} \Gamma^{|G|}\left(\frac{df+|T|}{2}\right) \Gamma^{-|G|}\left(\frac{df}{2}\right) \pi^{-\frac{|G||T|}{2}}}{\prod_{i \in G} \left(df s_0^2 + \sum_{k \in T} \left(u(Y_{ik}; p_i, q_i) - u(F_i(\vec{x}_k; \theta, \eta); p_i, q_i)\right)^2\right)^{\frac{df+|T|}{2}}} \tag{3.56}$$

$$=: f(\theta \mid Y, \eta, F) \tag{3.57}$$

By this definition, $f(\theta \mid Y, \eta, F)$ is proportional to the posterior density of $\theta \mid Y, \eta, F$ and monotonically increases as the prediction error of $u(Y_{ik}; p_i, q_i)$ decreases. $f^{(pen)} f^{(s)} f^{(r)}$ defines the prior distribution of θ and has an explicit form as defined in Subsection 3.5.6.

We optimize θ to maximize the function f (3.57) or estimate the posterior distribution of θ according to it.

3.6.3 Optimizing Interaction Parameters with Evolutionary Algorithm

As the tuning parameters η have been pre-determined and the noise parameters σ_i^2 marginalized, it remains to fit the regulator-target interaction parameters θ.

By the definition (3.57), f is an unnormalized posterior density function of $\theta \mid \eta, F$ and can be calculated explicitly. Maximization of f will provide the optimal value of θ; by integrating or sampling according to f one can sample from the posterior distribution of θ.

In Chapter 2, the interaction parameters are fitted in the regression problem with regulator levels being explanatory variables and target levels being response variables. The optimization method from Chapter 2 however fails to serve the task of predicting gene expression levels solely from perturbation information. In this task, regulator levels are not available as predictors, especially given the fact that many target genes are themselves regulators of other genes.

Instead, we deem perturbation information as the only explanatory variables and aim to optimize parameter θ so that $H_{ik}(\theta; \eta)$ approaches Y_{ik} in the training data.

This is the most computationally challenging part of the problem, since the function $H_{ik}(\theta; \eta)$ (hence f) is of very complicated form and may exhibit chaotic behavior in certain ranges of θ. With the landscape of the posterior distribution of $\theta \mid \eta, F$ unknown and potentially very complicated, conventional sampling methods may be inefficient in simulating its distribution.

Ideally the entire posterior distribution of θ is to be estimated. However, the calculation of its unnormalized posterior density f (3.57) consumes a significant amount of computational resource even for one fixed value of θ (e.g. around 3 CPU seconds in some of our fitting tasks). Given the computational cost and that the landscape of f might be very complicated, we only attempt to optimize θ to a certain level and sample in a small neighborhood around the optimized value.

The evolutionary optimization approach has proved to be an effective tool for optimizing θ.

We start with an initial pool of parameter matrices: $\theta^{(11)}, \theta^{(12)}, \ldots, \theta^{(1K)}$, which can be random, trivial, or naïve guesses of parameter values.

For any array $\theta^{(i1)}, \theta^{(i2)}, \ldots, \theta^{(iK)}$, referred to as generation i, the pool is updated in the following manner:

First, we randomly pick two different indices j_{i0} and j_{i1} from $1, \ldots, K$ with probabilities proportional to their posterior density:

$$\frac{\left(f(\theta^{(i1)}), \quad f(\theta^{(i2)}), \quad \ldots, \quad f(\theta^{(iK)}) \right)}{f(\theta^{(i1)}) + f(\theta^{(i2)}) + \ldots + f(\theta^{(iK)})} \tag{3.58}$$

Next, we mix $\theta^{(ij_{i0})}$ and $\theta^{(ij_{i1})}$ by rows. This step is referred to as "mating" in the terminology of evolutionary algorithm. We generate independent standard Bernoulli variables $k_{i11}, k_{i12}, \ldots$, each takes value 0 or 1 with equal probabilities. The value 0 indicates taking a row from $\theta^{(ij_{i0})}$ and 1 indicates taking a row from $\theta^{(ij_{i1})}$. Thus the sth row of the mixed parameter is taken from the sth row of $\theta^{(ij_{i0})}$ if $k_{i1s} = 0$, or of $\theta^{(ij_{i1})}$ if $k_{i1s} = 1$.

The mixed parameter is denoted as $\theta^{(i)(mix_1)}$. In the same way we generate independently $\theta^{(i)(mix_2)}, \ldots, \theta^{(i)(mix_L)}$; these are referred to as the offspring.

The third step is mutation. Each offspring matrix is mutated randomly and independently. For $\theta^{(i)(mix_l)}$, we randomly pick a few entries (< 20) from it and add to each of these entries an independent Gaussian variable. The standard deviation of the Gaussian variables is pre-determined and denoted as $s^{(muta)}$. The probability of each entry being picked is determined by a prior score of likelihood of regulation. Denote

$$\theta^{(i)(mix_l)} = \begin{pmatrix} b_1^{(i)(mix_l)} & c_{11}^{(i)(mix_l)} & c_{12}^{(i)(mix_l)} & \ldots \\ b_2^{(i)(mix_l)} & c_{21}^{(i)(mix_l)} & c_{22}^{(i)(mix_l)} & \ldots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix} \tag{3.59}$$

Since $c_{\alpha\beta}^{(i)(mix_l)}$ represents the regulation relation between target α and TF β, its probability of being chosen to mutate is higher if we have assigned a higher regulation score to this pair of target-regulator. The regulation score is defined as

$$0.1 + |\text{corr}\left((U_{i1}, U_{i2}, \ldots), (\log(X_{j1}), \log(X_{j2}), \ldots)\right)| \tag{3.60}$$

where X_{jk} is the observed level of regulator j in training sample k. Alternatively, other regulation scoring system such as NetProphet [4] may also be used.

The mutated parameter is denoted as $\theta^{(i)(muta_l)}$.

The last step in an updating cycle is "selection". The combined array $comb^{(i)} = \left(\theta^{(i)(muta_1)}, \theta^{(i)(muta_2)}, \ldots, \theta^{(i)(muta_L)}, \theta^{(i1)}, \theta^{(i2)}, \ldots, \theta^{(iK)}\right)$ consists of previous pool and cur-

rent mutated parameter matrices. Function f (3.57) is applied on each of them to calculate their unnormalized posterior density. The K members with largest posterior probabilities are chosen to form the new generation $\left(\theta^{((i+1)1)}, \theta^{((i+1)2)}, \ldots, \theta^{((i+1)K)}\right)$.

These steps are repeated to generate newer generations until the best performance of members in the latest generations cease to improve significantly.

3.6.4 Pipelining Perturbation-based Prediction and Regulator-based Prediction

The method described in this chapter mainly addresses the problem of predicting expression levels from genetic perturbation information. The methods described in Chapter 2 are intended for predicting target gene levels from regulator levels. In practice, we can incorporate the methods in Chapter 2 to execute a part of the task of this chapter, greatly reducing the computational cost.

Consider the task of predicting the genome-wide expression levels from perturbation information: In both *Saccharomyces cerevisiae* and *Cryptococcus neoformans*, this requires the prediction of nearly 7000 genes, up to 300 of which are considered regulators. It would demand an extremely high dimensionality of parameter space and result in enormous space and time complexity.

However, the problem can be decomposed to two stages: 1) predicting regulator expression levels from perturbation information and 2) predicting levels of non-regulators from the predicted levels of regulators.

Stage 2) employs methods from Chapter 2, which demands much less spatial and temporal complexity compared to the task of perturbation-based prediction. Stage 1) now handles only the regulator genes and fall into the affordable range of computational complexity (see Subsection 3.7.1).

In the data of *C. neoformans*, for example, the work flow of the expression prediction system and its training process can be summarized as:

Figure 3.1.: Predicting genome-wide expression levels from perturbation information

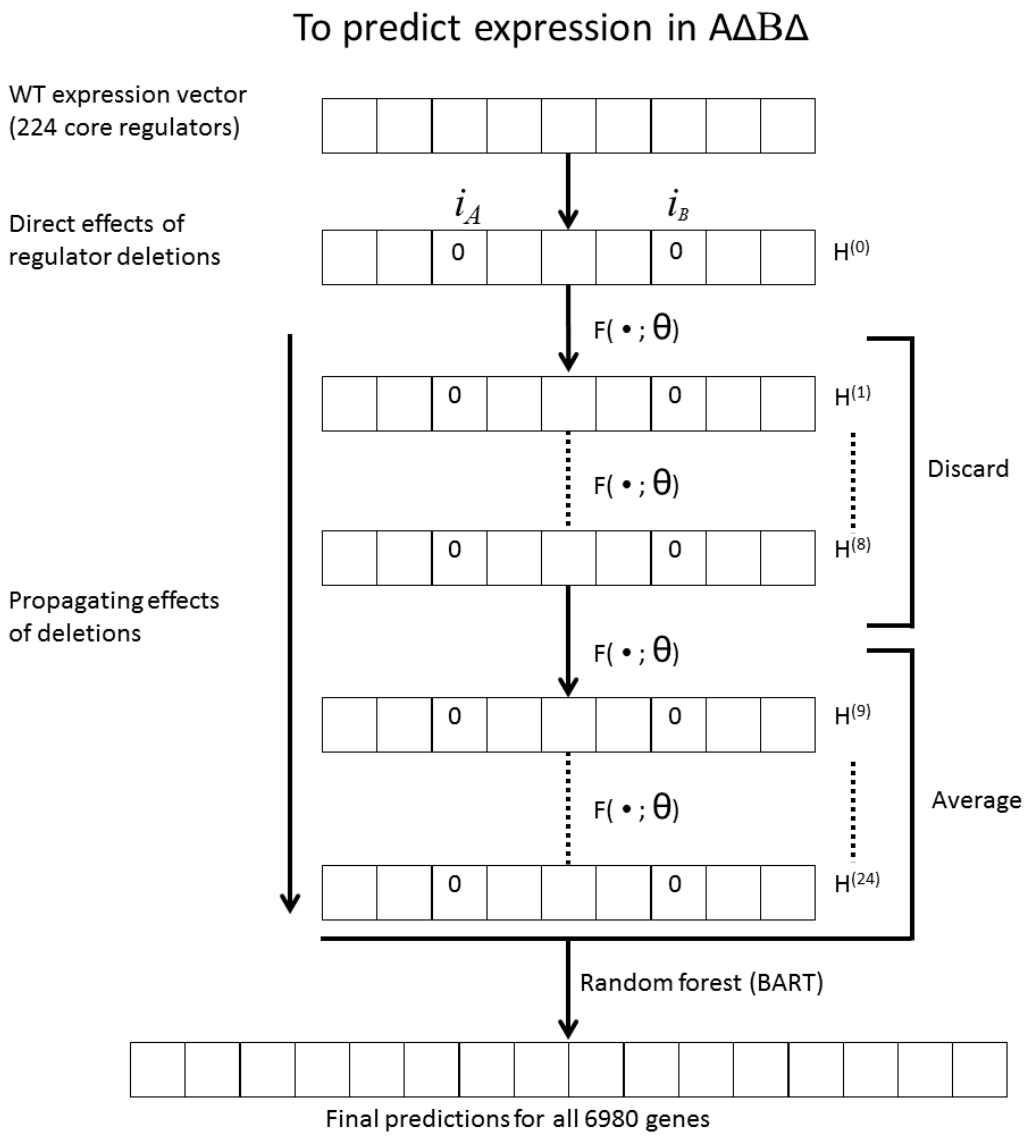
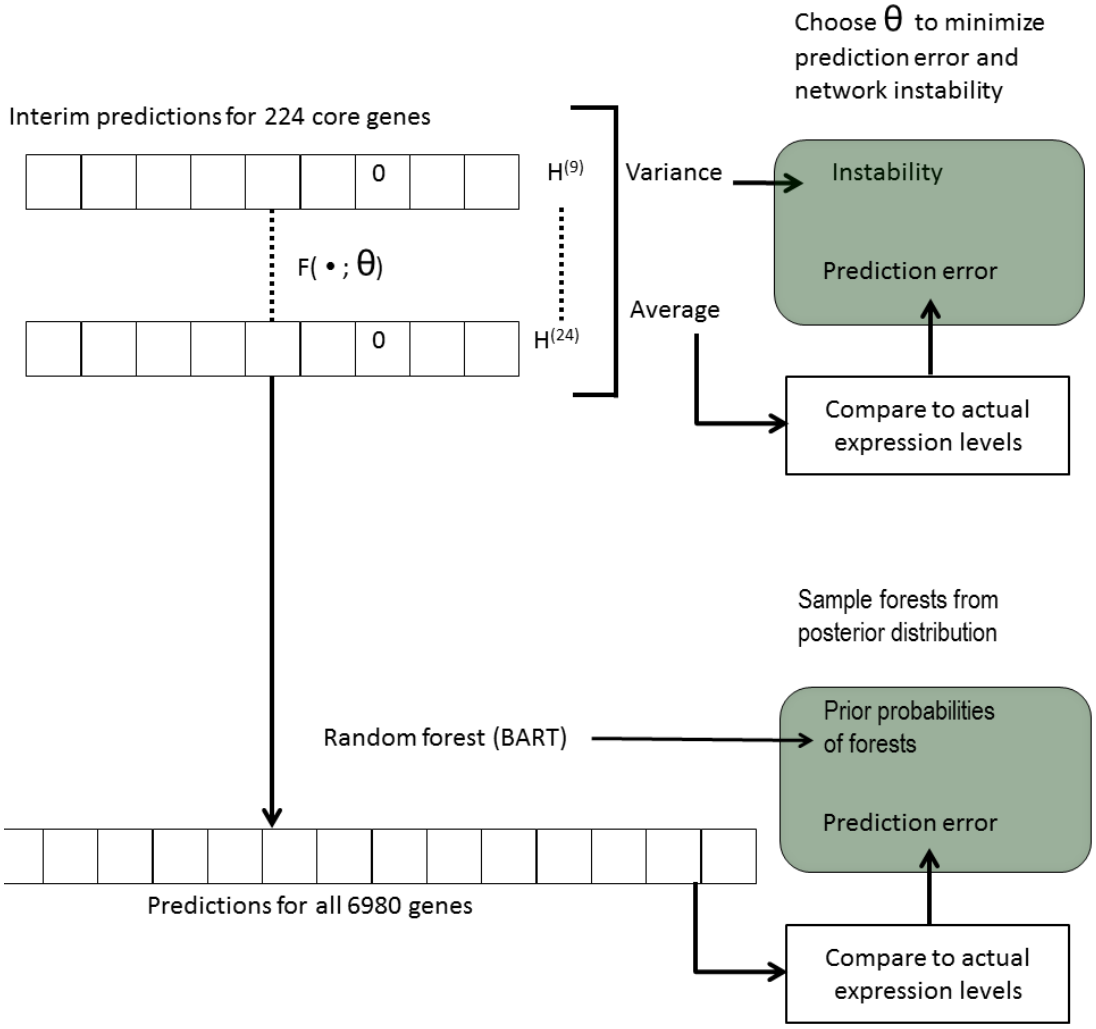


Figure 3.2.: Training the simulation module and the regression module in the expression prediction system



3.7 Results

3.7.1 Predicting Expression Levels of Regulators

RNA-Seq data of *Cryptococcus neoformans* produced in Brent Lab and Doering Lab is used to measure the performance of the methods. For data normalization, the pseudocount in (2.2) is set as $\nu = 5249$, which is the 0.5 percentile of nonzero expression levels in FPKM from all single deletion and wild type strains.

The tuning parameters are determined using the procedures described in 3.6.1. In particular, $M = 8, N = 16$ and p_i and q_i are determined using the genome-wide expression data of single deletion samples and wild type samples by the same method as in the previous chapter (details can be found in 2.7.1). The parameters follow the semantics of Table 3.1.

First, we trained two network simulation models, based on the choice log-linear regulation $F = F^{(llnp)}$ (3.12) and the choice log-linear-cutoff regulation $F = F^{(llcp)}$ (3.14) respectively, of the 224 regulator genes of *Cryptococcus neoformans* as listed in Appendix E using the expression matrix of the single deletion strains and wild types for supervision.

The major time consumption of the training process is due to the calculation of the unnormalized posterior density f (3.57) of θ. Denote the number of genes in the network as $|I|$ and the number of training samples $|T|$, the time complexity for calculating each value of $f(\theta \mid \eta, F)$ is approximately $\mathcal{O}(|I|^2|T|(M+N))$.

With 224 genes, all of which are regulators, and 291 samples, the calculation of f took 3 to 4 CPU seconds for each value of θ. We invoked 32 parallel processes using

Message Passing Interface (MPI) to calculate f of the offspring parameter matrices in the evolutionary algorithm. The generation size was 256 and offspring size of each generation was 32. The updating process of each generation therefore consumed 3 to 4 real-world seconds; the optimization of both models were completed in one week.

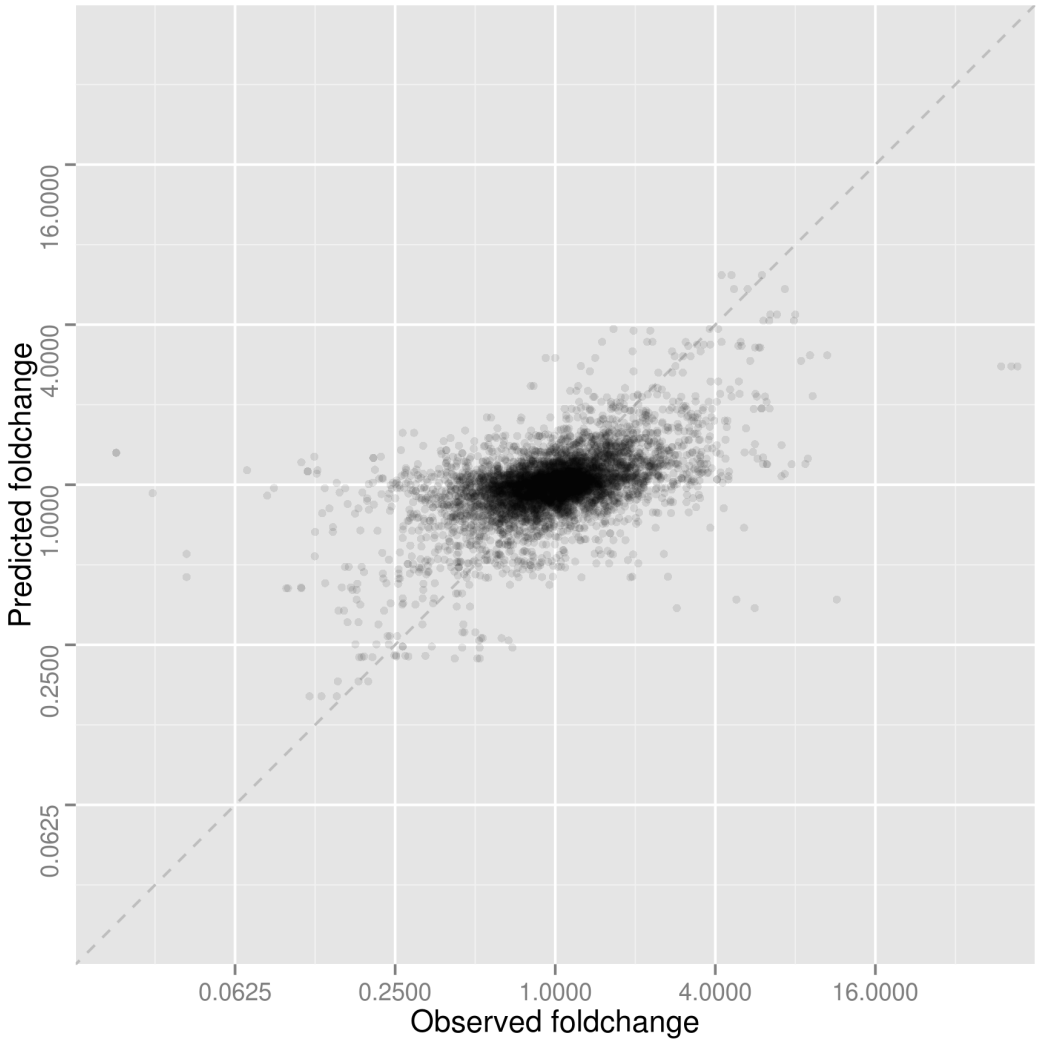
We used these network simulation models to predict the expression levels of the 224 regulator genes in the double deletion strains and compared its accuracy with the naïve prediction made from taking the geometric mean of the single deletion strain profiles as discussed in Section 3.4. The performance is:

Table 3.2: Prediction accuracy of $\log_2$ levels of 224 *C. neoformans* regulators

	R^2	rmse	corr
NetSim$\|_{F=F^{(llnp)}}$	0.19	0.74	0.53
NetSim$\|_{F=F^{(llcp)}}$	0.29	0.68	0.54
SingleΔ geometric mean	0.25	0.70	0.51

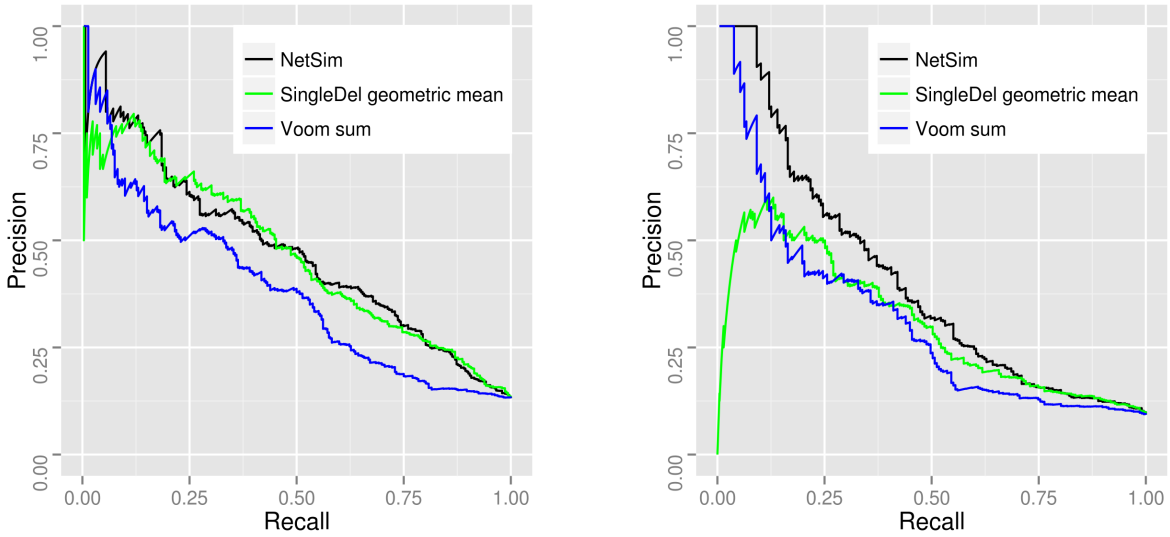
Here NetSim$\|_{F=F^{(llnp)}}$ is based on log-linear interaction model and NetSim$\|_{F=F^{(llcp)}}$ is based on log-linear-cutoff interaction model; the latter exhibit much stronger predictive power than the former and than the prediction produced by taking geometric mean of single deletion strain profiles. Moreover, the R^2 of log-fold-change predicted on the training data by NetSim$\|_{F=F^{(llcp)}}$ is 0.30, only slightly larger than its R^2 on the test data, indicating that over-fitting has been minimal.

Figure 3.3.: Prediction of 224 *C. neoformans* regulators by network simulation



We also tested the ability of NetSim$|_{F=F^{(llcp)}}$ in recovering differential expressions (DE) of the 224 regulators in the double deletion strains. To generate a gold standard, we used voom [3] to identify the statistically significant differential expressions in the double deletion strains compared to their same-batch wild type samples. An expression with p-value smaller than 0.001 is considered significantly differential. Using the procedure described in Subsection 3.3.5, the precision-recall curves (PRC) of DE recovery are:

Figure 3.4.: Recovery of differential expression in 224 *C. n.* regulators



(a) Recovery of positive DE

(b) Recovery of negative DE

For comparison, we included "voom sum", which stands for a naïve DE recovery method made from combining the voom scores of single deletion strains. Denote the DE p-value of gene i in strain j as p_{ij} and the fold change as m_{ij}, we define voom scores as

$$s_{ij} = \begin{cases} |\log(p_{ij})| & p_{ij} \leq 0.001 \wedge m_{ij} > 1 \\ 0 & p_{ij} > 0.001 \\ -|\log(p_{ij})| & p_{ij} < 0.001 \wedge m_{ij} < 1 \end{cases} \tag{3.61}$$

To recover DE in strain $\Delta A \Delta B$, the voom sum $s_{i\Delta A} + s_{i\Delta B}$ is used as a naïve score and tested using the procedures described in Subsection 3.3.5.

Alternatively we tried training the network simulation model on a larger network, involving 332 genes, all of which are considered regulators, and the same 291 samples.

The time complexity increased but no clear improvement in prediction accuracy was observed.

3.7.2 Predicting Expression Levels of All Genes

After the expression of the 224 regulator genes was predicted, we used the methods in Chapter 2 to extend the prediction to all the 6980 studied genes of *C. neoformans*.

Contrasting from Chapter 2, the expression of non-regulator genes in the test data is predicted from network simulation predicted regulator levels instead of observed regulator levels. Correspondingly when training methods borrowed from Chapter 2, we use not only the observed regulator levels in the training data, but also the network simulation fitted regulator levels.

First we applied the trained network simulation model $\text{NetSim}|_{F=F^{(llcp)}}$ on the training data (i.e. single deletions and wild types) itself, producing a fit of the regulator expression levels $\hat{X}_{jk}^{(train)}$. Denoting the observed regulator levels in the training data as X_{jk} and the sample indices of training data as $1, 2, \ldots, K$, we constructed a data set

$$X^{(cons)} = \begin{pmatrix} X_{11} & X_{12} & \ldots & X_{1K} & \hat{X}_{11}^{(train)} & \hat{X}_{12}^{(train)} & \ldots & \hat{X}_{1K}^{(train)} \\ X_{21} & X_{22} & \ldots & X_{2K} & \hat{X}_{21}^{(train)} & \hat{X}_{22}^{(train)} & \ldots & \hat{X}_{2K}^{(train)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix} \tag{3.62}$$

$$Y^{(cons)} = \begin{pmatrix} Y_{11} & Y_{12} & \ldots & Y_{1K} & Y_{11} & Y_{12} & \ldots & Y_{1K} \\ Y_{11} & Y_{12} & \ldots & Y_{1K} & Y_{11} & Y_{12} & \ldots & Y_{1K} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix} \tag{3.63}$$

so that $X^{(cons)}$ contains both the observed level of regulators and the predicted level of regulators in the training data, while $Y^{(cons)}$ contains two copies of observed target levels.

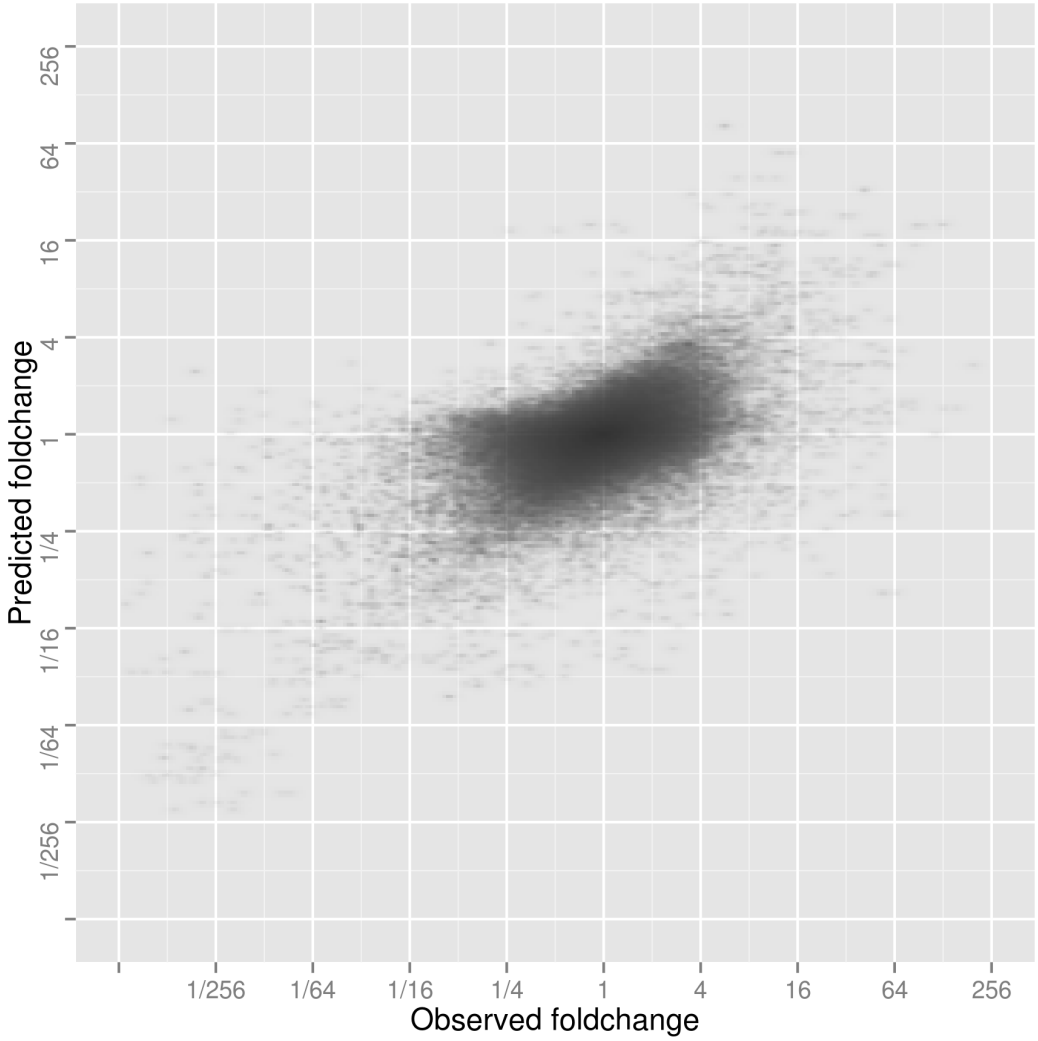
We used $X^{(cons)}$ and $Y^{(cons)}$ to train models described in Chapter 2. Finally, we applied the trained model on the regulator levels predicted by the network simulation model in the test data, generating genome-wide expression predictions.

The two models from Chapter 2 used here are 1) the BART-based semi-parametric model and 2) the lasso-based log-linear model (2.8).

Table 3.3: Prediction accuracy of $\log_2$ levels of all 6980 *C. neoformans* genes

	R^2	rmse	corr
NetSim→(BART+lasso)	0.27	0.86	0.53
NetSim→BART	0.27	0.87	0.52
NetSim→lasso	0.26	0.87	0.52
SingleΔ geometric mean	0.23	0.89	0.49

Figure 3.5.: Prediction of all 6980 genes by network simulation



We partitioned the entries in the test data according to their absolute value of log-fold-change and analyzed the performance of our predictions in each log-fold-change window:

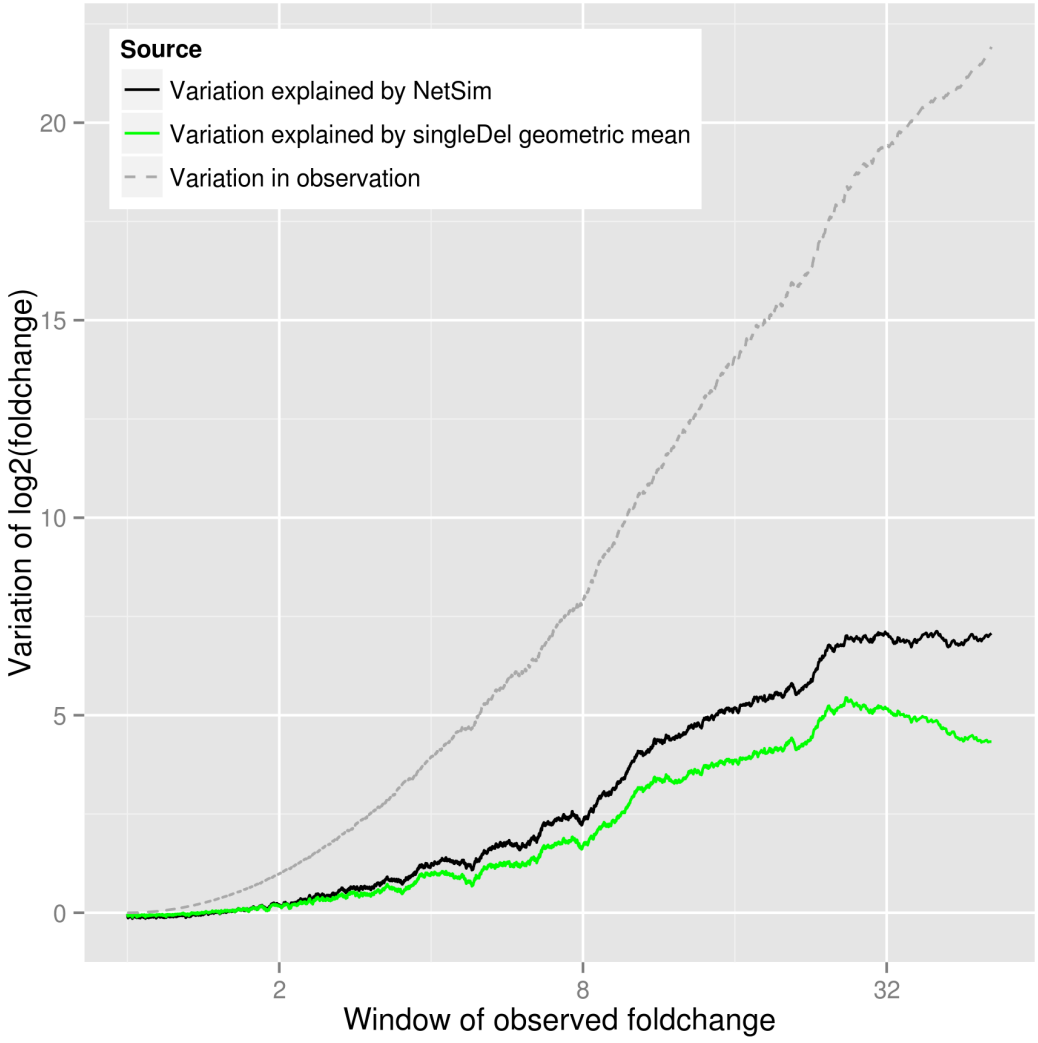
Table 3.4: RMSE of $\log_2$ expression by fold windows

	NetSim→(BART + lasso)		SingleΔ geometric mean	
Range of $\lvert\log_2(Y)\rvert$	rmse	R^2	rmse	R^2
$[0, \log_2(1.5))$	$\log_2(1.34)$	-0.84	$\log_2(1.29)$	-0.46
$[\log_2(1.5), 2)$	$\log_2(1.95)$	0.23	$\log_2(1.98)$	0.19
$[2, 3)$	$\log_2(3.86)$	0.30	$\log_2(4.16)$	0.22
$[3, 4)$	$\log_2(6.17)$	0.36	$\log_2(6.99)$	0.27
$(4, \infty)$	$\log_2(13.1)$	0.34	$\log_2(16.3)$	0.22

As the fold change of actual expression level increases, rmse of $\log_2(\hat{Y})$ increases but R^2 increases as well. Except in the log-fold window $(-\log_2(1.5), \log_2(1.5))$ which may not include many biologically significant expression changes, the network simulation model appear to provide better prediction than the geometric mean.

We ranked the observed expression levels of all genes in all test samples by $\lvert\log_2(Y)\rvert$ and created a sliding window of 1000 observations in the sorted list. Variance of $\log_2(Y)$ and mse and R^2 of the predictions $\log_2(\hat{Y})$ are calculated for each position of the sliding window in order to analyze the fraction of variance explained by the predictions:

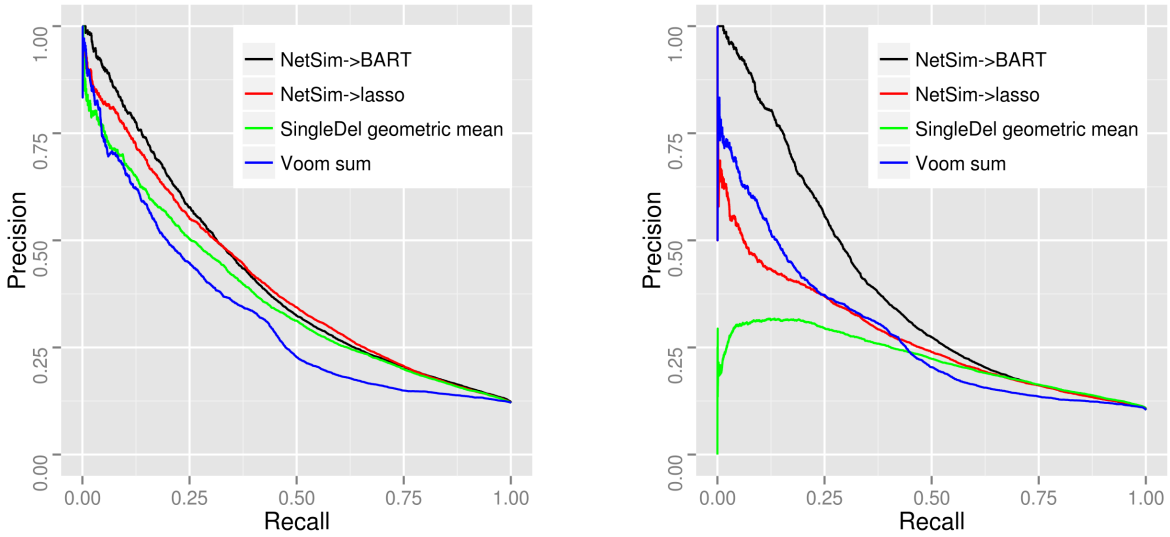
Figure 3.6.: Variance explained by predictions per fold window



The improvement in predictive power that the network simulation model provided over the naïve prediction (geometric mean) is consistent across the broad strata of $|\log_2(Y)|$.

BART-based semi-parametric model also generates a posterior predicative distribution. Such a distribution can be used in the procedure in Subsection 3.3.5 for discovering differential expression in the test data. The PRCs are:

Figure 3.7.: Recovery of differential expression in all 6980 *C. n.* genes



(a) Recovery of positive DE

(b) Recovery of negative DE

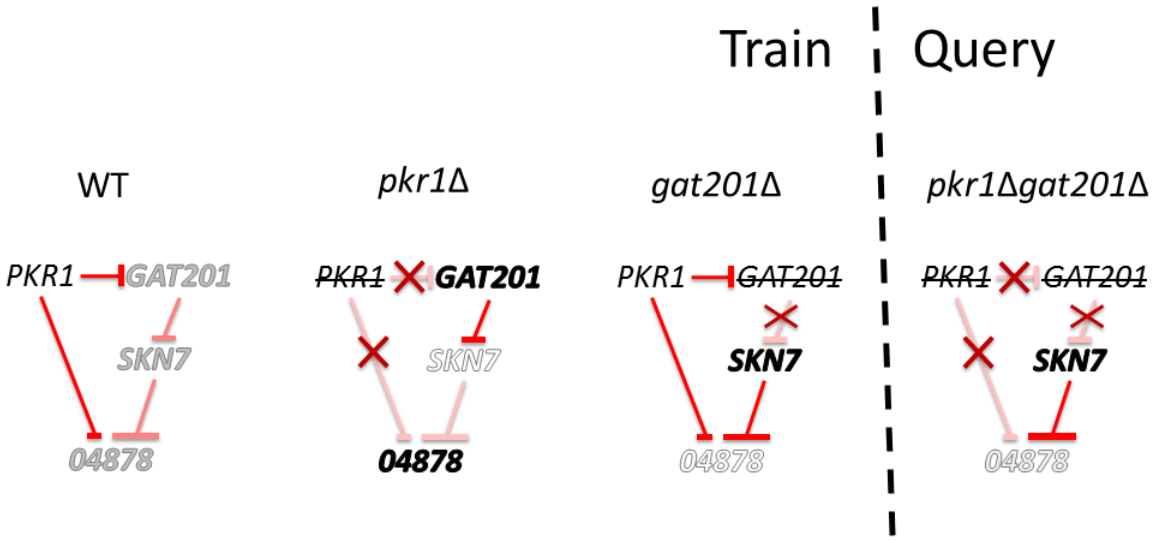
Network simulation pipelined with BART-based expression prediction provides the most accurate and efficient inference of differential expression in the test data.

3.7.3 Zooming into Well Predicted Network Logic

We investigated individual cases where the network simulation model is able to correctly predict the direction of differential expression but the geometric mean fails to do so.

The first case involves the genes *PKR1* (i. e. CNAG_00570, homolog of *BCY1* in *S. cerevisiae*), *GAT201* (i. e. CNAG_01551, a homolog of *GAT2* in *S. cerevisiae*), and CNAG_04878.

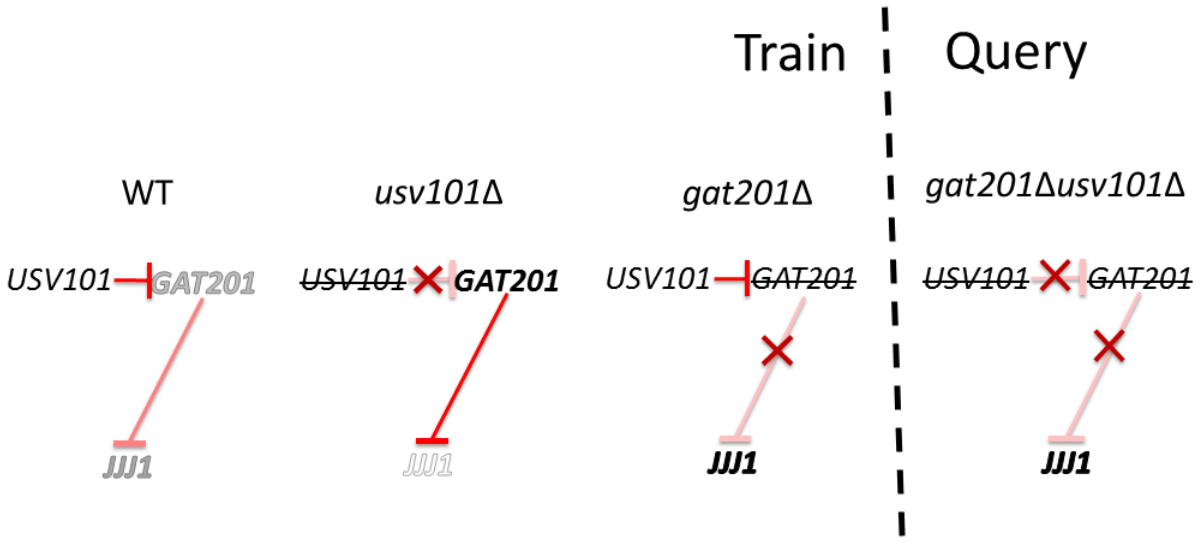
Figure 3.8.: Prediction of CNAG_04878 in *pkr1*Δ*gat201*Δ



In the training data, CNAG_04878 is expressed higher in *pkr1*Δ than in wild type and lower in *gat201*Δ than in wild type. In the test data, CNAG_04878 is expressed significantly lower in the double deletion strain *pkr1*Δ*gat201*Δ than in wild type. Simply taking the arithmetic or geometric mean of these two single deletion strains does not provide a correction prediction of the differential expression of CNAG_04878 observed in the double deletion strain. The network simulation model has inferred CNAG_04878 to be directly repressed by *PKR1* and indirectly activated by *GAT201* (through intermediate repressors including *SKN7*). Moreover, *PKR1* is inferred to repress *GAT201*, forming a consistent feed forward loop. Deletion of the direct represssor *PKR1* alone is explained to both directly de-repress CNAG_04878 and indirectly de-repress it by lowering the level of *SKN7*, but the deletion of both *PKR1* and *GAT201* is predicted to unleash *SKN7* on CNAG_04878 and cause more repression than de-repression.

The second case involves *GAT201*, *USV101* (CNAG_05420, a homolog of *USV1* in *saccaromyces cerevisiae*), and *JJJ1* (CNAG_05538).

Figure 3.9.: Prediction of *JJJ1* in *gat201*Δ*usv101*Δ



Compared to wild type, *JJJ1* is expressed higher in *gat201*Δ and lower in *usv101*Δ. In the double deletion strain *gat201*Δ*usv101*Δ of the test data, *JJJ1* is expressed significantly higher than in wild type. Taking the arithmetic or geometric mean of the single deletion expression levels does not provide a correct prediction about the double deletion strain. The network simulation model has inferred that *USV101* represses *GAT201* which in turn represses *JJJ1*, therefore predicting that the removal of both regulators would induce a higher expression level of *JJJ1*.

3.8 Discussion

Initially, we have attempted to use parameters trained in Chapter 2 to describe regulator-target interactions for the task for predicting expression profiles from perturbation information. Those parameters, intended for the regression problem of predicting target levels from regulators, failed the task of this chapter.

The methods developed in this chapter instead train the parameters to emulate the behavior of the gene regulatory network in the training data, including penalization of instability of the simulated network. The success of this pioneering effort indicates that the network nature of gene regulation relations should be emphasized; the mathematical modeling of this nature provides insight of gene regulation beyond the capability of regression analysis.

The development of methods in this chapter has focused on predicting the effect of novel combination of deletions, with each single deletion strain available in the training data. It would be another interesting task to predict novel deletion strains of genes that have never been deleted in the training data.

4. Discovering Gene Network Edges from Gene Expression

4.1 Background

The emphases in the studies of gene regulatory networks range between two ends: learning quantitative features of the kinetics of gene-protein interaction and inferring topological structure of networks.

In the well studied model species *Saccharomyces cerevisiae*, for example, there are hundreds of transcription factors (TF) and thousands of genes, resulting in a set of $> 10^6$ possible interactions; however, only a small fraction of this set are actual regulation relations. In *S. cerevisiae* and other species as well, it is an important task to infer which regulators interact with which target genes: 1) it helps breaking down the genome scale network into small sub-networks and provides a relative small set of genes of interest for specific metabolism research tasks; 2) It rules out impossible regulator-target interactions, simplifying and facilitating quantitative study of the network; 3) it narrows down the set of genes of interest for genetic modification and engineering challenges.

4.2 Previous Works

4.2.1 Differential Expression Analysis

One of the commonly used method for identifying targets of a regulator gene is differential expression (DE) analysis. To infer the targets of a certain regulator A, genome-wide expression profiles of wild type samples and samples of deletion strain $A\Delta$ are measured. DE methods such as LIMMA [16] and voom [3] are applied to identify the genes whose expression levels show significant differences between the wild type and the deletion strain. These genes are referred to as differentially expressed genes and are inferred to be targets of the regulator A. Quantitatively, a score can be generated from the statistical significance of differential expression, e.g. $1 - (\text{p.value})$ or $|\log(\text{p.value})|$.

DE is a powerful tool for discovering direct regulation targets. However it may also reveal targets of indirect regulations, leaving out the task of distinguishing direct and indirect regulations.

4.2.2 Regression-based Methods

Regression analyses may also be used for network inference: in Inferelator [15], a lasso [18] regression model that explains target levels from regulator levels is used to construct a regulatory network. A similar lasso-based approach is used also in NetProphet [4] as a module. In these methods typically, the lasso is applied on the expression array with normalized logarithm of regulator levels being predictor variables and normalized

logarithm of target levels being response variables. The magnitude of each entry S_{ij} in the coefficient matrix is then used as the regulation score for target i and regulator j.

4.2.3 NetProphet

NetProphet [4] is a network reconstruction method that combines the regulation score generated by differential expression analysis [16] and the score generated by lasso regression analysis [18] in a certain manner to generate a hybrid score. Using the micro-array data of *S. cerevisiae* published in [14], the hybrid score was shown to combine the strength of DE analysis and regression analysis and performed better than both DE and the lasso with a great margin.

4.3 Data and Evaluation Criteria

In this chapter, we develop novel approaches to address the problem of gene network inference. The methods are examined in *Saccharomyces cerevisiae* and compared with existing methods from the literature.

The data on which the methods are applied is the published micro-array expression profiles of wild type and mutant strains in [14] and [2].

In [14], 263 transcription factors were individually deleted and the mRNAs of the resulting strains were hybridized with that of wild type *S. cerevisiae* giving a micro-array measurement of gene expression levels of the mutant strains compared to the wild type.

The micro-array expression data in [2] is also given in the form of comparison with the wild type. This data consists of a much larger collection of single-deletion strains: 1484 genes have been individually deleted, among which 283 belong to the set of genes that we consider as regulators (Appendix F).

Both data sets provide multiple replicates of each mutant strain, enabling differential expression analysis and hence also the application of NetProphet. Besides the statistical significance of differential expression, DE analysis also estimates the fold change of expression level of each gene in each strain with respect to wild type. The fold-change matrices are used for regression analyses in both NetProphet and the new methods proposed in this chapter.

Each method generates a matrix of regulation scores. Denote the index set of genes as G and the index set of transcription factors as C, for each $(i, j) \in G \times C$, a method generates a score S_{ij} whose magnitude $|S_{ij}|$ represents the confidence that target i is directly regulated by regulator j. For some methods, S_{ij} can be positive or negative, representing activation or repression, respectively.

Two standards are used as true labels for examining the quality of the constructed network.

The first set of true labels is the Chromatin Immunoprecipitation (ChIP) evidenced network compiled in *Yeastract* (http://www.yeastract.com/). The gold standard is represented by the matrix (L_{ij}), where $L_{ij} = 1$ represents that gene i is bound by the protein of regulator j according to ChIP evidence and $L_{ij} = 0$ represents otherwise. There are 184 regulators with ChIP-evidenced true labels available.

The second set of true-labels is generated from PWM analysis. PWM (2.74) is a matrix representation of the sequence selectivity of a TF and can be used to calculate the PWM score (2.75), which is an inference of the affinity between the TF and a given sequence. In [4], a set of PWMs of regulators was compiled from previous literature and applied on the promoter sequences of *S. cerevisiae* generating a PWM score for each regulator-target pair. Among the ChIP-available regulators, 116 have PWM scores. For each of these regulators, a cutoff on the PWM score is determined so that the set of targets with a score larger than the cutoff recovers 10% of the ChIP-evidenced targets of that regulator. The targets with scores above the cutoff are considered PWM-evidenced targets of the regulator.

Both of these two sets of true labels are identical to those used in the initial published validation of NetProphet in [4]. Each set of true labels is only available for a subset of the regulators for which we construct the network.

With the inferred regulation score (S_{ij}) being the classifier and (L_{ij}) being the true labels, the network inference task is treated as a binary classification problem with variable threshold on S_{ij}. Precision recall curves (PRC) and precision-versus-number-of-positive-prediction curves may be plotted to reveal the performance of different inference methods.

4.4 New Approach Based on Bayesian Additive Regression Trees

4.4.1 Motivation

This approach is an application of the method described in Subsection 2.6.3, where a BART-based semi-parametric model has been developed to predict target levels from regulator levels.

Each target gene can have multiple regulators and the combination of their effects complicate the problem of investigating the interaction of the target and a single regulator. It naturally raises the question: Is it possible to fix the level of all other regulators while varying the level of one single regulator, therefore enabling the observation of the isolated effect of that regulator on potential target genes?

The *in vivo* realization of this task might be difficult, as the artificial perturbation of one regulator can affect the level of other regulators that are its downstream targets; moreover, biological noise may also be present and contribute to the fluctuation of the levels of all regulators.

However, equipped with reliable methods to predict target levels from regulators levels, this experiment can be simulated *in silico*. We simply construct a query data composed of desired regulator levels and predict the resulted target levels using a model trained on *in vivo* expression data.

4.4.2 Model Training

The model described in 2.6.3 is

$$\begin{cases} \sigma_i^2 \mid B_i, p_i, q_i \text{ i.i.d.} \sim \text{Inv}\Gamma\left(\frac{df_i}{2}, \frac{df_i s_{i0}^2}{2}\right) \\ B_i \mid p_i, q_i \sim \text{BART Prior Distribution} \\ U_{ik} \mid p_i, q_i, \sigma_i^2, B_i \sim \mathcal{N}\left(B_i(\log(x_{1k}), \log(x_{2k}), \ldots), \sigma_i^2\right) \\ Y_{ik} = u^{-1}\left(U_{ik}; p_i, q_i\right) \\ u(y; p, q) := \sqrt{1+p+q}\ln\left(\frac{y+p/2+\sqrt{y^2+py+q}}{1+p/2+\sqrt{1+p+q}}\right) \\ u^{-1}(\xi; p, q) := \left(1+\frac{p}{2}\right)\cosh\left(\frac{\xi}{\sqrt{1+p+q}}\right) + \left(\sqrt{1+p+q}\right)\sinh\left(\frac{\xi}{\sqrt{1+p+q}}\right) - \frac{p}{2} \end{cases} \tag{4.1}$$

where x_{ik} is the level of regulator i in sample k, Y_{ik} is the level of gene i in sample k, df_i and s_{i0}^2 are super parameters, $\sigma_1^2, \sigma_2^2, \ldots$ are unknown noise parameters, and $B_1, B_2, \ldots$ are unknown functions of a non-parametric family referred to as Bayesian Additive Regression Trees. The shape of the transformation $u(\cdot; p_i, q_i)$ depends on tuning parameters p_i and q_i, which are tuned in procedures described in Subsection 2.7.1. In particular, they are set to zero for micro-array data, simplifying $u(\cdot; p_i, q_i)$ to the log-transformation $\ln(\cdot)$.

Expression levels Y_{ik} and x_{ik} are measured as fold changes with respect to wild type: In micro-array data, the sequence library of mutant strains are commonly measured with hybridization with the wild type sequence library and the ratio between the probe signal strength of a gene in the mutant strain and that in the wild type strain is used as the expression level measurement. For RNA-seq data, the direct measurement is in CPM

(counts per million reads) and are converted to fold-change expression levels through the normalization process described in Subsection 2.3.3.

Finally, the posterior distribution of B_i and σ_i^2 are estimated on the training data using the method published in [9], with implementation in the R package *BayesTree*.

The posterior predicative function of Y_{ik} is:

$$u^{-1}\left(B_i(\log(x_{1k}), \log(x_{2k}), \ldots) + \sigma_i \epsilon_{ik}; p_i, q_i\right) \tag{4.2}$$

where $\{\epsilon_{ik}\}_{i=1,2,\ldots,k=1,2,\ldots}$ are independent standard normal variables representing the noise.

Note that both σ_i and ϵ_{ik} are modeled as random variables and B_i is a random function. The posterior distribution of σ_i and B_i can be estimated and the method is capable of generating an estimated posterior distribution of Y_{ik}.

4.4.3 Predicting Isolated Effect of Regulator Levels on Target Levels

We perturb the level of a regulator *in silico* by setting it to its minimum and maximum observed levels in the training data. Denote the index set of training data as T, we define

$$x_{i(\min)} = \min_{k \in T} x_{ik} \tag{4.3}$$

$$x_{i(\max)} = \max_{k \in T} x_{ik} \tag{4.4}$$

Two query matrices of regulator levels are constructed:

$$Q_{(low)} = \begin{pmatrix} x_{1(min)} & 1 & 1 & \cdots \\ 1 & x_{2(min)} & 1 & \cdots \\ 1 & 1 & x_{3(min)} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix} \tag{4.5}$$

$$Q_{(high)} = \begin{pmatrix} x_{1(max)} & 1 & 1 & \cdots \\ 1 & x_{2(max)} & 1 & \cdots \\ 1 & 1 & x_{3(max)} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix} \tag{4.6}$$

Each column vector is a query expression profile of regulators. Since expression levels are measured in fold change with respect to wild type, 1 represents the ideal wild type expression level of any gene. Each query regulator expression profile has one and only one regulator perturbed (being different from its wild type level).

The BART-based expression prediction method will generate an estimation of the posterior distribution of predictions for each query profile, represented by a finite number of posterior samples of BART predictions and t-distributions. The posterior predicted

expression matrices are henceforth denoted as $Y^{(low)}$ and $Y^{(high)}$ for the query matrices $Q^{(low)}$ and $Q^{(high)}$ respectively and

$$U_{ij}^{(low)} = u(Y_{ij}^{(low)}; p_i, q_i) \tag{4.7}$$

$$U_{ij}^{(high)} = u(Y_{ij}^{(high)}; p_i, q_i) \tag{4.8}$$

Note that

$$U^{(low)} \mid training.data$$

$$U^{(high)} \mid training.data$$

$$Y^{(low)} \mid training.data$$

$$Y^{(high)} \mid training.data$$

are all random variables with learned distributions.

Also, define

$$\hat{U}^{(low)} = \mathrm{E}(U^{(low)} \mid training.data) \tag{4.9}$$

$$\hat{U}^{(high)} = \mathrm{E}(U^{(high)} \mid training.data) \tag{4.10}$$

4.4.4 Scoring Regulator-target Interaction

The more $u(Y_{ij}^{(high)}; p_i, q_i)$ deviates from $u(Y_{ij}^{(low)}; p_i, q_i)$, the more likely the expression of gene i could be affected by the expression change of regulator j. Therefore the regulation score is defined as:

$$S_{ij} = \hat{U}_{ij}^{(high)} - \hat{U}_{ij}^{(low)} \tag{4.11}$$

We also explored an alternative option of scoring the regulations, using the posterior probability of differential expression:

$$\begin{aligned} \tilde{S}_{ij} = \max\{ & P(U_{ij}^{(low)} > u(1; p_i, q_i) \mid training.data), \\ & P(U_{ij}^{(low)} < u(1; p_i, q_i) \mid training.data), \\ & P(U_{ij}^{(high)} > u(1; p_i, q_i) \mid training.data), \\ & P(U_{ij}^{(high)} < u(1; p_i, q_i) \mid training.data)\} \end{aligned} \tag{4.12}$$

We found that this scoring system is less capable of precisely revealing network edges than (4.11) (see the result section).

4.5 Combining BART-based Network Construction with NetProphet

In [4], a way to combine the regulation scores generated by differential expression (DE) analysis and lasso regression analysis was defined. The performance of DE score, lasso score, and the combination of both was examined. The combined method was shown to

surpass both DE score and lasso score by a great margin in performance. Motivated by this observation, in this thesis we also look into the performance of combined scores for recovering network edges.

A new procedure of combining different scores is developed here.

In general, the magnitude of the score of an edge corresponds to the inferred likelihood of a network edge. However, different score systems may have different numerical representations of the likelihood of network edges. The scores cannot be interpreted in the same way and are not readily comparable or convertible to each other. For example, DE scores are usually given in $|\log(\text{p.value})|$, lasso scores in the magnitude of linear coefficients, and the BART-based scores developed in the previous section are given in estimated log-fold-change of target level in response to change of regulator level.

The difference in the semantics of the scores renders naïve arithmetic averaging invalid for combining the scores. Moreover, even the scales and distributions of different scoring systems may be distinct, thus directly averaging the scores may arbitrarily assign too much weight to the score system with the larger scale.

One way to circumvent the difference in interpretation of scoring systems is rank-averaging. The regulation score matrix from each scoring method is converted to a matrix of ranks, with smaller rank numbers corresponding to smaller scores. In case of ties, fractional ranking is used to assign equal non-integer ranking numbers to candidates in a tie. After the conversion, rank matrices are averaged to generate a combined regulation score matrix.

The rank-averaging combination scheme is capable of removing difference in semantics between different score systems, but at the price of losing almost all quantitative aspects of every score system: the non-uniform distributional information of each score matrix.

Here, we propose a simple yet robust method for combing score matrices without losing all the distributional information. It is then used to combine NetProphet regulation score [4] with BART-based regulation score developed in the previous section.

Consider two sign-less score matrices A and B (signed score matrices are taken absolute value of before combination). The former is available for regulation relations $(i, j) \in R_A$, the latter available for regulation relations $(i, j) \in R_B$, and $R_A \cap R_B \neq \emptyset$. We define F_A as the empirical cumulative distribution function (ecdf) representing the sample distribution of all available scores in matrix A and similarly F_B is defined as the ecdf of all available scores in B. The quantile functions of A and B are denoted as Q_A and Q_B respectively. For a value b of an entry in matrix B, the function $Q_A \circ F_B$ abstracts b to a percentile in the sample distribution of B and then converts it to a quantity in the sample distribution of A, making it comparable with the entries in A. We define:

$$S_{ij}^{(AB)} = \begin{cases} \frac{A_{ij} + Q_A \circ F_B(B_{ij})}{2} & (i,j) \in R_A \cap R_B \\ A_{ij} & (i,j) \in R_A - R_B \\ Q_A \circ F_B(B_{ij}) & (i,j) \in R_B - R_A \end{cases} \tag{4.13}$$

$S^{(AB)}$ represents the combination generated by first converting both score systems to conform to the distribution of A and then taking the average. Similarly, we define

$$S_{ij}^{(BA)} = \begin{cases} \frac{B_{ij}+Q_B \circ F_A(A_{ij})}{2} & (i,j) \in R_B \cap R_A \\ B_{ij} & (i,j) \in R_B - R_A \\ Q_B \circ F_A(A_{ij}) & (i,j) \in R_A - R_B \end{cases} \tag{4.14}$$

Now both $S^{(AB)}$ and $S^{(BA)}$ are combined scores with consideration of distributional information. The last step is to combine $S^{(AB)}$ and $S^{(BA)}$. Since the scale and sample distribution of entries of $S^{(AB)}$ and $S^{(BA)}$ might still be distinct, they need to be both converted to ranks or percentiles before averaging. Denoting the empirical cumulative distribution functions (ecdf) of them as $F_{S^{(AB)}}$ and $F_{S^{(BA)}}$, the final combined score is defined as

$$S_{ij}^{(\{A,B\})} = \frac{F_{S^{(AB)}}\left(S_{ij}^{(AB)}\right) + F_{S^{(BA)}}\left(S_{ij}^{(BA)}\right)}{2} \tag{4.15}$$

4.6 Results

4.6.1 BART-based Network Construction Competitive with DE and the Lasso

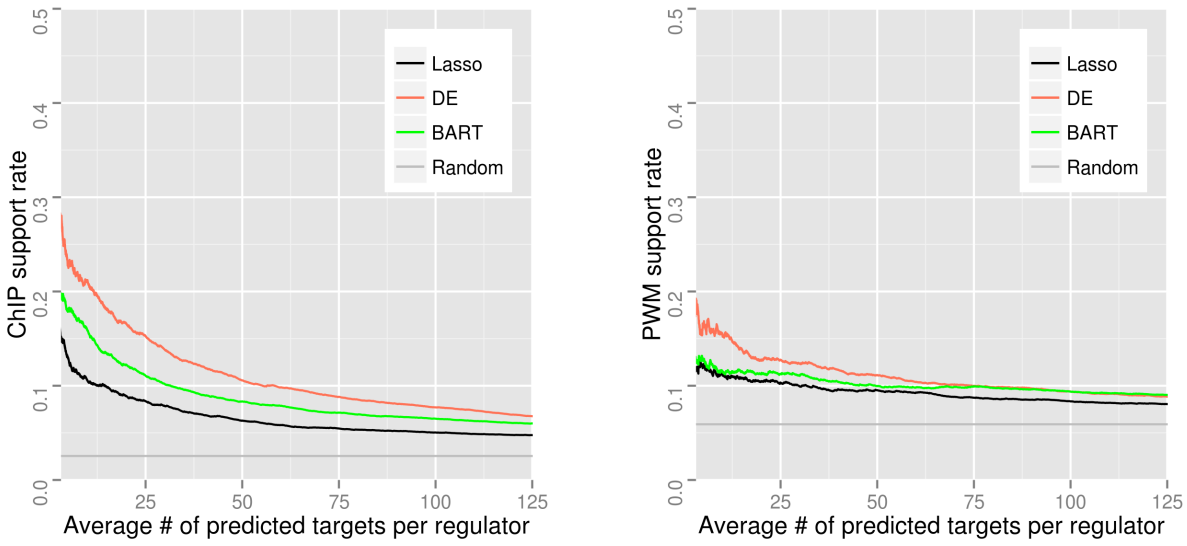
We applied the BART-based network construction method in Section 4.4 to the data set in [14] (henceforth referred to as the Hu Data) and the data set in [2] (henceforth referred to as the Holstege Data) of *S. cerevisiae*. For comparison, NetProphet [4], its differential expression module, and its lasso module are also applied individually on the

two data sets (in the initial validation of NetProphet published in [4], the methods were also applied on the Hu Data).

The 320 genes listed in Appendix F are considered regulators. ChIP evidence and PWM evidence described in Section 4.3 are used to determine whether a predicted network edge is supported by DNA-protein binding experiments.

Each constructed network is given as numeric scores. As the threshold of score varies, the precision, the recall, and other measurements of classification accuracy may vary. We plot ChIP-support rate and PWM-support rate against the average number of targets predicted per regulator:

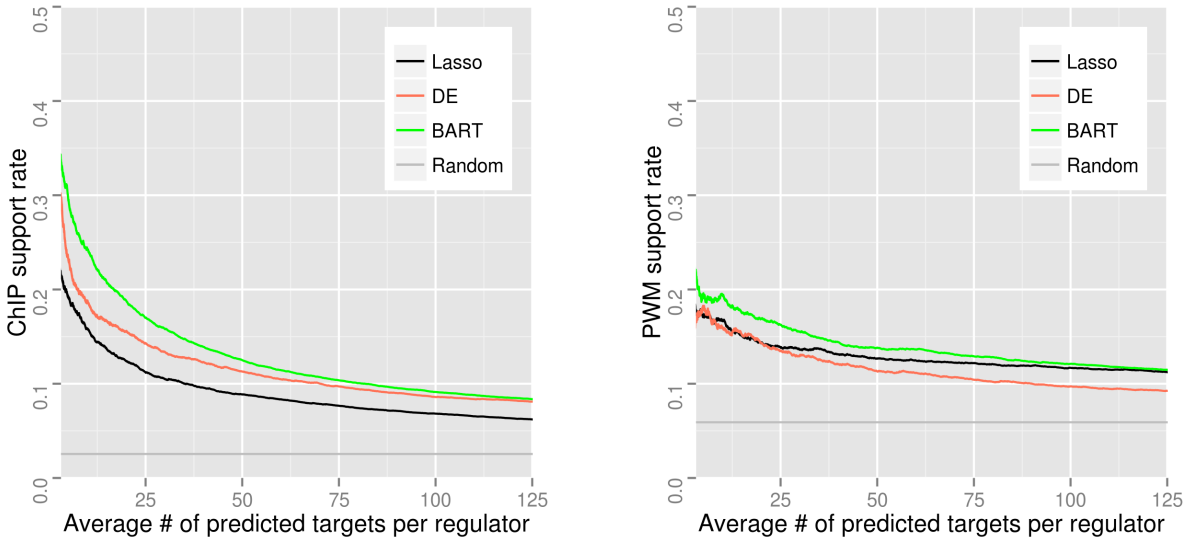
Figure 4.1.: Validation of lasso, DE and BART networks on the Hu Data



(a) ChIP validation

(b) PWM Validation

Figure 4.2.: Validation of lasso, DE and BART networks on the Holstege Data



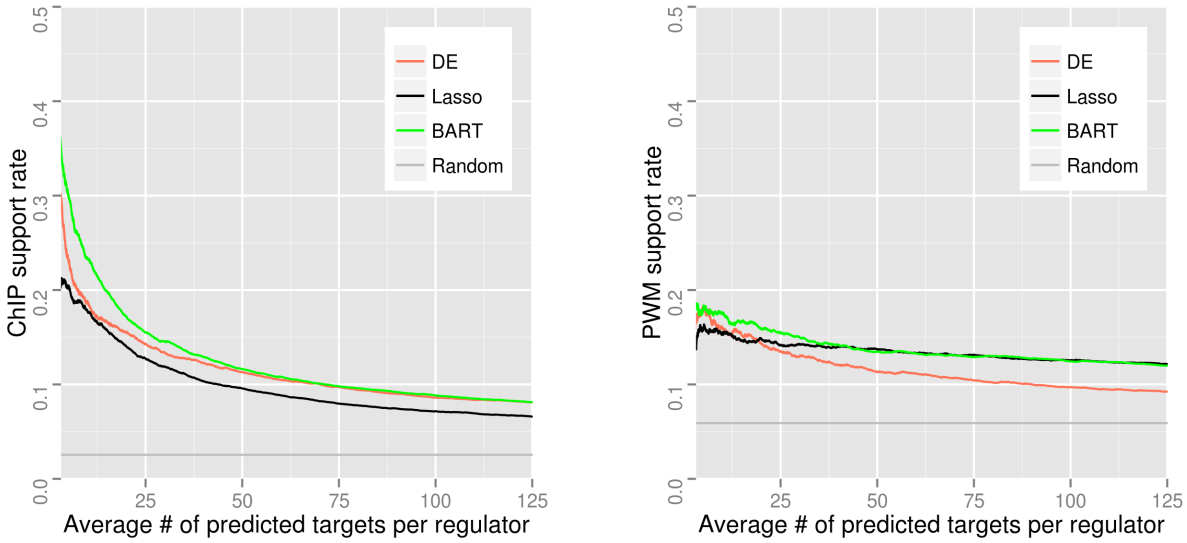
(a) ChIP validation

(b) PWM Validation

In both the Hu Data and the Holstege Data, BART-constructed networks performed better than lasso-constructed networks by a large margin.

The BART-constructed network is more accurate than the DE-constructed network in the Holstege Data but less accurate than the DE-constructed network in the Hu Data. Since the Holstege Data has much larger sample size (1484) than the Hu Data (269) and regression methods may benefit from large sample size, it is possible that sample size may be the major reason that the BART-constructed network is more accurate than the DE-constructed network on the Holstege Data. To test this possibility, we excluded the non-regulator deletion strains of the Holstege Data and ran the lasso and BART on only the regulator deletion strains on the Holstege Data, whose sample size is 283 and comparable to that of the Hu Data.

Figure 4.3.: Validation of lasso, DE and BART networks on Holstege regulator deletion strains



(a) ChIP validation

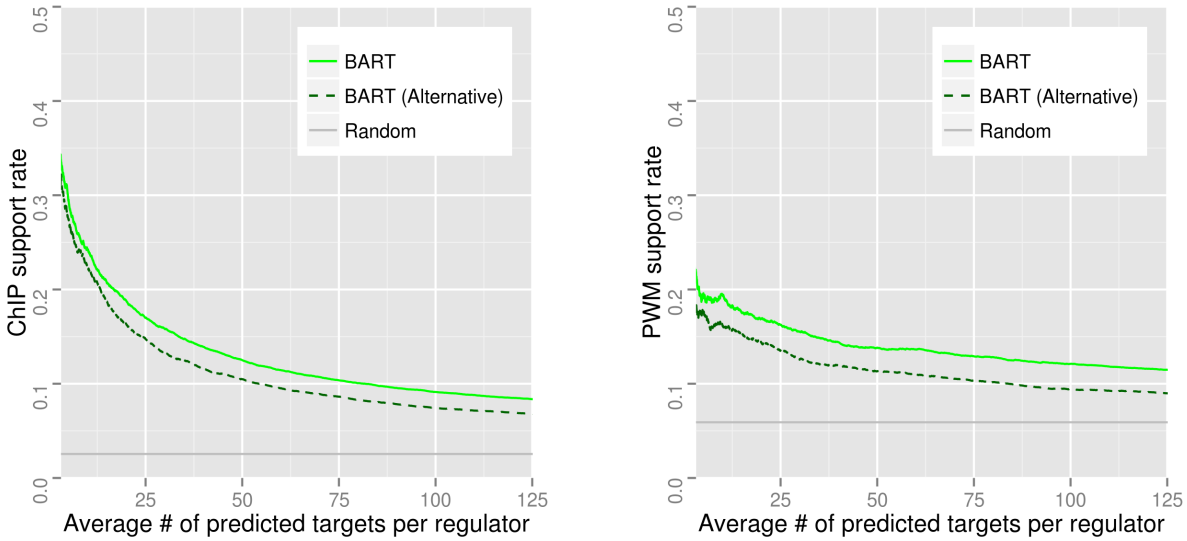
(b) PWM Validation

Despite the shrinkage of sample size, we continue to see the BART-constructed network more accurate than the DE-constructed network.

4.6.2 BART-predicted Fold-change More Informative than Posterior DE Probability

We explored the alternative scoring system provided by BART-based expression prediction defined in (4.12), which is calculated from the posterior probability of differential expression. Surprisingly, it did not perform as good as the log-fold-change score (4.11):

Figure 4.4.: Validation of BART networks on the Holstege Data



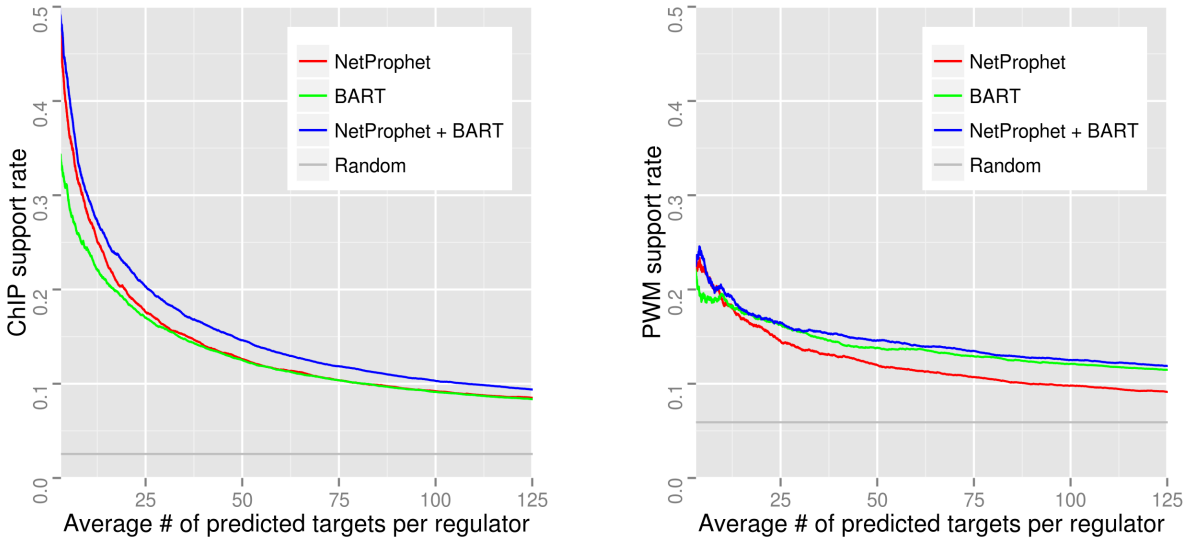
(a) ChIP validation

(b) PWM Validation

4.6.3 Further Accuracy Improvement by Combining BART-based Score and NetProphet Score

Next, we combined the network constructed by the BART-based method and the network constructed by NetProphet using the procedure described in Section 4.5. The resulting network is even more accurate than both NetProphet- and BART-constructed networks:

Figure 4.5.: Combination of BART network and NP network on the Holstege Data



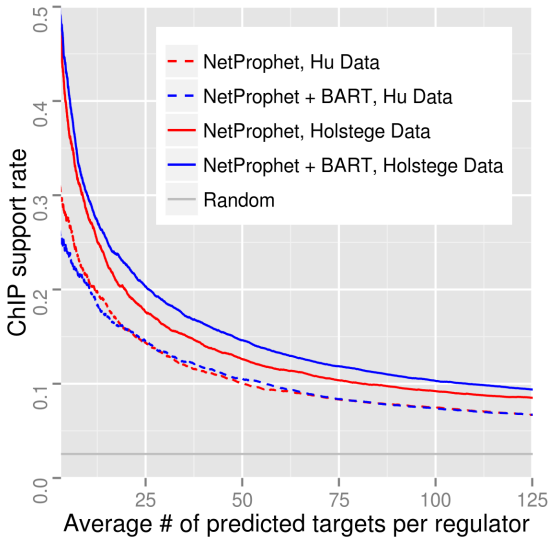
(a) ChIP validation

(b) PWM Validation

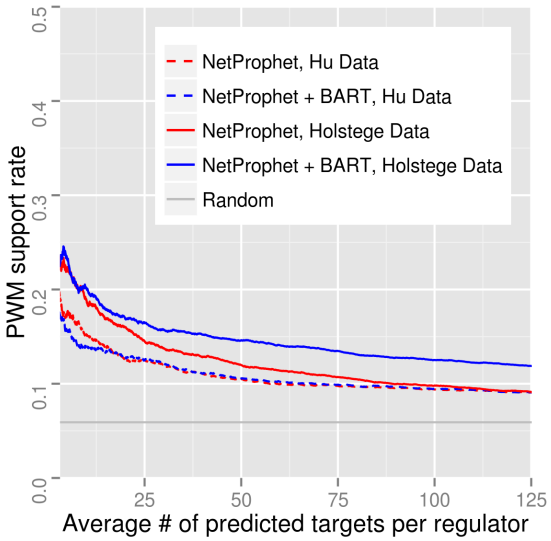
4.6.4 Improvement Provided by New Method and New Data Set

NetProphet inferred a much more accurate network using the Holstege Data than using the Hu Data. Same trend is true for the combination of NetProphet- and BART-constructed networks. We attribute this to the high precision and large sample size of the Holstege Data. The validation is done on the set of regulators equipped with both ChIP-evidenced and PWM-evidenced true labels:

Figure 4.6.: Network recovery on the Hu Data and the Holstege Data



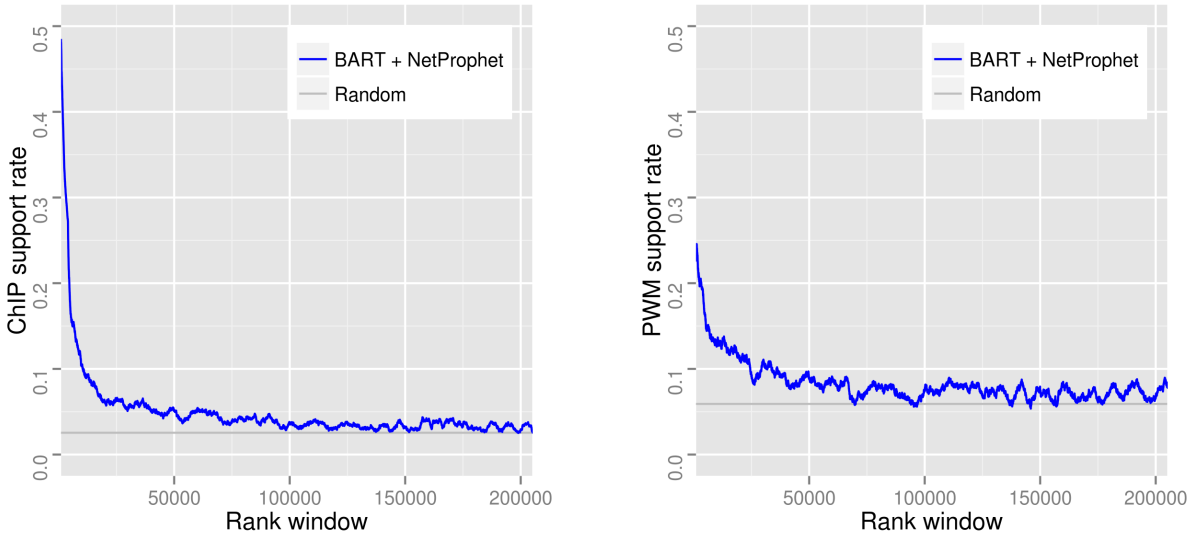
(a) ChIP validation



(b) PWM Validation

In addition, we made a sliding window of size 4000 in the sorted rank list of the combined scores and examined the ChIP-support rate and the PWM-support rate of each window:

Figure 4.7.: Precision of network recovery by rank window



(a) ChIP validation

(b) PWM validation

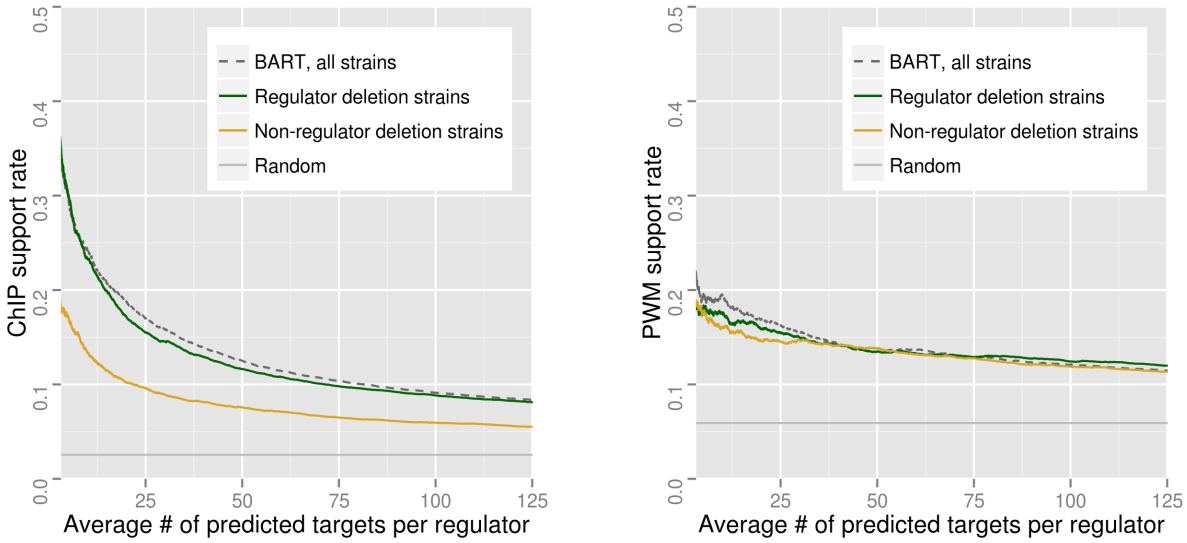
Each integer point x on the horizontal axis represent the rank window $[x - 3999, x]$ (or $[1, x]$ if $x < 4000$). Up to the 50,000th in the rank list, recovered edges are supported by ChIP and PWM more than chance.

4.6.5 More Insight Provided by Regulator Deletion Strains than Non-regulator Deletion Strains

The Holstege Data [2] consists of the single deletion strains of 1484 genes, 283 of which intersect with the genes that we consider as regulators (Appendix F) in this research. We partitioned the Holstege Data into the set of regulator deletion strains and the set of non-regulator deletion strains and asked which set of strains contributes more to the

predictive power of regression-based network recovery. We applied BART-based network construction and the lasso individually on these two sets of strains to answer the question:

Figure 4.8.: BART-based network recovery on subsets of the Holstege Data

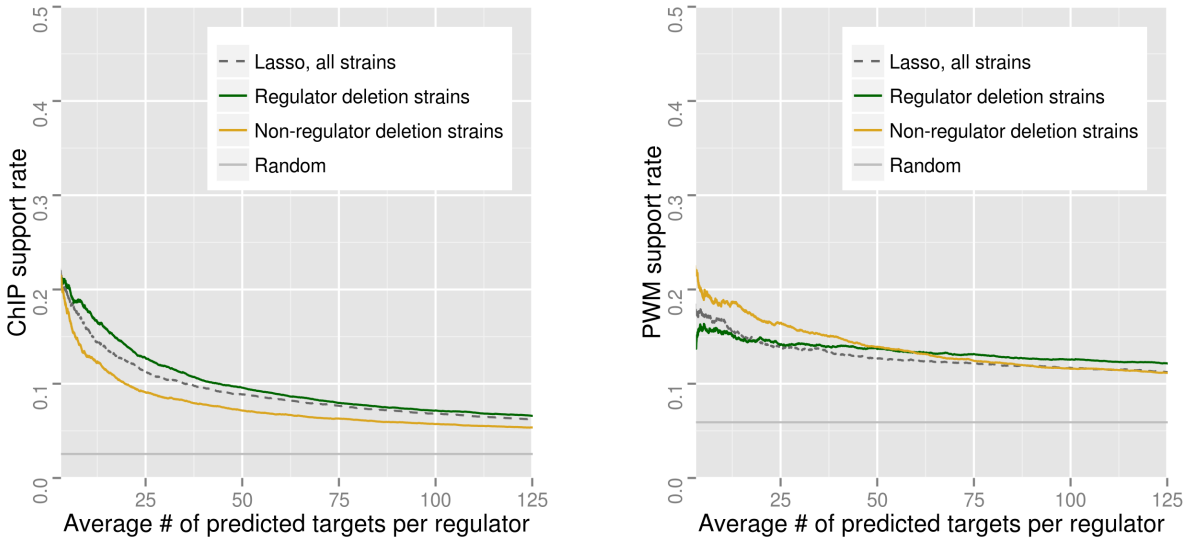


(a) ChIP validation

(b) PWM validation

The non-regulator deletion strains did inform BART better than random guesses. However, despite its large sample size (1201), the network learned from it is less accurate than the network learned from the regulator deletion strains, which has a much smaller sample size (283). This reaffirmed the great value of regulator deletions in network study, not only for enabling DE analyses but also for empowering regression analyses.

Figure 4.9.: Lasso-based network recovery on subsets of the Holstege Data



(a) ChIP validation

(b) PWM validation

As for lasso-based network recovery, ChIP and PWM validation disagree on whether regulator deletion strains or non-regulator deletion strains contribute more to the predictive power. Under ChIP validation, the regulator deletion strains provided a network more accurate even in comparison with the one constructed using the entire data set, which in turn is better than the network constructed from the non-regulation deletion strains. Under PWM validation, the order of performance is the opposite:the non-regulator deletion strains provided a network more accurate than both regulator deletion strains and the entire data set.

4.7 Discussion

In this chapter, we have developed a new method for recovering network edges from expression profiles. Both our new method and the usage of the latest large scale single deletion expression array [2] of *Saccharomyces cerevisiae* have contributed to the improved accuracy of network recovery.

Our new method uses the quantile combination of the predictions provided by previously published method NetProphet and the novel BART-based expression prediction method developed in this dissertation. The latter produces a network recovery more accurate than the lasso and even in some cases than differential analysis. Finally the combination of the BART-based method and NetProphet provided unprecedented accuracy.

The accuracy also owes to the high-quality of the latest expression array data published in 2014 [2], which we refer to as the Holstege data. Another comprehensive regulator deletion expression array of *S. cerevisiae* which preceded [2] was published in 2007, known as the Hu data [14]. The progress in micro-array technology over the years has enabled a much higher precision in measurement. In addition, the Holstege data contains much more mutant strains (> 1400) than the Hu data (≈ 260). Both the precision and the size of the Holstege data have helped our task of network recovery.

By assessing the network recovered from the subset of the Holstege data which contains only the 283 regulator deletion strains, we have confirmed that measurement precision contributed to the accuracy of network recovery more than data size did. Moreover,

regulator deletion strains provide more strength to network recovery than non-regulator deletion strains.

5. Discussion

5.1 Prospects

5.1.1 Non-parametric and Semi-parametric Bayesian Models for Gene Regulation

In Chapter 2, we have developed a BART-based semi-parametric model for predicting target expression levels from regulator levels and shown its robustness applied on different data sets. This suggests the potential of non-parametric and semi-parametric Bayesian methods in the field of gene regulatory network modeling.

Non-parametric Bayesian methods do not provide a direct interpretation of the mechanistic aspect of gene regulations. On the other hand, modeling approaches of gene regulation ranges from the highly abstract end to the highly mechanistic end.

Literature has seen various modeling approaches that emphasize the representation of the molecular and thermodynamic aspects of gene regulations, such as sequence affinity and cooperative and competitive DNA-binding. Converting mechanistic hypotheses to accurate quantitative and predictive models is not a trivial task. In fact, given the large amount of noise contributed by unknown factors, the theoretical correctness of an mech-

anistic explanation may not always be clearly reflected in quantitative precision. But in the long term, it is the compilation of mechanistic stories that defines our understanding of the biological processes and dictates our ability to quantitatively describe and predict their behavior.

It will be an interesting endeavor to bridge between mechanistic emphases and the predictive power of highly abstract semi-parametric and non-parametric models. Equipped with a powerful tool to predict target levels from regulators levels, it is possible to draw information from the trained models to indirectly hint possible mechanistic hypotheses. For example, constructing *in silico* query profiles of regulator levels that are experimentally unavailable and predicting their effect on target genes may allow the study of isolated or combinatorial effect of regulator perturbations, bringing our knowledge one step closer to their mechanistic nature; in a trained model of Bayesian Additive Regression Trees, the frequency of two explanatory variables appearing in a same decision tree might suggest the likelihood of cooperative or competitive regulation; isolating effects of individual regulators on target genes and searching for unusual response curves (such as non-monotonic curves) might provide a short list of complicated regulation relations; et cetera.

5.1.2 Expression Prediction with the Aid of Sequence Analysis

In Subsection 2.9.3 and Section 2.9, we have discussed various models for predicting expression levels from regulator levels and sequence affinity information. These models, compared to the simple regression models (2.8), (2.10), and (2.11), have much fewer parameters hence potentially more resistance to overfitting. However, the time complexity

of fitting and applying some of these models is several magnitudes higher than that of the simple regression models. The viability of these methods for genome-wide analysis will depend on the development of computational technology.

The value of these models lies not only in their ability to predict expression from sequence affinity, but also in the reverse way: assessing the accuracy of sequence affinity information based on their ability to explain expression data. Up to date, the majority of protein-sequence affinity knowledge such as PWM [17] and DNA accessibility has been gained from experimental data measuring protein binding locations (e. g. ChIP-seq, DNase-seq). Given the availability of large amount of expression data in the literature, a method that systematically transfers expression information to sequence affinity knowledge will provide new perspectives on gene-protein interaction.

5.1.3 Network Model for Network

The methods and results in Chapter 3 suggest that gene regulatory networks, especially the sub-networks composed of regulator genes, need to be understood as an entity that is capable of sustaining its state and responding to environmental or genetic perturbations through robust self-adjustment. Not failing its nomenclature, the network aspect of GRN needs to be emphasized in mathematical modeling.

In [13] which also aims at the task of predicting gene expression profiles from genotypes, a few number of "seed genes" were held accountable as the source of variance in the data; in [15], [11], and [4], gene regulatory networks were constructed by solving the regression problem where target expression levels are response variables and regulator levels are

explanatory variables. It is common in expression-based network studies to consider the network as a hierarchy of causal relations, where transcription factors are on the top the of causal chains. There has been less work in modeling the mutual interactions between regulators and reconstructing these mutual interactions from observed network behavior.

The ability of an *in vivo* network to adjust its gene expression levels to a certain state given the genetic or environmental perturbations and stabilize around the final state is a fairly non-trivial property. It is a strong requirement for a network model to exhibit this same property and network simulation is necessary for enforcing this requirement. Given a space of hypothetical networks, the expression data, the ability to simulate and predict the behavior of a hypothetical network, and proper computational approaches, one can construct a network model supported by the experimental data and the network behavior that it implies. The network simulation approach differs greatly from regression analysis and can provide new insights to the dynamics of gene regulatory networks.

5.1.4 Incorporating Semi-parametric Modeling to Network Simulation

Considering the power of semi-parametric models shown in Chapter 2 and the success of network simulation in Chapter 3, we find it an enticing direction to incorporate Bayesian non-parametric and semi-parametric modeling into gene regulatory network simulation. For example, the target-regulator interaction function F (3.27) in the network simulation framework may be replaced by a non-parametric function with Bayesian prior distribution.

It has to be admitted that both of the parametric descriptions (3.12) and (3.14) are highly simplified forms of actual target-regulator relations. Aside from these parametric

descriptions, non-parametric or semi-parametric families of functions, especially ensemble functions, give an alternative option for modeling complicated quantitative relations.

The usage of non-parametric functions might endow the ability to capture non-linear and non-monotonic target-regulator relations and unknown combinatorial effects of regulators and help avoiding the issues caused by over-extrapolating which is common to the simple parametric models.

Non-parametric approaches in network simulation may impose computational challenges in spatial and temporal complexity of the optimization process as well as convergence issues that is to be addressed by the developing theory of Bayesian non-parametric statistics and requires careful modeling and tuning.

5.1.5 Incorporating Advanced Network Structure Inference to Network Simulation

Part of the training process of the simulated networks in Chapter 3 may be preceded by and benefit from narrowing the set of possible network edges to a small subset. The edge selecting process could both protect against overfitting and reduce time complexity. On the other hand, we have developed methods in Chapter 4 that recovers network edges from expression data with a great performance. It is possible to use the methods developed in Chapter 4 to select network edges for the network simulation task.

5.1.6 Diversity Provides Strength

Throughout the studies, it has been repeatedly seen that combined models exhibit enhancement in performance. Bayesian Additive Regression Trees per se is an ensemble of weaker learners, which interestingly benefits from the absence of individual strong learners in the ensemble; in Chapter 4, the combination of the BART-based network reconstruction approach and NetProphet results in a method that is stronger than each of the components; moreover, NetProphet alone is a combination of differential expression analysis and lasso regression.

It is important and desirable to uncover an accurate and consistent mechanistic explanation of the biological process that we study; but until a perfect mechanistic model is reached, the combination of multiple models, even the ones based on apparently contradicting assumptions, remains a practical way to complement the unique limitations of different modeling approaches.

5.2 Conclusion

In this dissertation, various machine learning and Bayesian-statistical approaches have been studied for mapping and modeling gene regulatory networks.

In Chapter 2, a competitive and reliable semi-parametric model based on Bayesian Additive Regression Trees has been developed to predict target expression levels from regulator levels and various modeling approaches that relates sequence affinity to gene expression levels in theory have been discussed.

In Chapter 3, we took on the task of predicting expression levels in strains of novel combination of genetic perturbations and have shown that network simulation is an effective approach for studying and predicting the propagation of genetic perturbation effects. With the new method developed, we succeeded to predict the expression level changes in double deletion strains of *C. neoformans* quantitatively and qualitatively.

Chapter 4 has shown that our ability of discovering gene interactions and distinguishing between direct and indirect regulation relations has been greatly improved both by our novel methods and by the currently available high-quality data in the field of bioinformatics; in addition, we further validated the value and performance of methods developed earlier in Chapter 2.

This dissertation has revealed the great potential of theories and methods of modern machine learning and Bayesian statistics in the research of gene regulatory networks and developed powerful novel methods to solve problems in both the more explored and the uncharted territories of bioinformatics. With advanced mathematical tools and strong computational power, we have significantly improved our efficiency and precision in extracting qualitative and quantitative features of gene regulation networks from the ocean of data.

APPENDICES

A. Previous Methods on Expression Prediction from Genotype

The method in [13] is recapitulated here.

Consider an expression array matrix D, where the (i, k) entry denotes the log-expression-level of gene i in strain k and strain 1 stands for wild type. The construction of the model in [13] requires decomposition of D into an "influence" matrix X and a "genotype" matrix G so that $D \approx XG$.

The major regulators potentially responsible for causing most of variances in the expression array were referred to as the "seed genes". In the study of [13] particularly, the five transcription factors that were deleted in the training data were considered seed genes.

The $(i, 1)$ entry of X represents the background expression level of gene i and the $(i, j+1)$ entry represents the influence (direct or indirect) of seed gene j on the expression level of target gene i.

All entries in the first row and the first column of G are set to 1. The $(j+1, k)$ entry of G represents the "activity level" of seed gene j in strain k. "Activity level" was described to be an abstract representation of the overall magnitude of influence of that seed gene on other genes in a certain strain. Naturally, $G_{j+1,k}$ is set to zero if seed gene j is deleted in strain k.

To simplify the computation, expression array D is singular-value-decomposed into uvw^T, where u and w are orthogonal matrices and v is a diagonal matrix (not necessarily square) of decreasing non-negative diagonal entries. Denoting the index set of seed genes as J and the number of seed genes $|J|$, D is then approximated by $\tilde{D} = \tilde{u}\tilde{v}\tilde{w}^T$, where $\tilde{u}$ is the first $|J|+1$ columns of u, $\tilde{v}$ is the top-left $(|J|+1) \times (|J|+1)$ block of v, and $\tilde{w}$ the first $|J|+1$ columns of w. The approximation is based on the authors' belief that the $|J|+1$ largest singular values of matrix v accounts for most of the variance caused by the $|J|$ seed genes and the presence of a background.

The next step in the method was to perform a least square fit $\tilde{v}\tilde{w}^T \sim xG$ solving for matrices $x_{(|J|+1)\times(|J|+1)}$ and G, where G is used as the fore-mentioned genotype matrix and $\tilde{u}x$ is used as the influence matrix.

The least square fit $\tilde{v}\tilde{w}^T \sim xG$ is done with the constraint that 1) the first row and first column of G are all equal to 1.0 and 2) $G_{j+1,k} = 0$ if seed gene j is deleted in strain k.

The number of unknown entries in matrix G is therefore $|J|(|K|-1) - \sum_{k\in K} |\Delta_k|$, where K denotes the index set of strains and $|\Delta_k|$ the number of deleted genes in strain k. The number of unknown entries in matrix x is $(|J|+1)^2$.

For the least square fit $\tilde{v}\tilde{w}^T \sim xG$, the total degree of freedom of the unknowns is thus $(|J|+1)^2 + |J|(|K|-1) - \sum_{k\in K} |\Delta_k|$. The number of "observations" in matrix $\tilde{w}$ is $(|J|+1)|K|$. Without consideration of degeneracy, the number of observations has to be

at least as large as the degree of freedom of the unknowns in order to guarantee a unique best fit, i. e. requiring that

$$(|J|+1)^2 + |J|(|K|-1) - \sum_{k\in K} |\Delta_k| \leq (|J|+1)|K| \tag{A.1}$$

That is

$$|J|^2 + |J| + 1 - \sum_{k\in K} |\Delta_k| \leq |K| \tag{A.2}$$

$$|J|^2 + |J| + 1 \leq \sum_{k\in K} (|\Delta_k| + 1) \tag{A.3}$$

As the number $|J|$ of seed genes goes bigger, this implies that $\sum_{k\in K}(|\Delta_k|+1)$ has to have at least a magnitude of $\mathcal{O}(|J|^2)$, which is impossible to achieve when the majority of the strains are single deletions of seed genes, as $|K| \approx \mathcal{O}(|J|)$ and $|\Delta_k| \approx \mathcal{O}(1)$.

To predict expression levels in strains of novel deletions or novel deletion combinations, it is necessary to construct a genotype matrix $\hat{G}$ for the novel query strains and an influence matrix $\hat{X}$ for each query strain.

To construct the influence matrix $\hat{X}$ for the query strain, the method relies on prior knowledge of the regulatory network structure of the studied species. Note that the $(i, j+1)$ entry of X describes the effect of seed gene j on the target gene i. If the deletion in the query strain cuts off all regulator pathway(s) that transmit the regulatory signal from seed gene j to target gene i, $\hat{X}_{i,j+1}$ is set to zero; otherwise, $\hat{X}_{i,j+1} = X_{i,j+1}$.

To construct the novel genotype matrix $\hat{G}$, the authors assumed an underlying pattern in the genotype matrix G of the training strains. Denoting column k of G as

$$\begin{pmatrix} 1 \\ \\ \vec{g}_k \end{pmatrix} \tag{A.4}$$

the activity levels of seed genes in strain k is then represented by $\vec{g}_k$. In wild type for example, the author assumed that the wild type genotype vector $\vec{g}_1$ satisfies that

$$\begin{pmatrix} 1 \\ \\ \vec{g}_1 \end{pmatrix} \approx A \begin{pmatrix} 1 \\ \\ \vec{g}_1 \end{pmatrix} \tag{A.5}$$

where

$$A = \begin{pmatrix} 1 & 0 & \dots & 0 \\ & & & \\ \vec{g}_0 & & m & \end{pmatrix} \tag{A.6}$$

and $m_{|J|\times|J|}$ is an unknown matrix whose (i, j) entry describes the effect that seed gene j exerts on the activity level of seed gene i; $\vec{g}_0$ is an unknown basal activity vector. The diagonal entries of m are assumed to be zero. (A.5) is equivalent as saying that the first column of G is an eigenvector of the matrix A with eigenvalue 1.

(A.5) can be rewritten as

$$\vec{g}_1 \approx m\vec{g}_1 + \vec{g}_0 \tag{A.7}$$

Generally, for the column vector $\vec{g}_k$, if genes $j_1, j_2, \ldots$ are deleted in strain k, rows $j_1, j_2, \ldots$ of g_k are by definition equal to zero. One can define a matrix D_k such that $D_k A$ extracts all the rows of any matrix A except rows $j_1, j_2, \ldots$. With this notation, (A.7) may be generalized to

$$D_k \vec{g}_k \approx D_k(m\vec{g}_k + \vec{g}_0) \tag{A.8}$$

Note that $\vec{g}_k = D_k^T D_k \vec{g}_k$. Therefore (A.8) can be reshaped as

$$D_k \vec{g}_k \approx D_k(mD_k^T D_k \vec{g}_k + \vec{g}_0) \tag{A.9}$$

$$(I - D_k m D_k^T) D_k \vec{g}_k \approx D_k \vec{g}_0 \tag{A.10}$$

$$D_k \vec{g}_k \approx (I - D_k m D_k^T)^{-1} D_k \vec{g}_0 \tag{A.11}$$

The authors then solved the least square fitting problem $D_k \vec{g}_k \sim (I - D_k m D_k^T)^{-1} D_k \vec{g}_0$ using the columns of genotype matrix G of the training data solving for m and g_0 under the constraint that all diagonal entries of m are zero. They then used (A.11) again with the learned m and g_0 to construct the columns of the genotype matrix $\hat{G}$ of the test data.

B. Dependence of Performance of kNN-based Expression Prediction on Parameter k

Expression levels of target genes are predicted from expression levels of regulator genes using an method based on k-nearest-neighbors. The parameter k stands for the number of neighbors used for prediction in each query profile. Here options of $k = 1, 2, \ldots 30$ are probed and their corresponding performance in R^2 is listed below.

Data	*C. neoformans*		*S. cerevisiae*, #1		*S. cerevisiae*, #2	
Training set	WT and singleΔ		Regulator deletion		Non-regulator deletion	
Test set	DoubleΔ		Non-regulator deletion		Regulator deletion	
Method	kNN.log	kNN.linear	kNN.log	kNN.linear	kNN.log	kNN.linear
	R^2	R^2	R^2	R^2	R^2	R^2
$k = 1$	0.14	-0.06	0.13	0.09	-0.08	-0.12
2	0.23	0.16	0.20	0.18	0.14	0.20
3	0.25	0.19	0.22	0.20	0.20	0.24
4	0.26	0.20	0.22	0.21	0.22	0.25
5	0.26	0.20	0.22	0.22	0.23	0.26
6	0.26	0.20	0.22	0.22	0.23	0.26
7	0.26	0.20	0.21	0.22	0.24	0.27
8	0.25	0.20	0.21	0.21	0.24	0.26
9	0.25	0.20	0.21	0.21	0.24	0.26
10	0.25	0.20	0.20	0.21	0.24	0.26
11	0.24	0.20	0.20	0.21	0.24	0.25
12	0.24	0.20	0.20	0.21	0.24	0.25
13	0.24	0.20	0.19	0.21	0.23	0.25
14	0.24	0.19	0.19	0.20	0.23	0.25
15	0.23	0.19	0.19	0.20	0.23	0.25
16	0.23	0.19	0.18	0.20	0.23	0.24
17	0.23	0.19	0.18	0.20	0.23	0.24
18	0.22	0.19	0.18	0.20	0.23	0.24
19	0.22	0.19	0.17	0.19	0.22	0.24
20	0.22	0.19	0.17	0.19	0.22	0.23
21	0.22	0.18	0.17	0.19	0.22	0.23
22	0.22	0.18	0.17	0.19	0.22	0.23
23	0.21	0.18	0.16	0.19	0.22	0.23
24	0.21	0.18	0.16	0.19	0.21	0.23
25	0.21	0.18	0.16	0.19	0.21	0.23
26	0.21	0.18	0.16	0.18	0.21	0.22
27	0.21	0.18	0.16	0.18	0.21	0.22
28	0.20	0.17	0.15	0.18	0.21	0.22
29	0.20	0.17	0.15	0.18	0.20	0.22
30	0.20	0.17	0.15	0.18	0.20	0.22

C. Prediction-observation Plots of Target Gene Levels Predicted from Regulator Levels

Figure C.1.: Prediction of *C. neoformans* target levels from regulator levels by the log-linear model

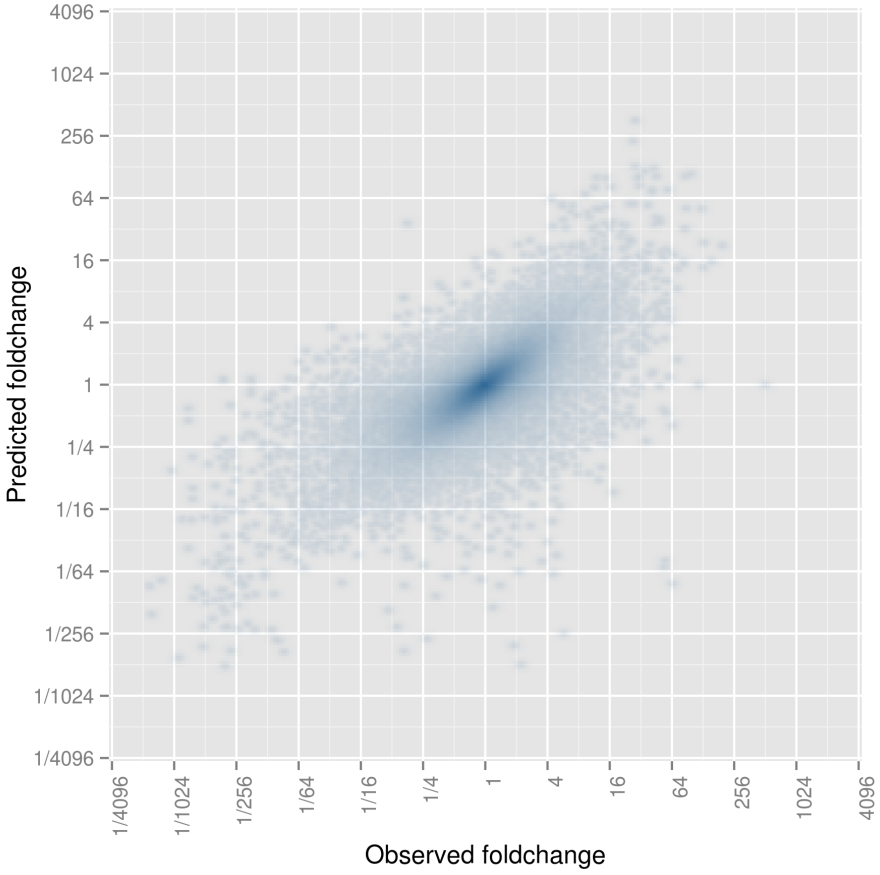


Figure C.2.: Prediction of *C. neoformans* target levels from regulator levels by the exponential model

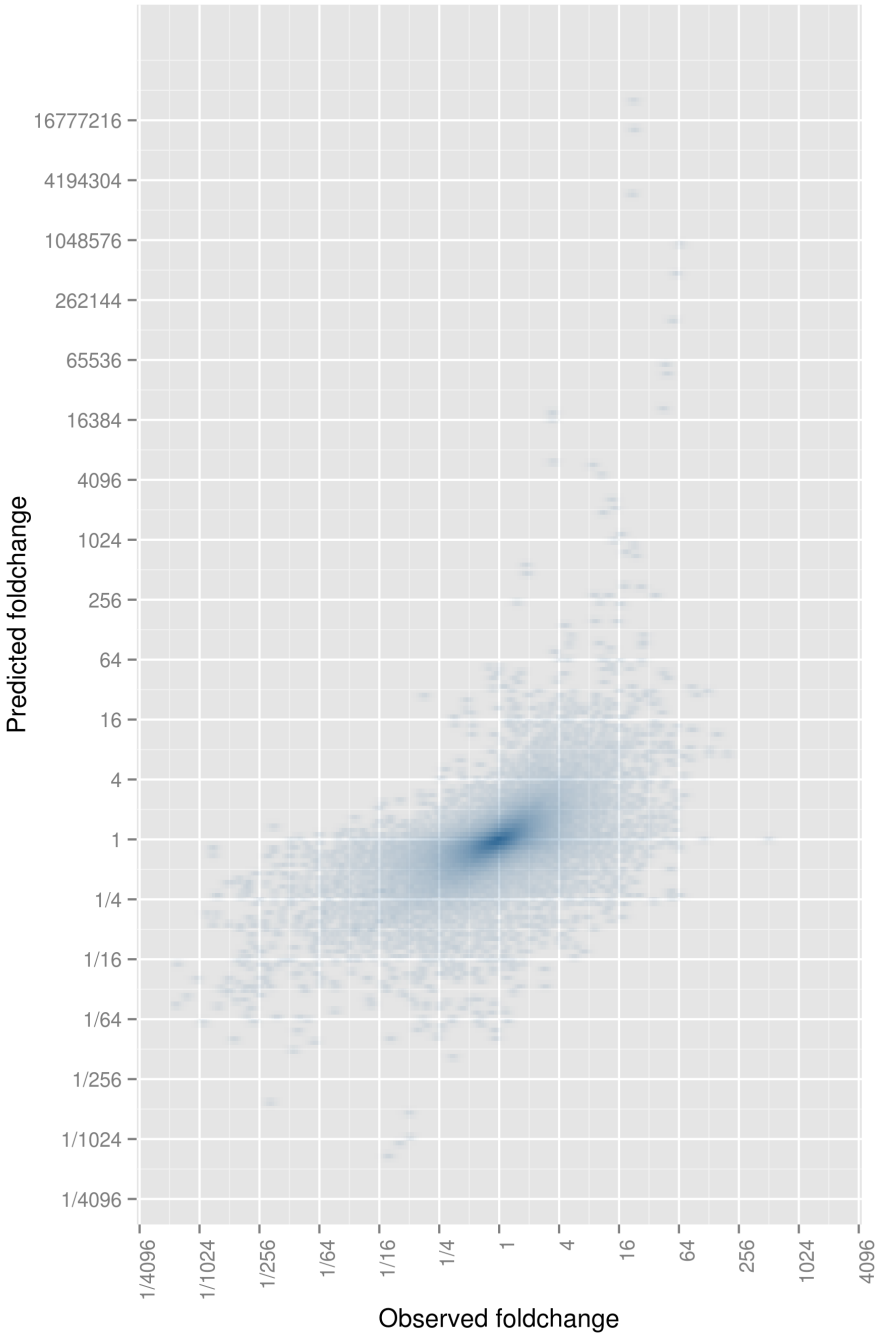


Figure C.3.: Prediction of *C. neoformans* target levels from regulator levels by the log-linear-exponential model

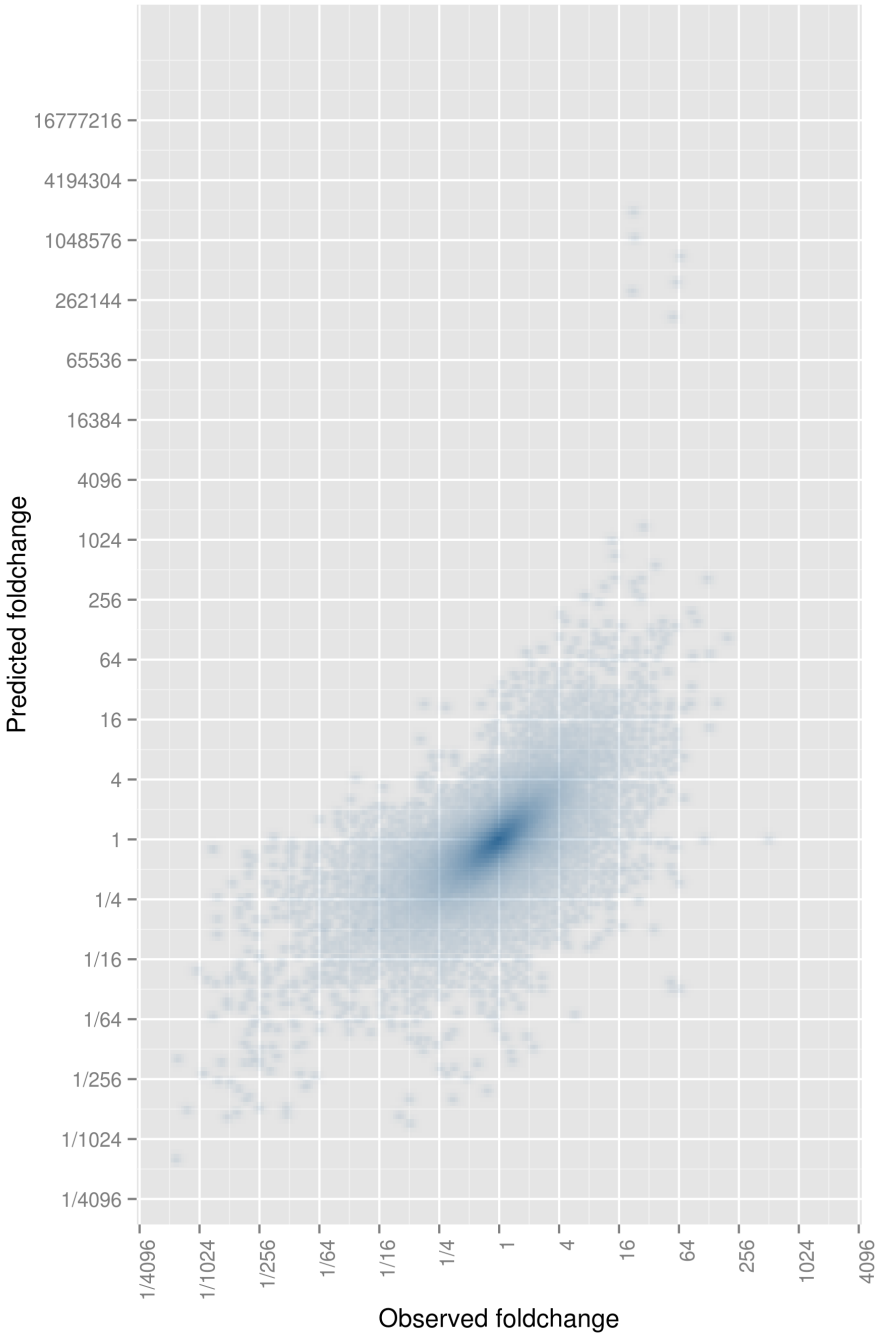


Figure C.4.: Prediction of *C. neoformans* target levels from regulator levels by the BART-based semi-parametric model

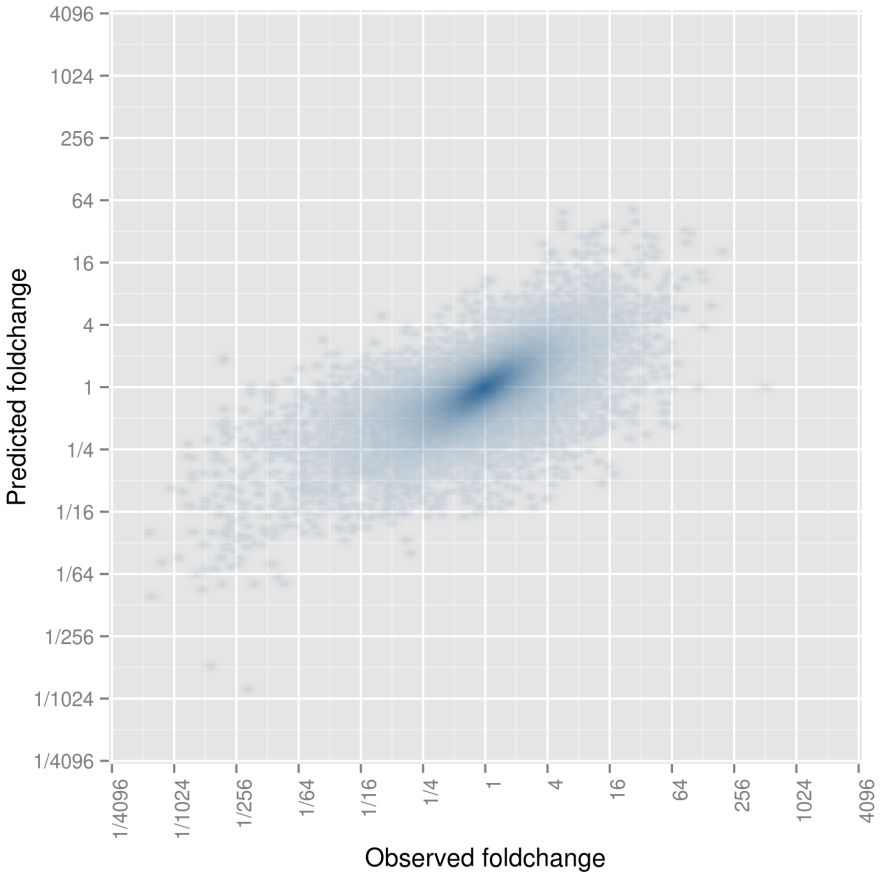
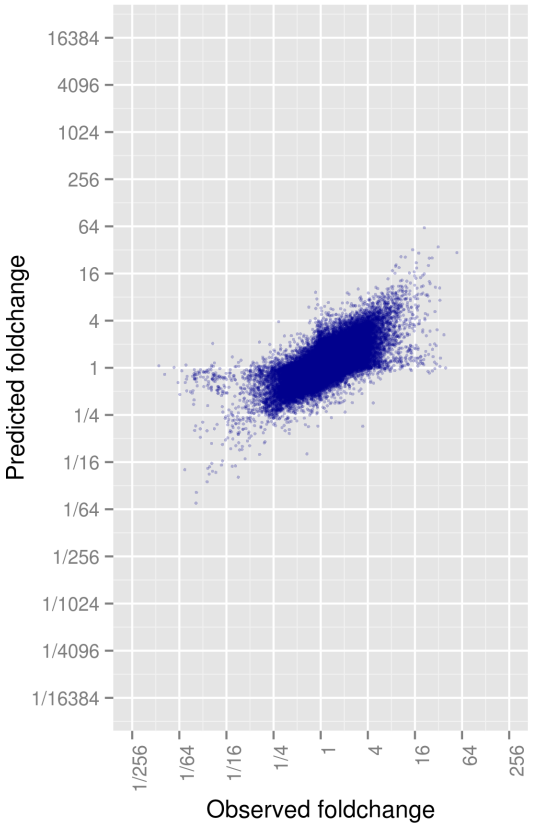
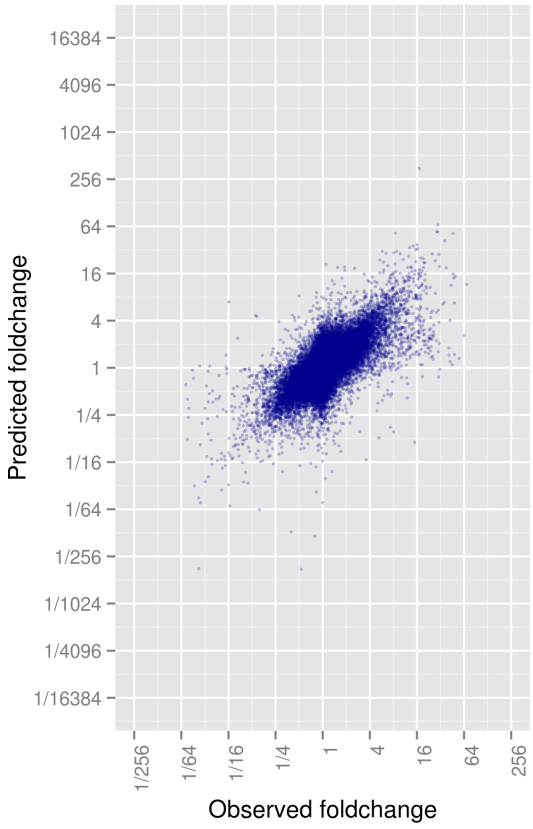


Figure C.5.: Prediction of *S. cerevisiae* target levels from regulator levels by the log-linear model

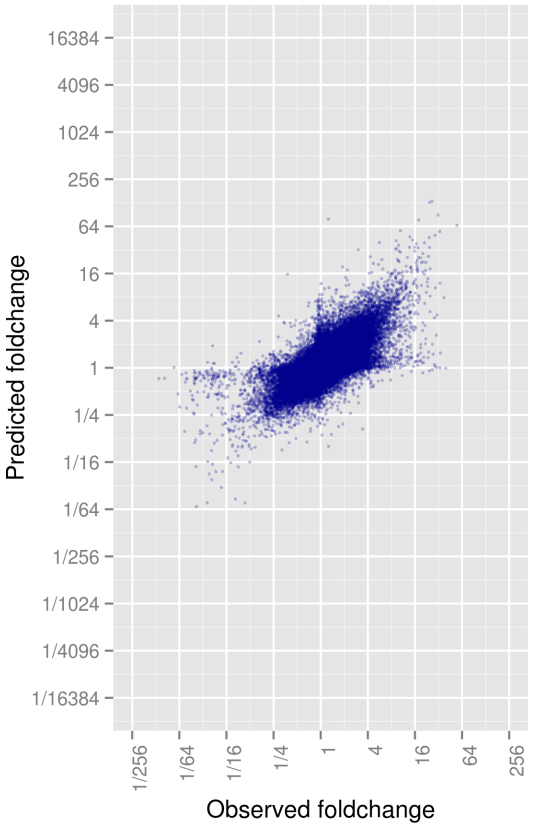


(a) Trained on regulator deletion strains, tested on non-regulator deletion strains

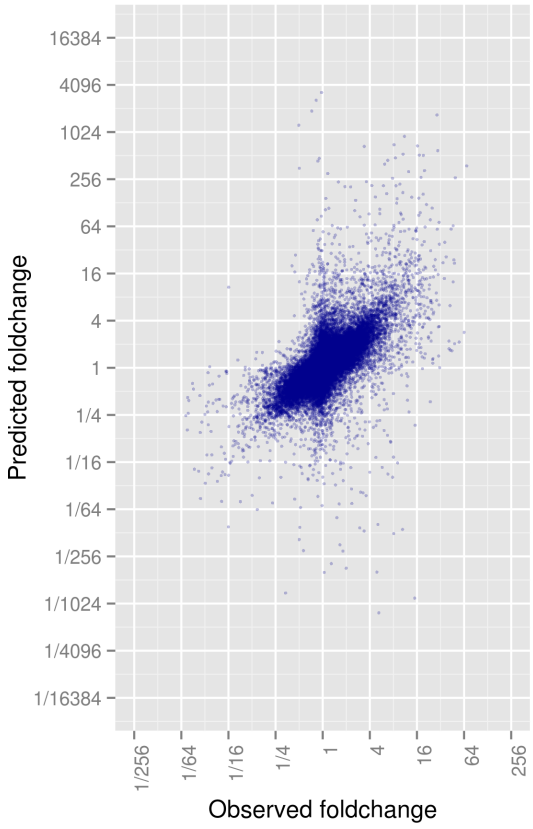


(b) Trained on non-regulator deletion strains, tested on regulator deletion strains

Figure C.6.: Prediction of *S. cerevisiae* target levels from regulator levels by the exponential model

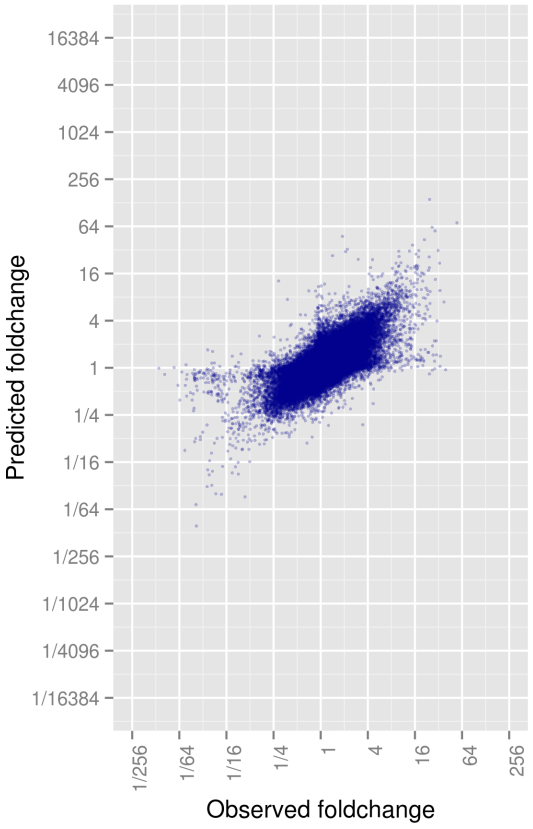


(a) Trained on regulator deletion strains, tested on non-regulator deletion strains

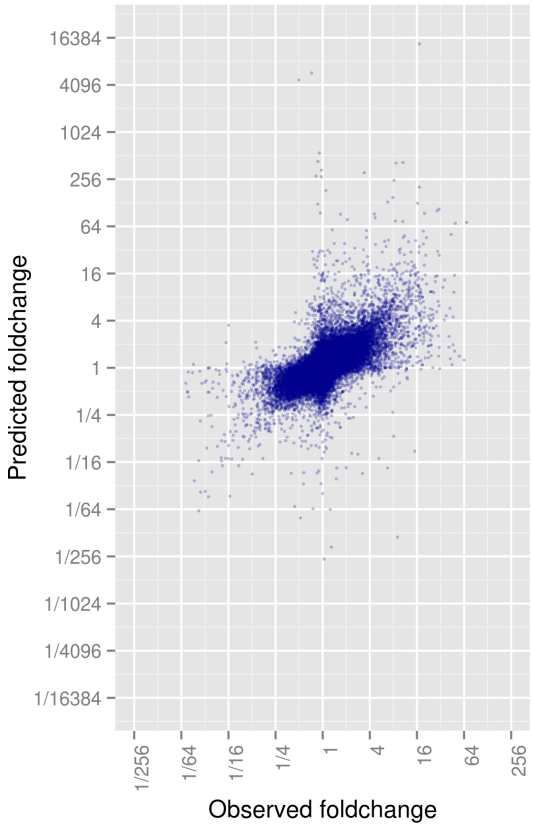


(b) Trained on non-regulator deletion strains, tested on regulator deletion strains

Figure C.7.: Prediction of *S. cerevisiae* target levels from regulator levels by the log-linear-exponential model

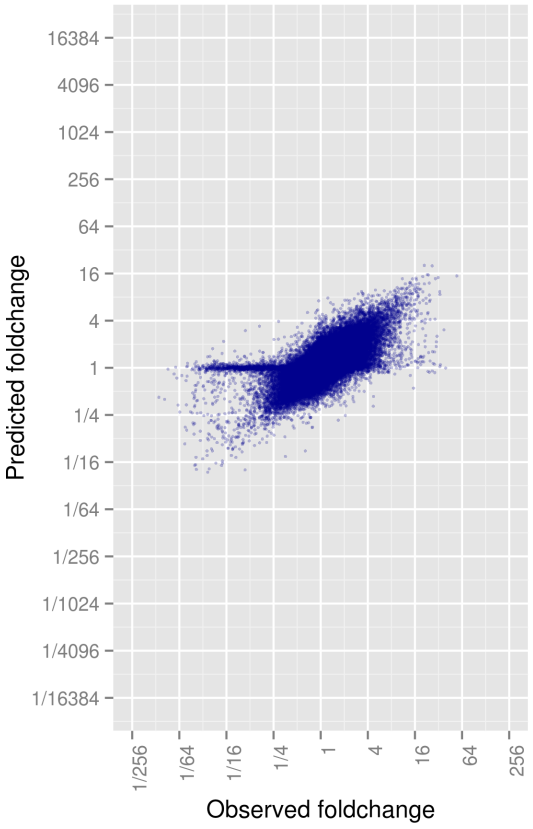


(a) Trained on regulator deletion strains, tested on non-regulator deletion strains

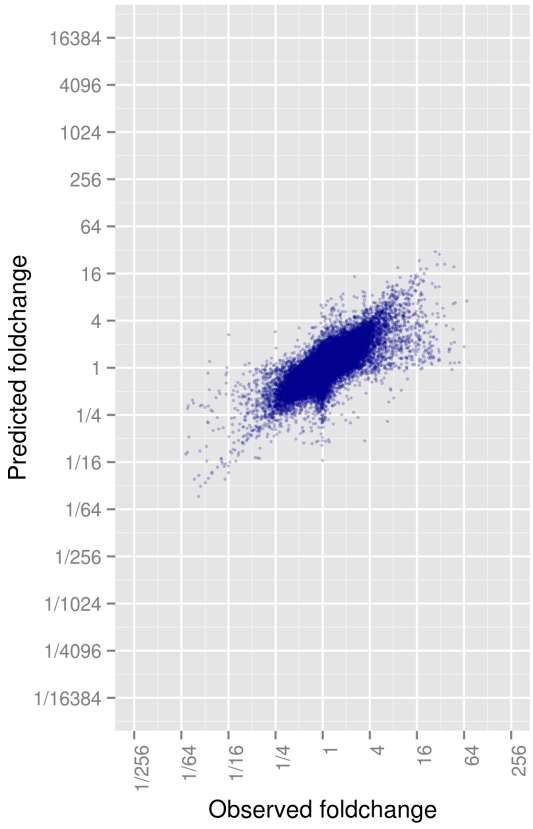


(b) Trained on non-regulator deletion strains, tested on regulator deletion strains

Figure C.8.: Prediction of *S. cerevisiae* target levels from regulator levels by the BART-based semi-parametric model



(a) Trained on regulator deletion strains, tested on non-regulator deletion strains



(b) Trained on non-regulator deletion strains, tested on regulator deletion strains

D. List of *Cryptoccous neoformans* Strains Used in the Study

Wild type

Strain
Serotype A reference strain H99

Single overexpression strain

Strain	CNAG ID of overexpressed gene
*USV101*Over	CNAG_05420

Double deletion strains

Strain	CNAG ID of deleted genes
*pkr1*Δ*rim101*Δ	CNAG_00570, CNAG_05431
*tup1*Δ*rim101*Δ	CNAG_02153, CNAG_05431
*nrg1*Δ*usv101*Δ	CNAG_05222, CNAG_05420
*gat201*Δ*usv101*Δ	CNAG_01551, CNAG_05420
*tup1*Δ*cac1*Δ	CNAG_02153, CNAG_03202
*hog1*Δ*cac1*Δ	CNAG_01523, CNAG_03202
*gat201*Δ*tup1*Δ	CNAG_01551, CNAG_02153
*pkr1*Δ*gat201*Δ	CNAG_00570, CNAG_01551
*ada2*Δ*usv101*Δ	CNAG_01626, CNAG_05420
*ada2*Δ*tup1*Δ	CNAG_01626, CNAG_02153

Single deletion strains

Strain	CNAG ID of deleted gene
*mlr1*Δ	CNAG_00031
*ssn801*Δ	CNAG_00440
*pkr1*Δ	CNAG_00570
*ccd3*Δ	CNAG_00732
*clr3*Δ	CNAG_00871
*ecm2201*Δ	CNAG_00883
*swi6*Δ	CNAG_01438
*hog1*Δ	CNAG_01523
*gat201*Δ	CNAG_01551
*ada2*Δ	CNAG_01626
*tup1*Δ	CNAG_02153
*hap3*Δ	CNAG_02215
*fkh2*Δ	CNAG_02566
*mal13*Δ	CNAG_02774
*asg101*Δ	CNAG_03018
*cac1*Δ	CNAG_03202
*ccd4*Δ	CNAG_03279
*clr2*Δ	CNAG_03378
*asg1*Δ	CNAG_03849
*pdr802*Δ	CNAG_03894
*rds2*Δ	CNAG_03902
*yrm103*Δ	CNAG_04093
*aro8001*Δ	CNAG_04345
*clr1*Δ	CNAG_04353
*none*Δ	CNAG_04369
*cir1*Δ	CNAG_04864
*clr4*Δ	CNAG_04908
*clr5*Δ	CNAG_05067
*nrg1*Δ	CNAG_05222
*usv101*Δ	CNAG_05420
*rim101*Δ	CNAG_05431
*fkh101*Δ	CNAG_05861
*ccd6*Δ	CNAG_06252
*cep3*Δ	CNAG_06276
*bik1*Δ	CNAG_06352
*hap2*Δ	CNAG_07435
*mbs1*Δ	CNAG_07464
*fap1*Δ	CNAG_07506
*hap5*Δ	CNAG_07680
*clr6*Δ	CNAG_07797
*mcm1*Δ	CNAG_07924

E. List of *Cryptococcus neoformans* Genes Considered as Regulators in the Study

224 genes of *Cryptococcus neoformans* are considered as regulators. Their CNAG ID numbers, names, and orthologs in *Saccharomyces cerevisiae* are given here.

CNAG_ID	Name	*S. c.* ortholog	
CNAG_00017		YCR065W	*HCM1*
CNAG_00018		YPL248C	*GAL4*
CNAG_00031	*MLR1*	YLR014C	*PPR1*
CNAG_00039		YNL027W	*CRZ1*
CNAG_00055		YOR028C	*CIN5*
CNAG_00068		YKL062W	*MSN4*
CNAG_00132		YNL167C	*SKO1*
CNAG_00156	*SP1*	YNL027W	*CRZ1*
CNAG_00184		YDR228C	*PCF11*
CNAG_00193	*GAT1*	YFL021W	*GAT1*
CNAG_00239		YML007W	*YAP1*
CNAG_00332		YGR142W	*BTN2*
CNAG_00376		YCR042C	*TAF2*
CNAG_00440	*SSN801*	YNL025C	*SSN8*
CNAG_00460	*LIV1*	YOR032C	*HMS1*
CNAG_00505		YLR098C	*CHA4*
CNAG_00514		YER040W	*GLN3*
CNAG_00559		YNL167C	*SKO1*
CNAG_00570	*PKR1*	YIL033C	*BCY1*
CNAG_00670		YBR297W	*MAL33*
CNAG_00732	*CCD3*	YML076C	*WAR1*
CNAG_00791		YJR060W	*CBF1*
CNAG_00828	*SIP401*	YLR098C	*CHA4*
CNAG_00830		YIL130W	*ASG1*
CNAG_00841		YML076C	*WAR1*
CNAG_00871	*CLR3*	YFL031W	*HAC1*
CNAG_00883	*ECM2201*	YLR228C	*ECM22*
CNAG_00896		YBL005W	*PDR3*
CNAG_00998		YLR313C	*SPH1*
CNAG_01014		YDR146C	*SWI5*
CNAG_01018		YER025W	*GCD11*
CNAG_01069		YIL130W	*ASG1*
CNAG_01173	*PAN1*	YDR333C	
CNAG_01242	*HAPX*	YDR259C	*YAP6*
CNAG_01317		YJR060W	*CBF1*
CNAG_01370		YFL036W	*RPO41*
CNAG_01431		YDL106C	*PHO2*
CNAG_01438	*SWI6*	YLR182W	*SWI6*
CNAG_01454	*STE12a*	YHR084W	*STE12*
CNAG_01523	*HOG1*	YLR113W	*HOG1*
CNAG_01549		YDL030W	*PRP9*
CNAG_01551	*GAT201*	YMR136W	*GAT2*
CNAG_01626	*ADA2*	YDR448W	*ADA2*
CNAG_01645		YJR151C	*DAN4*
CNAG_01708		YMR136W	*GAT2*
CNAG_01841	*GLN3*	YER040W	*GLN3*
CNAG_01858		YDR451C	*YHP1*
CNAG_01883		YJL110C	*GZF3*
CNAG_01948		YPL248C	*GAL4*
CNAG_01952		YPL088W	
CNAG_01973		YPL230W	*USV1*
CNAG_01977		YML076C	*WAR1*
CNAG_01999		YGR171C	*MSM1*
CNAG_02066		YLR451W	*LEU3*
CNAG_02134	*RSC8*	YFR037C	*RSC8*
CNAG_02153	*TUP1*	YCR084C	*TUP1*
CNAG_02215	*HAP3*	YBL021C	*HAP3*
CNAG_02241		YMR087W	
CNAG_02305		YDR421W	*ARO80*
CNAG_02322		YDL170W	*UGA3*
CNAG_02364		YLR266C	*PDR8*
CNAG_02435	*CWC2*	YMR136W	*GAT2*
CNAG_02476	*YRM101*	YOR172W	*YRM1*
CNAG_02516		YOR032C	*HMS1*
CNAG_02555	*SIP402*	YJL089W	*SIP4*
CNAG_02566	*FKH2*	YNL068C	*FKH2*
CNAG_02589		YDL058W	*USO1*
CNAG_02603		YDR216W	*ADR1*
CNAG_02671		YMR213W	*CEF1*
CNAG_02698		YBL047C	*EDE1*
CNAG_02700		YNL027W	*CRZ1*
CNAG_02723		YBR150C	*TBS1*
CNAG_02774	*MAL13*	YKL038W	*RGT1*
CNAG_02788		YDR423C	*CAD1*
CNAG_02877		YPR196W	
CNAG_02936		YCR093W	*CDC39*
CNAG_03018	*ASG101*	YIL130W	*ASG1*
CNAG_03055		YLR074C	*BUD20*
CNAG_03059		YJL206C	
CNAG_03086		YLR098C	*CHA4*
CNAG_03115		YDR213W	*UPC2*
CNAG_03116	*HCM1*	YCR065W	*HCM1*
CNAG_03125		YMR029C	*FAR8*
CNAG_03129		YPR186C	*PZF1*
CNAG_03132		YKL038W	*RGT1*
CNAG_03183		YIL130W	*ASG1*
CNAG_03202	*CAC1*	YJL005W	*CYR1*
CNAG_03212		YCR065W	*HCM1*
CNAG_03229	*YOX101*	YML027W	*YOX1*
CNAG_03261		YJL056C	*ZAP1*
CNAG_03279	*CCD4*	YDR213W	*UPC2*
CNAG_03336		YHR178W	*STB5*
CNAG_03346		YNL167C	*SKO1*
CNAG_03366	*ZNF2*	YNL027W	*CRZ1*
CNAG_03378	*CLR2*	YLR399C	*BDF1*
CNAG_03401		YMR136W	*GAT2*
CNAG_03409	*SKN7*	YHR206W	*SKN7*
CNAG_03431		YJL206C	
CNAG_03527		YDR266C	
CNAG_03561		YBL066C	*SEF1*
CNAG_03710		YLR228C	*ECM22*
CNAG_03741		YKL015W	*PUT3*
CNAG_03768		YLR228C	*ECM22*
CNAG_03788		YLR403W	*SFP1*
CNAG_03790		YBL066C	*SEF1*
CNAG_03817		YCR087C-A	
CNAG_03826	*ESA1*	YOR244W	*ESA1*
CNAG_03849	*ASG1*	YIL130W	*ASG1*
CNAG_03894	*PDR802*	YLR256W	*HAP1*
CNAG_03902	*RDS2*	YPL133C	*RDS2*
CNAG_03904		YDR394W	*RPT3*
CNAG_03914		YGL059W	*PKP2*
CNAG_03976		YNL167C	*SKO1*
CNAG_03998	*RLM1*	YPL089C	*RLM1*
CNAG_04012		YBR150C	*TBS1*
CNAG_04023		YOR162C	*YRR1*
CNAG_04036		YGL073W	*HSF1*
CNAG_04090	*ATF1*	YNL167C	*SKO1*

CNAG_ID	Name	*S. c.* ortholog	
CNAG_04093	*YRM103*	YLR014C	*PPR1*
CNAG_04130		YPL248C	*GAL4*
CNAG_04176		YGL073W	*HSF1*
CNAG_04184		YHR178W	*STB5*
CNAG_04226		YOR308C	*SNU66*
CNAG_04262		YDR421W	*ARO80*
CNAG_04263		YFL021W	*GAT1*
CNAG_04268		YBL019W	*APN2*
CNAG_04345	*ARO8001*	YDR421W	*ARO80*
CNAG_04352	*ZAP103*	YJL056C	*ZAP1*
CNAG_04353	*CLR1*	YJR127C	*RSF2*
CNAG_04369		YMR123W	*PKR1*
CNAG_04398		YDR421W	*ARO80*
CNAG_04457		YDR421W	*ARO80*
CNAG_04518		YDR146C	*SWI5*
CNAG_04530		YJL048C	*UBX6*
CNAG_04583		YGL133W	*ITC1*
CNAG_04586		YDL106C	*PHO2*
CNAG_04588		YBR239C	
CNAG_04594		YDL170W	*UGA3*
CNAG_04600		YLR234W	*TOP3*
CNAG_04630		YIR018W	*YAP5*
CNAG_04637	*MBF1*	YOR298C-A	*MBF1*
CNAG_04774		YKR064W	*OAF3*
CNAG_04790		YJL056C	*ZAP1*
CNAG_04798	*GCN4*	YEL009C	*GCN4*
CNAG_04804	*SRE1*	YOR032C	*HMS1*
CNAG_04807		YMR019W	*STB4*
CNAG_04836		YIL130W	*ASG1*
CNAG_04837	*MLN1*	YBL103C	*RTG3*
CNAG_04841		YIL130W	*ASG1*
CNAG_04855		YKR054C	*DYN1*
CNAG_04864	*CIR1*	YJL110C	*GZF3*
CNAG_04878		YKL222C	
CNAG_04895		YIL130W	*ASG1*
CNAG_04908	*CLR4*	YGR089W	*NNF2*
CNAG_04916		YKR064W	*OAF3*
CNAG_05010		YKL062W	*MSN4*
CNAG_05019		YLR450W	*HMG2*
CNAG_05049		YDR034C	*LYS14*
CNAG_05055	*RTS2*	YOR077W	*RTS2*
CNAG_05067	*CLR5*	YLR399C	*BDF1*
CNAG_05093		YPL124W	*SPC29*
CNAG_05112		YOR337W	*TEA1*
CNAG_05153		YER040W	*GLN3*
CNAG_05170		YOR363C	*PIP2*
CNAG_05176		YDL106C	*PHO2*
CNAG_05222	*NRG1*	YDR043C	*NRG1*
CNAG_05255		YJL206C	
CNAG_05311		YFR023W	*PES4*
CNAG_05314	*GLO3*	YER122C	*GLO3*
CNAG_05375		YBL103C	*RTG3*
CNAG_05380		YJL206C	
CNAG_05392	*ZAP104*	YJL056C	*ZAP1*
CNAG_05420	*USV101*	YPL230W	*USV1*
CNAG_05431	*RIM101*	YHL027W	*RIM101*
CNAG_05436		YEL009C	*GCN4*
CNAG_05535	*FHL1*	YPR104C	*FHL1*
CNAG_05538	*JJJ1*	YNL227C	*JJJ1*
CNAG_05642		YJL206C	
CNAG_05785	*STB4*	YMR019W	*STB4*
CNAG_05861	*FKH101*	YIL131C	*FKH1*
CNAG_05940		YPR008W	*HAA1*
CNAG_06097		YLL054C	
CNAG_06134	*BZP1*	YFL031W	*HAC1*
CNAG_06156		YHR056C	*RSC30*
CNAG_06163		YOR326W	*MYO2*
CNAG_06188		YKL038W	*RGT1*
CNAG_06223		YDR409W	*SIZ1*
CNAG_06252	*CCD6*	YDR213W	*UPC2*
CNAG_06276	*CEP3*	YMR168C	*CEP3*
CNAG_06283	*LIV4*	YDR026C	
CNAG_06322	*SAS3*	YBL052C	*SAS3*
CNAG_06327	*MIG1*	YGL035C	*MIG1*
CNAG_06339		YCR106W	*RDS1*
CNAG_06352	*BIK1*	YCL029C	*BIK1*
CNAG_06425		YLR014C	*PPR1*
CNAG_06483		YEL071W	*DLD3*
CNAG_06719		YHR178W	*STB5*
CNAG_06742		YLR387C	*REH1*
CNAG_06751		YFR034C	*PHO4*
CNAG_06762	*GAT204*	YMR136W	*GAT2*
CNAG_06814	*SXI1α*	YPL177C	*CUP9*
CNAG_06818	*HAP1*	YLR256W	*HAP1*
CNAG_06826		YDR485C	*VPS72*
CNAG_06871		YJL206C	
CNAG_06921		YKR002W	*PAP1*
CNAG_07011		YLR014C	*PPR1*
CNAG_07329		YLR403W	*SFP1*
CNAG_07370		YER045C	*ACA1*
CNAG_07411	*RUM1*	YJR119C	*JHD2*

CNAG_ID	Name	*S. c.* ortholog	
CNAG_07435	*HAP2*	YGL237C	*HAP2*
CNAG_07443		YJR060W	*CBF1*
CNAG_07460		YGL073W	*HSF1*
CNAG_07464	*MBS1*	YDL056W	*MBP1*
CNAG_07506	*FAP1*	YNL023C	*FAP1*
CNAG_07528		YOR317W	*FAA1*
CNAG_07560		YIL036W	*CST6*
CNAG_07607		YIR004W	*DJP1*
CNAG_07680	*HAP5*	YOR358W	*HAP5*
CNAG_07724	*CUF1*	YPR008W	*HAA1*
CNAG_07797	*CLR6*	YKL070W	
CNAG_07901		YLR256W	*HAP1*
CNAG_07922		YHR178W	*STB5*
CNAG_07924	*MCM1*	YMR043W	*MCM1*
CNAG_07940		YIL036W	*CST6*

F. List of *Saccharomyces cerevisiae* Genes Considered as Regulators in the Study

OAF1 (YAL051W), *PDR3* (YBL005W), *HIR1* (YBL008W), *HAP3* (YBL021C), *SAS3* (YBL052C), *TOD6* (YBL054W), *SEF1* (YBL066C), *RTG3* (YBL103C), *EDS1* (YBR033W), *REB1* (YBR049C), *NRG2* (YBR066C), *TEC1* (YBR083W), *NHP6B* (YBR089C-A), *SIF2* (YBR103W), *CYC8* (YBR112C), *TBS1* (YBR150C), *SMP1* (YBR182C), *MSI1* (YBR195C), *ERT1* (YBR239C), *THI2* (YBR240C), *ISW1* (YBR245C), *RIF1* (YBR275C), *SNF5* (YBR289W), *MAL33* (YBR297W), *KAR4* (YCL055W), *HMLα1*, *HMLα2*, *SRD1* (YCR018C), *MATα2*, *MATα1*, *HCM1* (YCR065W), *SRB8* (YCR081W), *TUP1* (YCR084C), *HMRA2* (YCR096C), *HMRA1* (YCR097W), *RDS1* (YCR106W), *NHP10* (YDL002C), *RPN4* (YDL020C), *SIR2* (YDL042C), *SIT4* (YDL047W), *STP4* (YDL048C), *MBP1* (YDL056W), *BDF2* (YDL070W), *PHO2* (YDL106C), *UGA3* (YDL170W), *GAL3* (YDR009W), YDR026C, *LYS14* (YDR034C), *NRG1* (YDR043C), *VMS1* (YDR049W), *SNF11* (YDR073W), *PDC2* (YDR081C), *GIS1* (YDR096W), *INO2* (YDR123C), *SWI5* (YDR146C), *STB3* (YDR169C), *ARG82* (YDR173C), *HMO1* (YDR174W), *NGG1* (YDR176W), *SAS4* (YDR181C), *HST4* (YDR191W), *UME6* (YDR207C), *UPC2* (YDR213W), *ADR1* (YDR216W), *MET32* (YDR253C), *YAP6* (YDR259C), YDR266C, *MTH1* (YDR277C), *RSC3* (YDR303C), *SUM1* (YDR310C), *ESC2* (YDR363W), *SPT3* (YDR392W), *ARO80* (YDR421W), *CAD1* (YDR423C), *SSN2* (YDR443C), *ADA2* (YDR448W), *YHP1* (YDR451C), *STP1* (YDR463W), *SNF1* (YDR477W), *DIG2* (YDR480W), *PLM2* (YDR501W), *URC2* (YDR520C), *GCN4* (YEL009C), *HAT2* (YEL056W), *MIG3* (YER028C), *GLN3* (YER040W), *ACA1* (YER045C), *JHD1* (YER051W), *MOT2* (YER068W), *DOT6* (YER088C), *FLO8* (YER109C), *SWI4* (YER111C), YER130C, *SPT15* (YER148W), *SPT2* (YER161C), *RPH1* (YER169W), YER184C, *GAT1* (YFL021W), *HAC1* (YFL031W), *OTU1* (YFL044C), YFL052W, *CDC14* (YFR028C), *PHO4* (YFR034C), *PDR1* (YGL013C), *PIB2* (YGL023C), *PGD1* (YGL025C), *MIG1* (YGL035C), *AFT1* (YGL071W), *HSF1* (YGL073W), *TOS8* (YGL096W), *SNT2* (YGL131C), *NUT1* (YGL151W), *SUT1* (YGL162W), *CUP2* (YGL166W), *GTS1* (YGL181W), *MDS3* (YGL197W), *MIG2* (YGL209W), *HAP2* (YGL237C), *RTF1* (YGL244W), *RTG2* (YGL252C), *FZF1* (YGL254W), *KSS1* (YGR040W), *RME1* (YGR044C), *RSC1* (YGR056W), *SPT4* (YGR063C), YGR067C, *NNF2* (YGR089W), *ASK10* (YGR097W), *SRB5* (YGR104C), *MGA1* (YGR249W), *MAL13* (YGR288W), *YAP3* (YHL009C), *OPI1* (YHL020C), *SNF6* (YHL025W), *RIM101* (YHL027W), *STP2* (YHR006W), *SRB2* (YHR041C), *RSC30* (YHR056C), *STE12* (YHR084W), *NDT80* (YHR124W), *RTT107* (YHR154W), *STB5* (YHR178W), *RPN10* (YHR200W), *SKN7* (YHR206W), *DOT5* (YIL010W), *CST6* (YIL036W), *NOT3* (YIL038C), *SDS3* (YIL084C), *XBP1* (YIL101C), *RPI1* (YIL119C), *MET18* (YIL128W), *ASG1* (YIL130W), *FKH1* (YIL131C), *IMP2'* (YIL154C), *GAT4* (YIR013C), *MET28* (YIR017C), *YAP5* (YIR018W), *DAL81* (YIR023W), *MGA2* (YIR033W), *ZAP1* (YJL056C), *SIP4* (YJL089W), *GSM1* (YJL103C), *GZF3* (YJL110C), *ASF1* (YJL115W), *SPT10* (YJL127C), *SET2* (YJL168C), *SWI3* (YJL176C), YJL206C, *CBF1* (YJR060W), *IME1* (YJR094C), *IBA57* (YJR122W), *RSF2* (YJR127C), *HIR3* (YJR140C), *HMS2* (YJR147W), *BYE1* (YKL005C), *PUT3* (YKL015W), *SPT23* (YKL020C), *IXR1* (YKL032C), *RGT1* (YKL038W), *PHD1* (YKL043W), *MSN4* (YKL062W), *STB6* (YKL072W), *HAP4* (YKL109W), *ABF1* (YKL112W), *ASH1* (YKL185W), YKL222C, *DAL80* (YKR034W), *CAF4* (YKR036C), *OAF3* (YKR064W), *BAS1* (YKR099W), *SIR1* (YKR101W), YLL054C, *GAT3* (YLR013W), *PPR1* (YLR014C), *RIC1* (YLR039C), *CHA4* (YLR098C), *HOG1* (YLR113W), *ACE2* (YLR131C), *TIS11* (YLR136C), *RFX1* (YLR176C), *SWI6* (YLR182W), *TOS4* (YLR183C), *IFH1* (YLR223C), *ECM22* (YLR228C), *HAP1* (YLR256W), *PDR8* (YLR266C), YLR278C, *RSC2* (YLR357W), *STP3* (YLR375W), *SFP1* (YLR403W), *CDC73* (YLR418C), *SIR3* (YLR442C), *LEU3* (YLR451W), *RIF2* (YLR453C), *YAP1* (YML007W), *YOX1* (YML027W), *GAL80* (YML051W), *WAR1* (YML076C), *TDA9* (YML081W), *ARG81* (YML099C), *CAC2* (YML102W), *DAT1* (YML113W), *SOK2* (YMR016C), *STB4* (YMR019W), *MAC1* (YMR021C), *MSN2* (YMR037C), *ARG80* (YMR042W), *MCM1* (YMR043W), *STB2* (YMR053C), *MOT3* (YMR070W), *ABF2* (YMR072W), *RCO1* (YMR075W), *MSS11* (YMR164C), *CEP3* (YMR168C), *HOT1* (YMR172W), *RGM1* (YMR182C), *ZDS1* (YMR273C), *CAT8* (YMR280C), *ELP6* (YMR312W), *HDA1* (YNL021W), *CRZ1* (YNL027W), *FKH2* (YNL068C), *PHO23* (YNL097C), *MET4* (YNL103W), *THO2* (YNL139C), *SKO1* (YNL167C), *GCR2* (YNL199C), *SPS18* (YNL204C), *RAP1* (YNL216W), *SIN4* (YNL236W), *GIS2* (YNL255C), *SIP3* (YNL257C), *STB1* (YNL309W), *DAL82* (YNL314W), *RPD3* (YNL330C), *CSE2* (YNR010W), *POP2* (YNR052C), YNR063W, *SIN3* (YOL004W), *YAP7* (YOL028C), *GAL11* (YOL051W), *RTG1* (YOL067C), *HST1* (YOL068C), *HAL9* (YOL089C), *INO4* (YOL108C), *MSN1* (YOL116W), *SPT20* (YOL148C), *HST3* (YOR025W), *CIN5* (YOR028C), *HMS1* (YOR032C), *HIR2* (YOR038C), *RTS2* (YOR077W), *AZF1* (YOR113W), *SFL1* (YOR140W), *YRR1* (YOR162C), *YRM1* (YOR172W), *ULS1* (YOR191W), *SAS5* (YOR213C), *WTM2* (YOR229W), *WTM1* (YOR230W), *SNF2* (YOR290C), *MBF1* (YOR298C-A), *ISW2* (YOR304W), *TEA1* (YOR337W), *TYE7* (YOR344C), *HAP5* (YOR358W), *PIP2* (YOR363C), *NDD1* (YOR372C), *RDR1* (YOR380W), *HAT1* (YPL001W), *ECM23* (YPL021W), *MET31* (YPL038W), *SSN3* (YPL042C), *DIG1* (YPL049C), *GCR1* (YPL075W), *RLM1* (YPL089C), *TBF1* (YPL128C), *TAF14* (YPL129W), *RDS2* (YPL133C), *UME1* (YPL139C), *CUP9* (YPL177C), *AFT2* (YPL202C), *USV1* (YPL230W), *GAL4* (YPL248C), *HFI1* (YPL254W), *MDL2* (YPL270W), *HAA1* (YPR008W), *SUT2* (YPR009W), YPR013C, YPR015C, *RLF2* (YPR018W), YPR022C, *NHP6A* (YPR052C), *SMK1* (YPR054W), *ROX1* (YPR065W), *RDS3* (YPR094W), *FHL1* (YPR104C), *HPA2* (YPR193C), YPR196W, *ARR1* (YPR199C)

REFERENCES

[1] O'Duibhir E, Lijnzaad P, Benschop JJ, Lenstra TL, van Leenen D, Groot Koerkamp MJ, Margaritis T, Brok MO, Kemmeren P, Holstege FC. Cell cycle population effects in perturbation studies. *Molecular Systems Biology*. 2014; 10(6): 732. doi: 10.15252/msb.20145172.

[2] Kemmeren P, Sameith K, van de Pasch LA, Benschop JJ, Lenstra TL, Margaritis T, O'Duibhir E, Apweiler E, van Wageningen S, Ko CW, van Heesch S, Kashani MM, Ampatziadis-Michailidis G, Brok MO, Brabers NA, Miles AJ, Bouwmeester D, van Hooff SR, van Bakel H, Sluiters E, Bakker LV, Snel B, Lijnzaad P, van Leenen D, Groot Koerkamp MJ, Holstege FC. Large-Scale Genetic Perturbations Reveal Regulatory Networks and an Abundance of Gene-Specific Repressors. *Cell*. 2014; 157(3): 740-752.

[3] Law CW, Chen Y, Shi W, Smyth GK. voom: Precision weights unlock linear model analysis tools for RNA-seq read counts. *Genome Biology*. 2014; 15(2): R29. doi: 10.1186/gb-2014-15-2-r29.

[4] Haynes BC, Maier EJ, Kramer MH, Wang PI, Brown H, Brent MR. Mapping functional transcription factor networks from gene expression data. *Genome research*.

2013; 23: 1319-1328. doi: 10.1101/gr.150904.112.

[5] Marbach D, Roy S, Ay F, Meyer PE, Candeias R, Kahveci T, Bristow CA, Kellis M. Predictive regulatory models in Drosophila melanogaster by integrative inference of transcriptional networks. *Genome research*. 2012; 22(7): 1334-1349. doi: 10.1101/gr.127191.111.

[6] McCarthy DJ, Chen Y, Smyth GK. Differential expression analysis of multifactor RNA-Seq experiments with respect to biological variation. *Nucleic Acids Research*. 2012; 40: 4288-4297.

[7] Hahn S, Young ET. Transcriptional Regulation in Saccharomyces cerevisiae: Transcription Factor Regulation and Function, Mechanisms of Initiation, and Roles of Activators and Coactivators. *Genetics*. 2011; 189(3): 705-736. doi: 10.1534/genetics.111.127019.

[8] Kaplan T, Li XY, Sabo PJ, Thomas S, Stamatoyannopoulos JA, Biggin MD, Eisen MB. Quantitative Models of the Mechanisms That Control Genome-Wide Patterns of Transcription Factor Binding during Early Drosophila Development. *PLoS Genetics*. 2011; 7(2): e1001290. doi: 10.1371/journal.pgen.1001290.

[9] Chipman HA, George EI, McCulloch RE. BART: Bayesian Additive Regression Trees. *Annals of Applied Statistics*. 2010; 4(1): 266-298.

[10] Ruan J. A top-performing algorithm for the DREAM3 gene expression prediction challenge. *PLoS ONE*. 2010; 5(2): e8944. doi:10.1371/journal.pone.0008944.

[11] Segal E, Raveh-Sadka T, Schroeder M, Unnerstall U, Gaul U. Predicting expression patterns from regulatory sequence in Drosophila segmentation. *Nature*. 2008; 451(7178): 535-540. doi: 10.1038/nature06496.

[12] Murphy KP. *Conjugate Bayesian analysis of the Gaussian distribution*. 2007. http://www.cs.ubc.ca/ murphyk/Papers/bayesGauss.pdf. Accessed February 1, 2015.

[13] Carter GW, Prinz S, Neou C, Shelby JP, Marzolf B, Thorsson V, Galitski T. Prediction of phenotype and gene expression for combinations of mutations. *Molecular Systems Biology*. 2007; 3: 96. doi: 10.1038/msb4100137.

[14] Hu Z, Killion PJ, Iyer VR. Genetic reconstruction of a functional transcriptional regulatory network. *Nature Genetics*. 2007; 39(5): 683-687.

[15] Bonneau R, Reiss DJ, Shannon P, Facciotti M, Hood L, Baliga NS, Thorsson V. The Inferelator: an algorithm for learning parsimonious regulatory networks from systems-biology data sets de novo. *Genome biology*. 2006; 7(5): R36.

[16] Smyth GK. Limma: Linear Models for Microarray Data. *Statistics for Biology and Health*. 2005; *Bioinformatics and Computational Biology Solutions Using R and Bioconductor*: 397-420.

[17] Stormo GD. DNA binding sites: representation and discovery. *Bioinformatics*. 2000; 16(1) : 16-23. doi:10.1093/bioinformatics/16.1.16. PMID 10812473.

[18] Tibshirani R. *Regression shrinkage and selection via the lasso. Journal for the Royal Statistical Society, Series B.* 1996; 58(1): 267-288.

[19] Juan LJ, Walter PP, Taylor IC, Kingston RE, Workman JL. Nucleosome cores and histone H1 in the binding of GAL4 derivatives and the reactivation of transcription from nucleosome templates in vitro. *Cold Spring Harbor Symposia on Quantitative Biology.* 1993; 58: 213-223.

[20] Shapiro SS, Wilk MB. An analysis of variance test for normality (complete samples). *Biometrika.* 1965; 52(3-4): 591-611. doi:10.1093/biomet/52.3-4.591.

[21] Lilliefors H. On the Kolmogorov–Smirnov test for normality with mean and variance unknown. *Journal of the American Statistical Association.* 1967; 62(318): 399-402.