DREAM4: Combining Genetic and Dynamic Information to Identify Biological Networks and Dynamical Models

Alex Greenfield[1±], Aviv Madar[2±], Harry Ostrer[3] Richard Bonneau[1,2,4*]

1 Computational Biology Program, New York University Sackler School of Medicine, New York, NY, USA
2 Department of Biology, Center for Genomics & Systems Biology, New York University, New York, NY, USA
3 Human Genetics Program, Department of Pediatrics, New York University Langone Medical Center, New York, New York 10016, USA
4 Computer Science Department, Courant institute of Mathematical Sciences, New York University, New York, NY, USA
∗ E-mail: Bonneau@nyu.edu
± both authors contributed equally to this work

Abstract

Background Current technologies have lead to the availability of multiple genomics data types in sufficient quantity, and of sufficient quality, to serve as a basis for automatic global network inference. Accordingly, there is currently a large variety of network inference algorithms that learn regulatory networks to varying degrees of detail. Methods that use steady-state data of the system's response to genetic perturbations (or other interventions) can accurately learn the topology of the network, while methods that use time-series data, that captures the temporal response of the system to new stimuli, can learn dynamical models that can be used to predict the response of the system to new conditions.

Methodology We investigate methods that combine time-series data with steady-state genetic perturbation data to improve the overall accuracy of regulatory network prediction and compare these methods to simpler component methods. We test a two-stage method that includes: 1) a simple methods that uses only steady-state measurements following genetic perturbations (such as gene knock outs) and 2) a method that includes an explicit treatment of regulatory dynamics (time-series following perturbations). Our combined method is able to infer directed causal regulatory interactions and predict the response of the system to new conditions (in this case new double knock-out genetic perturbations).

Conclusion/significance Our method based solely on steady-state data describing the response of the system to genetic knockouts tied for 1st out of 19 teams in the DREAM4 100-gene *in-silico* network challenge. We demonstrate complementarity between this method and a method that utilizes time-series data. Combining data from these fundamentally different experimental designs resulted in improved predictive performance of the topology of the network, while retaining the ability to predict the response of the system to new conditions. Our evaluation of the performance of multiple methods for network inference suggests avenues for future methods developments and provides simple considerations for genomic experimental design.

Introduction

Predicting how a cell will respond, at the molecular level, to environmental and genetic perturbations is a key problem in systems biology. Molecular regulatory systems-level responses are governed by several regulatory mechanisms including the underlying transcriptional regulatory network (RN). Recently, there has been an increase in the number of genome-wide datasets appropriate for large scale network inference, which has driven a large interest in methods for learning regulatory networks from these datasets. In general, the question of inferring a transcriptional RN can be posed in the following way: given a set of regulators (transcription factors - TFs) and a set of targets (genes), what are the regulatory relationships between the elements in these two sets? These relationships can be directed (e.g. gene A regulates gene B) or undirected (e.g. there is a regulatory relationship between gene A and gene B). RN inference techniques use three main types of genome-wide data: 1) steady-state profiling of gene knockout, knockdown and over-expression, 2) collections of time series observations following relevant perturbations, and 3) measurements of TF-DNA binding. Different types of RN inference methods produce RNs that vary in detail and comprehension. One critical distinction is the scalability of any given method. Typically, methods that learn less detailed regulatory models scale to larger systems and data sizes than methods that learn more complex models. Another critical difference between methods is whether causal (directed) edges or undirected relationships are learned. Several current methods aim to learn dynamical parameters, such as TF-target activation rates and rates of degradation of gene products. Ideally, a computational biologist should choose the most detailed method that the data will support, as more detailed models can suggest more focused biological hypothesis and be used to model a system's behavior in ways that simple network models cannot. Given this constant need to balance the specific features of any given biological dataset with the capabilities of multiple RN inference algorithms, testing of RN inference methods using a variety of datasets is a critical field-wide activity. This is evidenced by several recent methods that aim to generate biologically meaningful datasets with a known underlying topology [1–4].

To this end, the Dialogue for Reverse Engineering Assessments and Methods (DREAM) [5] provides a set of networks which can be used to develop and test RN inference methods. The networks presented by DREAM make some simplifications of the networks found in a cell, and the corresponding datasets are ideal in their completeness. The control of cellular processes occurs on at least three distinct levels including transcript, protein, and metabolite. Measuring only transcript levels ignores the fact that cellular interactions happen on the level of proteins, and are mediated in many cases by metabolites. Accordingly, an ideal dataset for RN inference would contain time-series measurements of multiple levels of regulation (RNA, protein, protein-modifications, etc.) with the sampling rate on the order of the fastest reaction. Additionally, the cellular response to genetic perturbation (gene knockout or knockdown) would also be available. Although advances are currently being made in the cost and accuracy of genome-wide proteomics, metabolomics, and protein binding (ChIP-chip, ChIP-seq) [6, 7] measurements, the most mature and cost efficient technologies remain those that measure genome-wide transcription-level responses. Experimental and financial constraints typically prohibit obtaining these measurements in a finely time-resolved manner. The DREAM challenge removes many of these constraints and presents participants with an idealized expression dataset for which the true topology (gold-standard) is known. This presents a unique opportunity to develop RN inference methods and immediately test their performance by comparison with the gold-standard. It should be noted that biological systems present several advantages not relevant to the DREAM4 challenge. These advantages (not discussed here) are leveraged by integrative methods for learning modularity prior to inference [8–12], methods that use structured priors derived from compilations of validated biological regulatory interactions [13–16], and approaches to characterize binding sites [17, 18]. A thorough review of current network inference methods is beyond the scope of this introduction but can be found in [19–24]. Here we briefly review only the classes of methods that we utilized in our hybrid approach: mutual information based methods, ODE-based methods, and resampling methods.

Several methods for detecting significant regulatory associations are based on similarity metrics derived from information theory, such as Mutual Information. [25]. The Mutual Information (MI) between two signals (in this case the expression of a TF and its target) is calculated by subtracting the joint entropy of each signal from the sum of their entropies. It is similar to correlation (the higher the value, the stronger the relationship), but is more generally applicable as it does not assume a linear relationship between the two signals, nor does it assume continuity. At their core, methods that rely on mutual information generally infer undirected interactions, as the MI between two variables is a symmetric quantity [26–29], however modifications can be made that allow for the inference of direction [30, 31]. Here, we use an MI-based method, Time-Lagged Context Likelihood of Relatedness (TL-CLR) [31], which is based on CLR [29], to learn initial topology that is further optimized and parametrized by Inferelator 1.0 [32]. TL-CLR computes a directed measure that makes use of the temporal information contained in time series observations to estimate the directionality of a significant regulatory interaction. This method is described in [31] and is reviewed in the methods section. TL-CLR cannot be used to predict new data as it does not infer any dynamical parameters. A different approach is needed to calculate dynamical parameters for the regulatory edges in a given topology. In the context of a full regulatory network inference pipeline that includes fitting of dynamical parameters, TL-CLR can be used as a feature selection algorithm that identifies the most likely regulators for each target based on time-lagged, corrected MI.

Ordinary differential equation (ODE) based methods for RN inference attempt to learn not only the topology of the network, but also the dynamical parameters of each regulatory interaction. Regulatory network models resulting from these methods can be used to predict the system-wide response to previously unseen conditions, future time-points, and the effects of removing system components. A drawback of these methods is that they generally require time-series data and more complete datasets than many alternative methods. ODE methods model the rate of change in the expression of a gene as a function of TFs (and other relevant effects) in the system. ODE based methods differ in the underlying functional forms, how the ODE system of equations is solved (couple or uncoupled solution), and how structure priors and sparsity constraints are imposed on the overall inference procedure. For example, several methods have been proposed that use complex functional forms [33], and solve a coupled system [33, 34], while other methods [32, 35–38] solve a simplified linear system of ODEs. The Inferelator 1.0 [32], is an RN inference method which learns the network as a system of linear ODEs, where the rate of change for each gene is modeled as a function of the known regulators in the system. Inferelator 1.0 uses a finite difference approximation to estimate the change in the response over a given time interval, and uses an efficient implementation of l_1-constrained linear regression, LARS [39], to enforce model sparsity. The Inferelator 1.0 has previously been used to learn a large portion of the *Halobacterium salinarium* transcriptional regulatory network, and was able to predict mRNA levels of 85% of the genes in the genome over new experimental conditions [40]. Additionally, feature selection by TL-CLR followed by optimization and parameterization via Inferelator 1.0 was a top performing method for the DREAM3 network challenge [31]. One drawback of these scalable MI and ODE based methods is that they rely on point estimates for many network parameters and thus are not ideal for estimating the error in the inferred parameters [41]. One possible solution is to use a resampling approach [42, 43] to generate an ensemble of predicted networks from which the confidence interval for any parameter can be estimated.

Resampling refers to a broad class of statistical methods that are often used to assess confidence bounds on sample statistics by empirically generating distributions [42]. Recently, several groups have used resampling approaches in a biological context. In this setting resampling methods are an attractive means of determining confidence bounds on model parameters (such as the strength and directionality of a putative regulatory interaction) for two main reasons: 1) resampling methods are non parametric and thus applicable in cases where complex or ill-understood regulatory relationships might confound assumptions about the correct error distribution, and 2) resampling methods do not, in our case, decrease algorithm scalability. Resampling methods have been applied in several contexts to estimate error in a variety of genomics data-analysis contexts. Kerr et al. [44] used a resampling approach to assess confidence bounds of clusters from ANOVA models. Resampling of a gaussian process regression model was used by Kirk et al. [45] to show the sensitivity of the inferred network to uncertainty in the underlying data. Friedman et al. [46] used a resampling approach of a Bayesian network reconstruction algorithm to assess the confidence of inferred parameters. Additionally, Marbach et al. [47] showed that a resampling approach applied to a genetic algorithm for network inference was a top performering method in the DREAM2 five-gene network challenge. We show that by using resampling to generate ensembles of networks with our network inference pipeline we can improve the accuracy of our topology prediction.

Here, we present our best-performing method for the DREAM4 *in-silico* 100 gene network inference challenge. We show that using the genetic knockout data alone, in a relatively straightforward way, achieved best performance in predicting the topology of the DREAM4 100 gene networks. We also show that our network inference pipeline of TL-CLR $\rightarrow$ Inferelator 1.0 is capable of predicting the system-wide response to combinations of new genetic perturbations (predicting expression levels in the double knock-out challenge). We demonstrate that the integration of heterogeneous data types by a resampling approach can markedly improve our ability to infer network topology while retaining our ability to predict data in the double knockout challenge. We focus on comparing these methods in an effort to determine when ODE-based, MI-based, and genetic-perturbation based methods (or combinations thereof) can be expected to perform best. This comparison provides several simple considerations for determining the correct balance between genetic perturbations and time series data required for large-scale network inference.

Materials and Methods

Here we describe all methods used for the the DREAM4 *in-silico* 100-gene network reconstruction challenge. We first review the networks and data we were given and the metrics used to evaluate performance. We then describe the method we used to generate network-topology predictions using only the genetic knockout observations. Next, we present, in detail, the tlCLR-Inferelator pipeline we used to infer dynamical parameters and predict double knockout data. We conclude by describing how the rankings produced by tlCLR-Inferelator and those produced by a method that uses only the knockout data can be combined to improve the overall topology prediction while retaining the ability to generate double knockout predictions.

Problem Set Up

The DREAM4 *in-silico* network reconstruction challenge consists of five synthetic networks of 100 genes used to generate five corresponding datasets. The five networks vary in their topology, chosen to mimic either *Escherichia coli* or *Saccharomyces cerevisiae*, and their dynamical properties, determined by initial conditions and the kinetic parameters chosen for each of the five networks [1]. Stochastic differential equations, followed by the addition of noise proportional to the level of gene expression (as seen in real mircroarray datasets), were used to generate expression data from each topology. Denote the expression levels of the genes by $x = (x_1, \ldots, x_N)^T$. We are given four sets of observations: time-series (X^{ts}), wild-type (X^{wt}), knockout (X^{ko}), and knockdown (X^{kd}). To generate the time-series data a perturbation was introduced into the system for a period of time, and then removed. Measurements were taken at evenly spaced time intervals as the system responded to the perturbation, and as it relaxed. X^{wt} is composed of the first observation in each time series (of which there are ten), and one provided observation of wild-type expression. To generate the knockout data the transcription rate of each gene was set to zero (in turn), the network was equilibrated, and the steady-state expression for all genes in system was measured. Likewise, for the knockdown data the transcription rate of each gene was set to half of its wild-type rate, the network equilibrated, and the steady-state expression levels of all of the genes in the system were measured. For the main challenge participants were presented with this data, but not the underlying network topology or kinetic parameters, and asked to submit a ranked list of regulatory interactions sorted by confidence (highest-confidence interactions at the top of the list). The topology predictions were evaluated by area under the precision recall curve (AUPR) [5]. A perfect prediction would have all true regulatory interactions (i.e. true positives) ranked higher than false regulatory interactions (i.e. true negatives). In addition to this main challenge, participants also had the option of taking part in a bonus-round challenge aimed at assessing a method's ability to predict system-wide behavior in response to new genetic perturbations, the double knockout challenge. For each network participants were presented with twenty double knockout perturbations, in which the transcription rate of a pair of genes was set to zero simultaneously, and asked to predict the steady-state expression of all other genes in response to the perturbation. The accuracy of the prediction was evaluated by calculating the mean square error (MSE) between the actual and predicted expression of the N genes.

We now present the four methods we used to learn regulatory networks from the DREAM4 data. We developed a method based only on the knockout data, X^{ko}, which we refer to as mean-corrected Z-scores (MCZ). We submitted the predictions made by MCZ, tying for 1st out of 19 teams. We also developed a method that takes advantage of the time-series data to infer topology and dynamical parameters (based on our method for DREAM3 [31]), which we refer to as tlCLR-Inferelator (the TL-CLR $\rightarrow$ Inferelator 1.0 pipeline). We used the dynamical parameters from this method to submit predictions for the bonus round. We heuristically combined the tlCLR-Inferelator topology predictions with the topology predictions from MCZ to assess if the two are complemetary. We refer to this method as tlCLR-Inferaltor+MCZ, and submitted the corresponding predictions of topology, placing 8th out of 19 teams. We revisited the DREAM4 challenge after the results were made available and developed a method that combined a resampling version of tlCLR-Inferelator with MCZ . This method achieved optimal performance in reconstructing topology compared to the other three methods, while retaining the ability to predict the response of the system to new conditions. We refer to this method as Resampling+MCZ.

Correcting the wild-type measurements for stochastic effects

The underlying model for the expression data in DREAM4 was generated by stochastic differential equations. Each measurement can be thought of as the observation of only a few cells, as opposed to a population of cells. Therefore, each measurement of wild-type expression, contained in X^{wt}, is an estimate of the population wild-type expression derived from only a few samples, making it a relatively noisy observation. Thus, any single observation will not accurately describe the population wild-type expression, and methods that rely on such statistics (e.g. t-test) will suffer. To correct for this we need to increase the sample size. By considering all wild-type measurements and than taking the mean (or median) of the expression levels for each gene we can improve our estimate of the population mean. Median has the advantage of being more robust to outliers than the mean. We can further improve our estimate of the population wild-type expression by taking the median of x_i not only with respect to the wild-type observations, X^{wt}, but also with respect to the genetic knockout data, X^{ko}. We can do so under the assumption that the networks are sparse (i.e. each gene is regulated by relatively few regulators). Thus, in most single knockout observations the level of most genes will remain close to the wild-type expression. Accordingly, we consider the wild-type expression of gene x_i to be the median of its expression in X^{wt} and X^{ko}, and denote this median-corrected estimate of wild-type expression as $x^{wt'} = (x_1^{wt'}, x_2^{wt'}, \dots x_n^{wt'})$.

Learning topology using genetic knockout data

Previously, we have observed that the genetic knockout data, X^{ko}, is very informative in regards to the topology of the network [31]. Moreover, Yip et al. [48], showed that a simple global noise model to filter out non-significant interactions using only genetic knockout data was able to produce regulatory interaction ranks of high quality, resulting in the top-performing method for the DREAM3 *in-silico* network challenge. However, for DREAM4 the noise for each gene is a function of the gene's expression (higher noise for higher expression), more accurately simulating the noise found in real microarray measurements. Thus, we use a method that can take advantage of the genetic knockout dataset, and uses a more biologically relevant gene-specific noise model to predict the topology of the network. A natural way of identifying if a gene, x_i, is a target of a TF, x_j, is by comparing the expression level of x_i when x_j is knocked out to the corrected wild-type expression of x_i, $x_i^{wt'}$. We do so using the Z-score:

$$z(x_i|x_j(-/-)) = \frac{x_{i,j} - x_i^{wt'}}{\sigma_i + \sigma_o} \tag{1}$$

Where the notation $(-/-)$ indicates a knockout experiment (i.e. $z(x_i|x_j(-/-))$ corresponds to the Z-score of target gene x_i given that x_j is knocked out), $x_{i,j}$ is the expression of gene x_i when x_j is knocked out, σ_i is the standard deviation of gene x_i over X^{ko}, and σ_0 is a correction term (to bias against genes with very low standard deviation). To determine the proper value of σ_0 we tested this method on the DREAM3 100 gene networks, finding that setting σ_0 to zero performed best (this is not the case for the smaller 10 gene networks). We confirmed, once the gold-standard networks were made available, that the best σ_0 was zero for the DREAM4 100 gene networks. This implies that having 100 knockout conditions provides a good enough estimate of the population standard deviation. We use $z(x_i|x_j(-/-))$ as a measure of confidence for each regulatory interaction $x_j \rightarrow x_i$, which we store in:

$$Z^{ko} = \begin{pmatrix} z^{ko}_{1,1(-/-)} & z^{ko}_{1,2(-/-)} & \cdots & z^{ko}_{1,N(-/-)} \\ z^{ko}_{2,1(-/-)} & z^{ko}_{2,2(-/-)} & \cdots & z^{ko}_{2,N(-/-)} \\ \vdots & \vdots & \ddots & \vdots \\ z^{ko}_{N,1(-/-)} & z^{ko}_{N,2(-/-)} & \cdots & z^{ko}_{N,N(-/-)} \end{pmatrix} \tag{2}$$

We refer to the method described above as median corrected Z-scores (MCZ). We ranked the absolute value of the entries in Z^{ko} (from high to low), and submitted the resultant ranked list of regulatory interactions as our topology prediction for the DREAM4 competition. This method was a co-best performer (along with the team of Pinna et al.) in reconstructing the topology of the network. Although it is attractive in its simplicity, it relies on a dataset that is more complete than we can expect a real genomics data-collection to be. Additionally, this method is not satisfactory if the end goal is to model system-wide behavior following new stimuli or new genetic perturbations as it does not assign dynamical parameters to each regulatory interaction. Predicting how a system will behave under previously unobserved conditions, such as the removal of certain regulators, is a key problem with a wide range of applications and a focus of the remainder of this section. Thus, we implemented an inference pipeline that infers dynamical parameters as well as topology. We refer to this pipeline as tlCLR-Inferelator, and it is very similar to the method we used for the DREAM3 competition [31]. This pipeline is composed of two steps: 1) using Time-Lagged Contex Lilehood of Relatedness (TL-CLR) [31], a mutual information based method, to learn directed topology, and 2) optimization and parameterization of the topology learned by TL-CLR via Inferelator 1.0 [32].

Double knockout prediction

For learning topology we were able to generate high-quality predictions using solely X^{ko}. Conversely, when learning dynamical parameters we used all of the data, emphasizing the distinction between the time-series data and steady-state data. We treated all of the time series experiments as a single dataset, X^{ts}, with columns $t_1, t_2, \ldots, t_K$ corresponding to the K observation times, and all of the steady state data (X^{wt}, X^{ko}, X^{kd}) together as another dataset, X^{ss}. In order to make predictions for the double knockout challenge we needed to infer topology and dynamical parameters. To improve the accuracy of our approach we filtered out the least likely regulatory interactions (using the scores in MCZ), and utilized internal metrics of model performance to reduce the effect of incorrect models. Our procedure for calculating double knockout predictions has five steps: 1) computing a directed topology using TL-CLR, 2) filtering out least likely interactions using MCZ, 3) optimizing and fitting parameters to the inferred topology via Inferelator 1.0, 4) estimating the relative merit of all model components, and 5) generating double knockout predictions.

Step 1: Computing the most likely predictors for each gene using TL-CLR

Mutual information (MI) as a metric of statistical dependency between two random variables X and Y can be defined as [25]

$$I(X;Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \tag{3}$$

where $p(x,y)$ is the joint probability distribution function of X and Y, and $p(x)$ and $p(y)$ are the marginal probabilities that $X = x$ and $Y = y$, respectively. Note that MI is a symmetric measure. Faith et al. [29] have previously shown that Context Likelihood of Relatedness (CLR), a MI based method, performed well at identifying a large portion of the known *E.coli* regulatory associations as well as predicting novel interactions. However, CLR can only predict undirected edges (regulatory associations), and must rely on additional data to determine directionality (e.g. by knowing that one gene encodes for a TF and the other for an enzyme, directionality can be resolved). By taking advantage of the temporal information available from time-series observations, we have shown that CLR can be extended (in a method we call TL-CLR), allowing us to infer directed regulatory interactions, and increasing overall performance [31]. At the core of TL-CLR's ability to resolve directionality is its reliance on dynamic-MI instead of static-MI; below, we make our distinction between static- and dynamic-MI clear and explain how to compute each measure.

As previously suggested [26–29], MI can be used as a measure of similarity between the expression levels of gene-pairs, $I(x_i, x_j)$, where gene-pairs that show a significantly higher MI scores (compared to other gene-pairs) are more likely to have a regulatory interaction between them. Since $I(x_i, x_j) = I(x_j, x_i)$ both regulatory edges ($x_j \to x_i$ and $x_i \to x_j$) are equally likely. We refer to the MI calculated from $I(x_i, x_j)$ as static-MI, because it does not use the temporal information available from time-series data (treating time-series and steady-state data identically).

We now describe dynamic-MI, which is motivated by our previous work on the Inferelator 1.0 [40], an ODE-based method. We assume that the temporal changes in expression of each gene, x_i, can be approximated by the linear ODE:

$$\frac{dx_i(t)}{dt} = -\alpha_i x_i + \sum_{j=1}^{N} \beta_{i,j} x_j(t), \qquad i = 1, \ldots, N \tag{4}$$

where $\alpha_i > 0$ is the first-order degradation rate of x_i and the $\beta_{i,j}$'s are a set of dynamical parameters to be estimated. The value of $\beta_{i,j}$ describes the extent and sign of the regulation of target gene x_i by regulator x_j. We store the dynamical parameters in an $N \times N$ matrix, $\boldsymbol{\beta}$. Note that $\boldsymbol{\beta}$ is typically sparse, i.e. most entries are 0. Using a finite difference approximation, we can write (4) for time-series experiments as

$$\tau_i \frac{x_i(t_k + 1) - x_i(t_k)}{t_{k+1} - t_k} + x_i(t_k) = \tau_i \sum_{j=1}^{N} \beta_{i,j} x_j(t_k), \qquad i = 1, \ldots, N \qquad k = 1, \ldots, K \tag{5}$$

where $\tau_i = \frac{1}{\alpha_i}$ is related to the half-life time, $t_{1/2}$, of x_i by $t_{1/2} = \tau_i \log(2)$. For every gene pair (x_i, x_j), we define a time-series response variable,

$$y_i(t_{k+1}) = \tau_i \frac{x_i(t_{k+1}) - x_i(t_k)}{t_{k+1} - t_k} + x_i(t_k), \tag{6}$$

and a corresponding explanatory variable, $x_j(t_k)$, both derived from the left and right hand sides of (5), respectively. For steady state experiments, the derivative, $\frac{dx_i(t)}{dt}$, in (4) equals zero, and we can write (4) as

$$x_i(v_l) = \tau_i \sum_{j=1}^{N} \beta_{i,j} x_j(v_l), \qquad i = 1, \ldots, N \qquad l = 1, \ldots, 2N + 1 \tag{7}$$

Similarly, we define a steady-state response variable,

$$y_i(v_l) = x_i(v_l), \tag{8}$$

and a corresponding explanatory variable both derived from the left and right hand sides of (7), respectively. Taking the time-series and steady-state response variables together, we get the final response vector: $y_i = (y_i(t_2), \ldots, y_i(t_K), y_i(v_1), \ldots, y_i(v_{2N+1})$. Similarly, taking the corresponding time-series and steady-state explanatory variables together, we get the final explanatory variables vector:
$x_j = (x_j(t_1), \ldots, x_j(t_{K-1}), x_j(v_1), \ldots, x_j(v_{2N+1})$. Note that for time-series data the explanatory variables are time-lagged with respect to the response, as we desire.

As a measure of confidence for a directed regulatory interaction between a pair of genes ($x_j \to x_i$) we use, $I(y_i, x_j)$, where a pair that shows a high MI score (relative to other pairs) is more likely to represent a true regulatory interaction. Note that $I(y_i, x_j) \neq I(y_j, x_i)$, making one regulatory direction more likely than the other. We refer to the MI calculated from $I(y_i, x_j)$ as dynamic-MI, as it takes advantage of the temporal information available from time-series data (distinguishing time-series data from steady-state data).

As described above, we calculate $I(x_i, x_j)$ and $I(y_i, x_j)$ for every pair of genes and store the values in the form of two $N \times N$ matrices M^{stat} and M^{dyn}, respectively. Note that M^{stat} is symmetric while M^{dyn} is not. We now briefly describe how TL-CLR integrates both static- and dynamic-MI to produce a final confidence score for each regulatory interaction (unlike for CLR, TL-CLR confidence scores will not be symmetric). For a more detailed explanation we refer the reader to [31]. For each regulatory interaction $x_j \to x_i$ we compute two positive Z-scores (by setting all negative Z-scores to zero): one for the regulation of x_i by x_j based on dynamic-MI (i.e. based on M^{dyn}), z_1, and one for the regulation of x_i by x_j based on static-MI (i.e. based on M^{stat}), z_2. We combine the two scores into a final TL-CLR score, $z_{i,j}^{tl-clr} = \sqrt{(z_1^2 + z_2^2)}$. Note, that some entries in Z^{tl-clr} are zero, i.e. Z^{tl-clr} is somewhat sparse.

Step 2: Filtering least likely regulatory interactions using knock out data

We now apply a simple filtration step (similar to the one used in [31]) using the genetic knockout data, X^{ko}, with the aim of removing the least likely regulatory interactions from Z^{tl-clr} (by setting their scores to zero). For the results presented here, we filtered every regulatory interaction with MCZ score (stored in Z^{ko}) below the 50th percentile of scores in Z^{ko}. Z^{tl-clr} now contains only the putative interactions we are most sure of, and we can interpret row i of the filtered Z^{tl-clr} as a ranking of the regulators, x_j's, for gene x_i. Similar to the confidence scores in Z^{ko}, the confidence scores in Z^{tl-clr} recapitulate only the topology, not the dynamics of regulation, and cannot be used to make predictions of the response of the system to new conditions.

Step 3: Learning dynamical parameters using Inferelator 1.0

We use Inferelator 1.0 to learn a sparse dynamical model of regulation for each gene x_i. As potential regulators of x_i we consider only the P highest confidence (non-zero) regulators from Z^{tl-clr} (highest scoring x_j's, in row i of Z^{tl-clr}). Note that we cannot guarantee that every x_i will have P regulators meeting this criteria, thus we denote by $P^i (P^i \leq P)$ the number of regulators that do. Accordingly, for each gene, x_i, we denote this subset of potential regulators as x^i. We use Inferelator 1.0 to learn a sparse dynamical model for each x_i as a function of x^i's. We assume that the time evolution in the x_i's is governed by

$$\frac{dx_i(t)}{dt} = -\alpha_i x_i + \sum_{j=1}^{P^i} \beta_{i,j} x_j^i(t), \qquad i = 1, \ldots, N \tag{9}$$

which is exactly (4) with our constraint on the number of regulators. Least Angle Regression (LARS) [39] is used to efficiently implement an l_1 constraint on $\boldsymbol{\beta}$, which minimize the following objective function, amounting to a least-square estimate based on the ODE (9):

$$\mathcal{E}(\boldsymbol{\beta}) = \sum_{i=1}^{N} \mathcal{E}_i(\boldsymbol{\beta}) \tag{10}$$

where

$$\mathcal{E}_i(\boldsymbol{\beta}) = \sum_{k=1}^{K-1} \left| \frac{x_i(t_{k+1}) - x_i(t_k)}{t_{k+1} - t_k} + \alpha_i x_i(t_k) - \sum_{j=1}^{P^i} \beta_{i,j} x_j^i(t_k) \right|^2 \tag{11}$$

under an l_1-norm penalty on regression coefficients,

$$\sum_{j=1}^{P^i} |\beta_{i,j}| \leq s_i \sum_{j=1}^{P^i} |\beta_{i,j}^{\text{ols}}| \tag{12}$$

where $\boldsymbol{\beta}^{\text{ols}}$ is the over-fit ordinary least-squares estimate (i.e. the minimizer of (11) with no penalty), and s_i is a number between 0 and 1 referred to as the shrinkage parameter; setting $s_i = 1$ corresponds to ordinary least-square regression. To avoid over fitting, we chose the shrinkage parameter s_i by ten fold cross-validation at one standard deviation away from the minimum error (as described in [32]). Each resultant model (row of $\boldsymbol{\beta}$) is a parameterization of an ODE describing the temporal evolution of x_i. The l_1 constraint ensures that Inferelator 1.0 results in a sparse matrix, $\boldsymbol{\beta}$, with a small number of entries $|\beta_{i,j}| > 0$. These entries are dynamical parameters that can be used to predict the response of the system to new conditions, such as the removal of genes or future time-points (given initial time points in a time series).

Step 4: Estimating the relative explanatory power of each model and regulatory interaction

The dynamical parameters stored in $\boldsymbol{\beta}$ describe the regulation of each gene (target) as a function of regulators (TF's) in the system, with $|\beta_{i,j}|$ corresponding to the strength of the regulation, and the sign of $\beta_{i,j}$ indicating repression or activation. For the DREAM3 *in-silico* challenge we ranked regulatory interactions using $|\beta_{i,j}|$ as the measure of confidence for a regulatory interaction ($x_j \rightarrow x_i$) [31]. However, this ranking does not take into account the explanatory power of each predictor x_j in the ODE model for a target x_i (e.g. $|\beta_{i,j}|$ may be large even though the model for the regulation of x_i is not a good one). Here, we propose a confidence measure that incorporates the explanatory power of predictors (i.e. the quality of the model for x_i). When inferring a regulatory model for x_i we treated the finite difference approximated values, y_i, as the response variable, and modeled y_i as a function of corresponding explanatory variables, x_j, $j = 1, \ldots, N$ $(j \neq i)$. We refer to the predicted expression of y_i as $\hat{y}_i$, calculated as $\hat{y}_i = \tau_i \sum_{j=1}^{P^i} x_j^i \boldsymbol{\beta}_{i,j}$. The error in this approximation was measured as sum-of-squares, $\sum_{i=1}^{M} (y_i - \hat{y}_i)^2$, where M is the total number of observations. We estimated the predictive error of our model for y_i using mean error obtained from ten fold cross-validation. In order to place all model errors on the same scale, we normalized the absolute sum-of-squares error to derive a measure of relative error, $\frac{\sum_{i=1}^{m} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{m} y_i^2}$. Given this relative error, we defined the explanatory power of the model for y_i to be given by 1 minus relative error:

$$\gamma_i = 1 - \frac{\sum_{i=1}^{M} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{M} y_i^2} \tag{13}$$

where γ_i represents the merit of the model for y_i (i.e. how good of an estimate is $\hat{y}_i$). We can now calculate the contribution of each predictor $\beta_{i,j}$ to the explanatory power of the model for y_i, (i.e. the explanatory power of each regulatory interaction) as a weighted average

$$z_{i,j}^{\gamma} = \frac{\beta_{i,j}}{\beta_{i,0} + \sum_{j=1}^{N} \beta_{i,j}} \gamma_i \tag{14}$$

where $\beta_{i,0}$ is the bias term for the regulatory model of y_i. Note that here we use the fact that all the observations of the regulators x_j's, are on the same scale, as they were normalized to have zero mean and standard deviation of 1 before model selection by Inferlator 1.0 (a common step in a regression framework). We stored these values in the form of an $N \times N$ matrix, Z^{γ}. Next, we describe how we used $\boldsymbol{\beta}$ to generate predictions of the system's response to double knockouts.

Step 5: Generating the double knockout predictions

The challenge of predicting the response of the system to double knockouts (double-KO) can be phrased as: given a simultaneous knockout of two genes (i.e. $x_i, x_j = 0$ for some i and j), predict the steady-state expression of all other genes. In order to predict steady-state expression levels for each gene we used the steady-state limit of the core Inferelator 1.0 model [32] (7), which we rewrite here (in matrix notation) for the case of predicting the steady state data:

$$x^{ij(-/-)} = \tau \boldsymbol{\beta} x^0 \tag{15}$$

where $x^{ij(-/-)}$ is the level of all genes for the double-KO of genes x_i and x_j, and x^0 is some vector of initial conditions (satisfying $x_i, x_j = 0$). Throughout we had set τ_i to 40 (for all genes), which is the time-interval between measurements, assuming that the sampling frequency was on the time order of most regulatory reactions. Note that the only unknown left to determine (before we can make a prediction) is the vector of initial conditions, x^0. The rest of this section deals with computing a good initial condition vector. A simple way to pick this vector would be to set $x^0 = x^{wt'}$, with the exception that $x_i^0, x_j^0 = 0$. The results that we submitted for the DREAM4 bonus-round challenge were calculated using this initial condition. Note, however, that the system's response to the KO of genes x_i, x_j individually was already given to us in the single-gene knockout dataset, X^{ko}. Upon revisiting our initial results, after submission of the predictions, we reasoned that using the single gene knockout (single-KO) information to predict double-KO expression would most likely yield better results, as it reflects a system state that is closer to the state we are trying to predict. Indeed, using the single-KO data to determine initial conditions markedly improved the accuracy of our double knockout predictions. Next, we describe how we used single-KO observations to determine initial conditions.

It is clear that in order to determine x^0, given that x_i and x_j have been knocked out, it would be advantageous to use the single-KO observations of x_i and x_j. The remaining question is how to integrate the two single-KO observations into a single initial condition. One approach is to simply take their mean. However, a more informed approach is to use our previous knowledge regarding likely regulatory interactions, such as the confidence scores from MCZ (stored in Z^{ko}). We do so by computing the following weighted average:

$$x_l^0 = \frac{z_{l,i}^{ko} x_{l,i}^{ko} + z_{l,j}^{ko} x_{l,j}^{ko}}{z_{l,i}^{ko} + x_{l,j}^{ko}} \qquad l = 1, \ldots, K \tag{16}$$

where x_l^0 is our estimate for the initial expression level of gene x_l, $x_{l,i}^{ko}$ and $x_{l,j}^{ko}$ are the observed levels of x_l when genes x_i and x_j were knocked out, respectively, and $z_{l,i}^{ko}$ or $z_{l,j}^{ko}$ are the confidence scores (calculated by MCZ) for each regulatory interaction $x_i \to x_l$ and $x_j \to x_l$, respectively. In this manner we computed an initial condition vector, x^0, for every double-KO we were asked to predict. We now use these initial conditions to calculate a prediction of the expression of all genes in the presence of a double-KO of x_i, x_j via (15). We denote this prediction as $\tilde{x}^{ij(-/-)}$.

Note, however, that some models had more predictive merit than others, as measured by the explanatory power of each model (13). Thus we weighted the prediction of double knockouts by the predictive merit of each model. We computed the final double-KO predictions as follows:

$$x^{ij(-/-)} = \tilde{x}^{ij(-/-)}\gamma + x^0(\mathbf{1} - \gamma) \tag{17}$$

where $\gamma = (\gamma_1, \gamma_2, \dots, \gamma_N)^T$. Note that in (17) the final prediction $x^{ij(-/-)}$ is weighted by our estimate of the predictive performance the models, γ calculated in (13), and constrained, using the initial conditions, by our estimate of the model errors $(1 - \gamma)$.

Ranking regulatory interactions using their explanatory power, TL-CLR-Inferelator 1.0 confidence scores, and MCZ confidence scores

It is natural to think that the regulatory information gathered using different datasets and different methods is complimentary. Thus, we combined the results of the measures of confidence described above: Z^{ko}, Z^{tl-clr}, and Z^{γ}. The main challenge in combining these confidence scores is that they are not guaranteed to be on the same scale. Thus, we developed a single rank based heuristic (described previously in [31]) to combine two separate sets of confidence scores. Our approach is best explained by an example: Let Z^{pl} denote the resultant matrix from combining the confidence scores contained in Z^{tl-clr} and Z^{γ} (i.e. the results of our dynamic pipeline TL-CLR $\rightarrow$ Inferelator 1.0). We first replace the value of the highest-ranking entry in Z^{γ} with the value of the highest-ranking entry from Z^{tl-clr}. We then replace the value of the second highest-ranking entry from Z^{γ} with the value of the second highest-ranking entry from Z^{tl-clr}. We continue in such a way until all entries in Z^{γ} have been replaced by equally ranked entries in Z^{tl-clr}. This produces two ranked lists of regulatory interactions that are on the same scale. Once this assignment is done we can combine the two matrices as follows:

$$Z^{pl} = \sqrt{(Z^{\gamma})^2 + (Z^{tl-clr})^2}. \tag{18}$$

Note that here Z^{γ} refers to the matrix after the assignment of values from Z^{tl-clr}. We used the confidence scores in Z^{pl} to generate the ranked list of regulatory interactions for the TL-CLR $\rightarrow$ Inferelator 1.0 pipeline. We will refer to this method as tlCLR-Inferelator.

In order to assess how complimentary tlCLR-Inferelator and MCZ were we combined the confidence scores stored in Z^{γ} with those in Z^{tl-clr} (replacing scores of equal ranks from Z^{ko} into Z^{pl}, as above):

$$Z^{final} = \sqrt{(Z^{pl})^2 + (Z^{ko})^2}. \tag{19}$$

The confidence scores contained in Z^{final} were generated by a combination of our three methods, and we will refer to this integrated method as tlCLR-Inferelator+MCZ. Next we describe a resampling method that we used to generate an ensemble of predicted neworks. Picking the median network based on the ensemble (we describe this step below) dramatically improved our performance in recapitulating network topology, showing that the two methods, tlCLR-Inferelator and MCZ, held unique information with regard to the network topology, and were indeed complimentary (at least in this resampling approach).

Combining genetic and dynamic information in a resmapling approach

We generated an ensemble of networks, as opposed to a single network, as follows. Consider all of the provided observations as a single $N \times M$ matrix, $D = \{X^{wt}, X^{ko}, X^{kd}, X^{ts}\}$, with M being the number of total observations (columns). Let $c = \{1, \dots, M\}$ be the vector of column indices. We sample with replacement M times from c, storing the selected indices in $c^* = i_1, i_2, \dots, i_j, \dots, i_M$, $i_j \in \{1, \dots, M\}$. We now consider a permuted data matrix, D^*, comprised of the c^* columns of D. We generate $\boldsymbol{\beta}$, Z^{pl}, and Z^{final}, as described before, with the only difference being that we use D^* instead of D. We repeat this procedure B times, with $B = 200$ for the DREAM4 networks, each time generating $\boldsymbol{\beta}$, Z^{pl}, and Z^{final}. We store this ensemble of regulatory network predictions in:

$$E = \{[\boldsymbol{\beta}(1), Z^{pl}(1), Z^{\text{final}}(1)], [\boldsymbol{\beta}(2), Z^{pl}(2), Z^{\text{final}}(2)], \dots, [\boldsymbol{\beta}(B), Z^{pl}(B), Z^{\text{final}}(B)]\} \tag{20}$$

where $[\boldsymbol{\beta}(b), Z^{pl}(b), Z^{\text{final}}(b)]$ corresponds to the dynamical parameters, rankings based on tlCLR-Inferelator, and rankings based on tlCLR-Inferelator+MCZ, respectively at resample $b = 1, 2, \ldots, B$. Note that throughout this resampling procedure the ranks generated by MCZ remain constant. We used this ensemble of network predictions by selecting, for each regulatory interaction $x_j \to x_i$ (corresponding to entries $\beta_{i,j}$ and $z_{i,j}^{final}$), the median dynamical parameters in $\{\boldsymbol{\beta}(1), \boldsymbol{\beta}(2), \ldots, \boldsymbol{\beta}(B)\}$ and the median tlCLR-Inferelator+MCZ rank in $\{Z^{final}(1), Z^{final}(2), \ldots Z^{final}(B)\}$. We store these median values in $\boldsymbol{\beta}^{\text{eMed}}$ and Z^{eMed}, respectively. These matrices have entries:

$$z_{i,j}^{\text{eMed}} = \text{median}(z_{i,j}^{\text{final}}(1), z_{i,j}^{\text{final}}(2), \ldots, z_{i,j}^{\text{final}}(B)) \tag{21}$$

$$\beta_{i,j}^{\text{eMed}} = \text{median}(\beta_{i,j}(1), \beta_{i,j}(2), \ldots, \beta_{i,j}(B)). \tag{22}$$

To rank regulatory interactions we used Z^{eMed}. To estimate dynamical parameters we used $\boldsymbol{\beta}^{eMed}$. We refer to this resampling approach as Resampling+MCZ.

Results

Performance of tested methods: ranking putative regulatory interactions

The main challenge in the DREAM4 100 gene *in-silico* regulatory network competition was to predict the topology of five networks. Predictions were made in the form of a list of regulatory interactions ranked in decreasing order by confidence. We evaluated the performance of four methods for learning regulatory networks, namely: MCZ (2), tlCLR-Inferelator (18), tlCLR-Inferelator+MCZ (19), and Resampling+MCZ (21). In all four cases we evaluated the quality of the rankings of all possible regulatory interactions, as this was the basis for the evaluation of performance in DREAM3 and DREAM4. We submitted the results of MCZ as our ranked list of regulatory interactions for the DREAM4 challenge. This method tied for first place (out of 19 teams). In Figure 1 we show that MCZ (a relatively simple method) performs well (red). The tlCLR-Inferelator pipeline exhibits lower performance for most of the networks (yellow), but has the advantage of allowing us to calculate dynamical parameters (which we show below offer the ability to predict the system's response to double knockouts). In the method tlCLR-Inferelator+MCZ we combine the predictions made by MCZ with those made by tlCLR-Inferelator (orange). We see an improvement in performance as compared to the performance of tlCLR-Inferelator alone, but do not outpreform MCZ. However, if we use a resampling approach, Resampling+MCZ (20—22), to generate an ensemble of likely networks, we see a marked improvement over the performance of any other method (shown in purple). This improvement is most evident in networks 3-5, which appear to be more difficult to predict for all of the methods we tested.

Performance of methods based on genetic knockout data decreases with decreasing expression of the regulators

For the DREAM3 *in-silico* challenge all methods, including several similar to the ones we test herein, were found to perform significantly worse for networks with very high in-degree (targets regulated by many TFs) and to be relatively insensitive, performance-wise, to the out-degree of TFs [31, 48]. We did not find this trend in the current challenge (Figure 2C,D). However, we did find that performance varies considerably across the five 100 gene networks for all tested methods; performance was best for the first network and dramatically worse for the fifth network (Figure 1). We investigated possible reasons for this, finding that performance is correlated with the median expression of the regulators. Given a regulatory interaction, $x_j \to x_i$, our chance of correctly predicting that regulatory interaction (based on MCZ) tends to be higher if the median expression of x_j over all conditions in the knockout data-set, X^{ko}, is high. Conversely, the smaller the median expression of x_j, the worse our performance.

Figure 2A shows that our predictions for the regulatory interactions in network 1 have relatively low error (black box plot), and the corresponding median expression of the regulators in this network is relatively high (grey box plot). For network 5 we see relatively poor predictive performance, and the corresponding median expression of the regulators in this network is the lowest of the five networks. In Figure 2B we see a high correlation, $R^2 = .95$, between the median expression of the regulators and the performance of MCZ in terms of AUPR. By combining ranks from MCZ with our resampled network inference pipeline, Resampling+MCZ, we significantly improve performance on networks 3-5 (Figure 1), and lower the correlation between performance and median TF expression over all five networks to 0.81 (Figure 2B).

Regulatory interaction rankings derived from genetic knockout data and rankings derived from the resampling tlCLR-Inferelator are complimentary

In the above section we focused on differences between the performance of each method for each of the five networks. In this section we focus on the performance of each method in a gene-by-gene manner, in an effort to better understand how to best utilize heterogeneous data collections. Specifically, we investigated the performance of each network inference method MCZ, tlCLR-Inferelator, tlCLR-Inferealtor+MCZ, and Resampling+MCZ as a function of the median expression of the regulators, x_j's, in the network. We bin regulators based on their median expression, and compare the error made in predicting their respective targets.

In Figure 3 we see the performance of the ranks from MCZ is worse for regulators with low median expression than for the most actively expressed regulators (shown in red). This trend is more apparent in this gene-by-gene view than in our network-centric analysis. Looking at each bin, shown from low to high median expression, we see that predictions made by methods that incorporate rankings made by tlCLR-Inferelator perform better than the predictions made by MCZ for regulators whose median expression is less than .2. The error distributions of the predictions made by Resampling+MCZ (purple bars) are lower than those of MCZ (red bars) for regulators with a median expression upto .4, and on par with the predictions made by MCZ for regulators with a median expression of up to .6. The predictions made by Resampling+MCZ are better than those made by tlCLR-Inferelator and tlCLR-Inferelator+MCZ for all bins.

Predicting the effect of double knockouts

For each 100-gene network we were asked to predict the cell's steady-state mRNA levels given that a pair of genes is knocked out. There are twenty such pairs of genes ($x_i, x_j = 0$) for each network. We make these predictions using the parameterization, $\boldsymbol{\beta}$, of the system obtained from our inference pipeline, tlCLR-Inferelator. We also make these predictions using the parameters obtained by taking the median weight from the ensemble, $\boldsymbol{\beta}^{eMed}$ (22), generated by Resampling+MCZ.

The measure of performance for the DREAM4 double knockout predictions was mean squared error (MSE). As a baseline, we compare the error of our prediction to the error we would make if we used $x^{wt'}$ as the prediction. We reason that a completely wrong parameterization will cause drastic changes resulting in a complete distortion of the system, hence the the wild-type expression is a satisfactory baseline.

In Figure 4 we take a gene-centric approach and see that both sets of predictions have lower error than the baseline (shown in green). Additionally, the error made by the baseline predictions increases significantly as the median expression of the knocked-out genes increases, while the error of our predictions increases only marginally. The predictions made by using parameters derived from Resample+MCZ show almost idenitical performance as those made by using the parameterization derived from tlCLR-Inferelator. Table 1 summarizes the accuracy of double knockout predictions for each network. We see that the performance of both sets of parameters is nearly indentical. Additionally we see (in columns

2 and 3) that using initial conditions derived from the wild-type expression, $x^{wt'}$, to make predictions by tlCLR-Inferelator beat the baseline but performed much worse than the predictions made if we use initial conditions based on the knockout data, X^{ko}. This is also the case for predictions made using Resampling+MCZ (data not shown).

Discussion

We participated in the DREAM4 100-gene *in-silico* network inference competition. The method that we submitted, and that was the co-best performer on the 100-gene *in-silico* challenge, was the rankings derived from the median corrected Z-scores of the genetic knockout data, MCZ. The power of the genetic knockout data, as also shown by Yip et al. in DREAM3 [48], is an important point to consider for experimental design. However, it does have limitations that can be complemented by integrating other data-types, particularly time-series data. We observed that as the median expression of the regulators in a network decreases, error in predicting regulatory interactions using MCZ increases (Figure 2). A plausible explanation for why a low median expression of regulators leads to poor performance is that if an active regulator (i.e. a regulator whose wild-type expression is high) is removed, then the corresponding effect on its targets will be relatively large. Conversely, if a regulator that is not very active (i.e. its wild-type expression is low) is removed, then the effect on its targets will be marginal. Perhaps, the targets of such regulators will be most apparent in over-expression experiments. If over-expression experiments are not available the poor performance in predicting the targets of these regulators can be mitigated by combining the predictions made by MCZ with predictions made by a method that takes advantage of time-series data (where the regulator was active at some point).

We used tlCLR-Inferelator, which takes advantage of the time-series data, to predict topology and dynamical parameters for each network in a way that was more robust to the median expression of the regulators than methods that use solely genetic knockout data. We submitted topology predictions and bonus-round (double knockout) predictions generated by tlCLR-Inferelator. The topology predictions ranked 8th out of 19 teams. Using a very similar inference pipeline for the DREAM3 challenge resulted in predictions that ranked 2nd out of 22 teams. Notably, in terms of AUPR the results of the inference pipeline on the DREAM4 networks were better then on the DREAM3 networks [31]. This discordance between a worse performance relative to other teams, but improved ability to recapitulate network topology is probably due to a more concentrated use of the knockout data by participants of DREAM4.

Upon receiving the gold-standard networks we analyzed our ability to predict topology using tlCLR-Inferelator. Dissecting our performance, in a gene-by-gene manner, we saw the that there are instances when predictions made by tlCLR-Inferelator are more accurate than those made by MCZ. Given the performance of each of the methods, as evaluated by AUPR (Figure 1), this is a surprising and promising result, implying that methods that use only genetic knockout data and those that take advantage of time-series data produce complimentary topology predictions. Further demonstrating this point, we showed that applying a resampling approach to tlCLR-Inferelator and combining the results with MCZ, by aggregating the ranks derived from each method, produces a final prediction that is better than the predictions generated by either method alone. The improvements from Resampling+MCZ are most evident on networks 3-5 (Figure 1), which have the lowest median expression of the regulators (Figure 2A), and are hence hardest to predict using the genetic knockout data alone. We note that alternate ways of combining predictions from multiple methods may further improve upon our results. We also note that in our resampling approach the predictions of MCZ remain constant for each network in the ensemble. This implies that although a single network generated by tlCLR-Inferelator may perform poorly, our resampling approach generates sufficient alternate topologies such that picking a network based on the ensemble-median produces a much more accurate topology prediction. This resampling approach also infers an ensemble of dynamical parameters, retaining the ability to predict the response of the network to new conditions.

We submitted predictions of system-wide expression in the presence of double knockouts for the DREAM4 bonus-round challenge. The predictions we submitted were based on the initial conditions derived from the wild-type ($x^{wt'}$). The quality of our double knockout predictions was very sensitive to the initial conditions (Table 1). We found that using single-knock out data, X^{ko}, as the basis of our initial conditions dramatically improves our predictive performance (compared to using initial conditions based on the wild-type). This is due to the fact that the single-gene knockouts present a closer network state to what we are trying to predict (network response to double knockouts) than does the wild-type. Once we properly picked the initial conditions we observed that we can accurately predict the response of the system to new genetic perturbations (Figure 4). We accurately predicted the response of the network to double knockouts using dynamical paramters calculated by tlCLR-Inferelator (whose topology prediction was poor relative to those of other methods). Thus, we show that our ability to predict data can tolerate a remarkable amout of error in the predicted topology and still make accurate predictions of the system's response to new perturbations. This is perhaps not surprising, as the Inferelator 1.0 [32] was designed to optimize data prediction error. We have also shown that a parameterization picked from the median of an ensemble of networks retains, but does not significantly improve, our predictive performance. Perhaps an alternative way of picking parameters from the ensemble of networks can improve upon the ability to predict new data.

We have shown the complementarity between predictions made using genetic knockout data and those made using time series data. We have shown that using solely genetic knockout data can result in accurate topology predictions, which can be further improved upon by correctly incorporating predictions made using time-series data. To this end, we have developed a relatively simple method for combining the predictions made from genetic knockout and time-series datasets, showing an improved ability to infer network topology while maintaing the ability to predict the response of the system to new conditions. We suggest that investigating alternate means of combining genetic and dynamic experimental designs (leveraging the complementarity between these two data-types), as well as methods that incorporate direct binding data, will continue to be fruitful avenues of future investigation.

Acknowledgments

The authors would like to thank Dennis Shasha, David Reiss, Eric Vanden-Eijnden, Yongzhao Shao, Thadeous Kacmarczyk, Peter Waltman, and Francis Kirigin, for their helpful comments and discussions. We would like to thank the DREAM4 organizers for their tireless efforts.

References

1. Marbach D, Schaffter T, Mattiussi C, Floreano D (2009) Generating Realistic in silico Gene Networks for Performance Assessment of Reverse Engineering Methods. Journal of Computational Biology 16: 229–239.

2. Haynes BC, Brent MR (2009) Benchmarking regulatory network reoncstruction with GRENDEL. Bioinformatics 25: 801–807.

3. Mendes P, Sha W, Ye K (2003) Artificial gene networks for objective comparison of analysis algorithms. Bioinformatics 19: 122–129.

4. Cantone I, Marucci L, Iorio F, Ricci MA, Belcastro V, et al. (2009) A yeast synthetic network for in vivo assessment of reverse-engineering and modeling approaches. Cell 137: 172–81.

5. Prill RJ, Marbach D, Saez-Rodriguez J, Sorger PK, Alexopoulos LG, et al. (2010) Towards a rigorous assessment of systems biology models: the DREAM3 challenges. PloS one 5: e9202.

6. Horak CE, Snyder M (2002) ChIP-chip: a genomic approach for identifying transcription factor binding sites. Methods in enzymology 350: 469–83.

7. Johnson DS, Mortazavi A, Myers RM, Wold B (2007) Genome-wide mapping of in vivo protein-DNA interactions. Science 316: 1497–502.

8. Eisen MB, Spellman PT, Brown PO, Botstein D (1998) Cluster analysis and display of genome-wide expression patterns. Proceedings of the National Academy of Sciences of the United States of America 95: 14863–8.

9. Tanay A, Sharan R, Shamir R (2002) Discovering statistically significant biclusters in gene expression data. Bioinformatics 18: 136–144.

10. Ihmels J, Friedlander G, Bergmann S, Sarig O, Ziv Y, et al. (2002) Revealing modular organization in the yeast transcriptional network. nature genetics 31: 370378.

11. Bergmann S, Ihmels J, Barkai N (2003) Iterative signature algorithm for the analysis of large-scale gene expression data. Physical review E 67: 31902.

12. Reiss DJ, Baliga NS, Bonneau R (2006) Integrated biclustering of heterogeneous genome-wide datasets for the inference of global regulatory networks. BMC bioinformatics 7.

13. Mardis ER (2007) ChIP-seq: welcome to the new frontier. Nature methods 4: 613–4.

14. Schmid CD, Bucher P (2007) ChIP-Seq data reveal nucleosome architecture of human promoters. Cell 131: 831–2; author reply 832–3.

15. Facciotti MT, Reiss DJ, Pan M, Kaur A, Vuthoori M, et al. (2007) General transcription factor specified global gene regulation in archaea. Proceedings of the National Academy of Sciences of the United States of America 104: 4630–5.

16. Lee TI, Rinaldi NJ, Robert F, Odom DT, Bar-Joseph Z, et al. (2002) Transcriptional regulatory networks in Saccharomyces cerevisiae. Science 298: 799–804.

17. Gama-Castro S, Jiménez-Jacinto V, Peralta-Gil M, Santos-Zavaleta A, Peñaloza Spinola MI, et al. (2008) RegulonDB (version 6.0): gene regulation model of Escherichia coli K-12 beyond transcription, active (experimental) annotated promoters and Textpresso navigation. Nucleic acids research 36: 120–4.

18. Matys V, Wingender E (2003) TRANSFAC(R): transcriptional regulation, from patterns to profiles. Nucleic Acids Research 31: 374–378.

19. D'haeseleer P, Shoudan L, Somogyi R (2000) Genetic network inference: from co-expression clustering to reverse engineering. Bioinformatics 16: 707–726.

20. Smith VA, Jarvis ED, Hartemink AJ (2003) Influence of network topology and data collection on network inference. Pacific Symposium on Biocomputing Pacific Symposium on Biocomputing : 164–75.

21. Bonneau R (2008) Learning biological networks: from modules to dynamics. Nature Chemical Biology 4: 658–664.

22. He F, Balling R, Zeng Ap (2009) Reverse engineering and verfication of gene networks: principles, assumptions and limiations of present methods and future perspectives. Journal of Biotechnology 144: 190–203.

23. Hecker M, Lambeck S, Toepfer S, Someren EV, Guthke R (2009) Gene regulatory network inference: Data integration in dynamic modelsA review. BioSystems 96: 86–103.

24. Bansal M, Belcastro V, Ambesi-Impiombato A, di Bernardo D (2007) How to infer gene networks from expression profiles. Molecular systems biology 3: 78.

25. Shannon C (1948) A Mathematical Theory of Communication. The Bell System Technical Journal 27: 379–423.

26. Basso K, Margolin Aa, Stolovitzky G, Klein U, Dalla-Favera R, et al. (2005) Reverse engineering of regulatory networks in human B cells. Nature genetics 37: 382–90.

27. Margolin AA, Nemenman I, Basso K, Wiggins C, Stolovitzky G, et al. (2006) ARACNE: An Algorithm for the Reoncstruction of Gene Regulatory Networks in a Mammalian Cellular Context. BMC Bioinformatics 15: 1–15.

28. Butte AJ, Kohane IS (2000) Mutual information relevance networks: functional genomic clustering using pairwise entropy measurements. Proceedings of the 5th Pacific Symposium on Biocomputing : pp418–429.

29. Faith JJ, Hayete B, Thaden JT, Mogno I, Wierzbowski J, et al. (2007) Large-Scale Mapping and Validation of Escherichia coli Transcriptional Regulation from a Compendium of Expression Profiles. PLoS Biology 5: 54–66.

30. Liang KC, Wang X (2008) Gene regulatory network reconstruction using conditional mutual information. EURASIP Journal on Bioinformatics & Systems Biology 2008.

31. Madar A, Greenfield A, Vanden-eijnden E, Bonneau R DREAM3: Network Inference Using Dynamic Context Likelihood of Relatedness and the Inferelator. PloS one .

32. Bonneau R, Reiss DJ, Shannon P, Facciotti MT, Hood L, et al. (2006) The Inferelator: an algorithm for learning parsimonious regulatory networks from systems-biology data sets. Genome Biology 7.

33. Mazur J, Ritter D, Reinelt G, Kaderali L (2009) Reconstructing nonlinear dynamic models of gene regulation using stochastic sampling. BMC Bioinformatics 10.

34. Madar A, Greenfield A, Ostrer H, Vanden-Eijnden E, Bonneau R (2009) The inferelator 2.0: A scalable framework for reconstruction of dynamic regulatory network models. Conference proceedings : Annual International Conference of the IEEE Engineering in Medicine and Biology Society IEEE Engineering in Medicine and Biology Society Conference 1: 5448–51.

35. James, Gardner TS, Collins JJ, di Bernardo D, Lorenz D (2003) Inferring Genetic Networks and Identifying Compound Mode of Action via Expression Profiling. Science : 102–105.

36. van Someren EP, Vaes BLT, Steegenga WT, Sijbers AM, Dechering KJ, et al. (2006) Least absolute regression network analysis of the murine osteoblast differentiation network. Bioinformatics 22: 477–84.

37. Bansal M, Gatta GD, di Bernardo D (2006) Inference of gene regulatory networks and compound mode of action from time course gene expression profiles. Bioinformatics 22: 815–22.

38. di Bernardo D, Thompson MJ, Gardner TS, Chobot SE, Eastwood EL, et al. (2005) Chemogenomic profiling on a genome-wide scale using reverse-engineered gene networks. Nature Biotechnology 23: 377–83.

39. Efron B, Hastie T, Johnstone I, Tibshirani R (2004) Least angle regression. Annals of statistics data: 407451.

40. Bonneau R, Facciotti MT, Reiss DJ, Schmid AK, Pan M, et al. (2007) A Predictive Model for Transcriptional Control of Physiology in a Free Living Cell. Cell 131: 1354–1365.

41. Alvarez M, Luengo D, Lawrence N (2009) Latent Force Models. Proceedings of the Twelfth International Workshop on Artificial Intelligence and Statistics 5: 916.

42. Efron B, Tibshirani R (1993) An Introduction to the Bootstrap. Chapman & Hall.

43. Shasha D, Manda W (2008) Statistics is Easy! Morgan & Claypool.

44. Kerr MK, Churchill GA (2001) Bootstrapping cluster analysis: assessing the reliability of conclusions from microarray experiments. Proceedings of the National Academy of Sciences of the United States of America 98: 8961–5.

45. Kirk P, Stumpf M (2009) Gaussian process regression bootstrapping: exploring the effects of uncertainty in time course data. Bioinformatics 25: 1300.

46. Friedman N, Goldszmidt M, Wyner A Data analysis with Bayesian networks: A bootstrap approach. In: UAI99. Citeseer, p. 206215.

47. Marbach D, Mattiussi C, Floreano D (2009) Combining multiple results of a reverse-engineering algorithm: application to the DREAM five-gene network challenge. Annals of the New York Academy of Sciences 1158: 102–13.

48. Yip KY, Alexander RP, Yan KK, Gerstein M (2010) Improved Reconstruction of In Silico Gene Regulatory Networks by Integrating Knockout and Perturbation Data. PLoS ONE 5: e8121.

Figure Legends

Tables

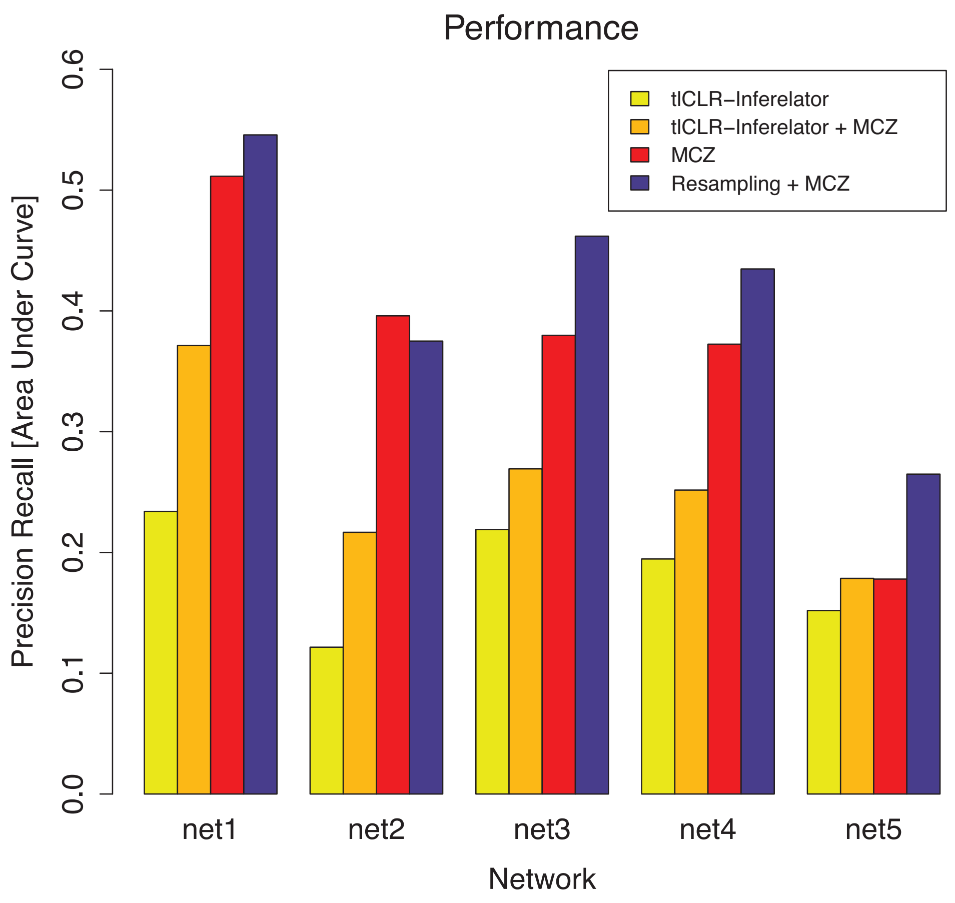

Figure 1. Area under precision recall curve for each ranking scheme We evaluated our performance in predicting the topology for each network using area under the precision recall curve (AUPR). We developed and tested four methods: 1) the TL-CLR → Inferelator 1.0 network inference pipeline — tlCLR-Inferelator, shown in yellow 2) combing the confidence scores of the inference pipeline with those derived from Z-scores based on the knockout data —tlCLR-Inferelator + MCZ, shown in orange 3) Z-scores on the genetic knockout data —MCZ, shown in red 4) resampling the network inference pipeline and combining with Z-scores—Resampling + MCZ, shown in purple. We see that Resampling + MCZ generally outpreforms all other methods.

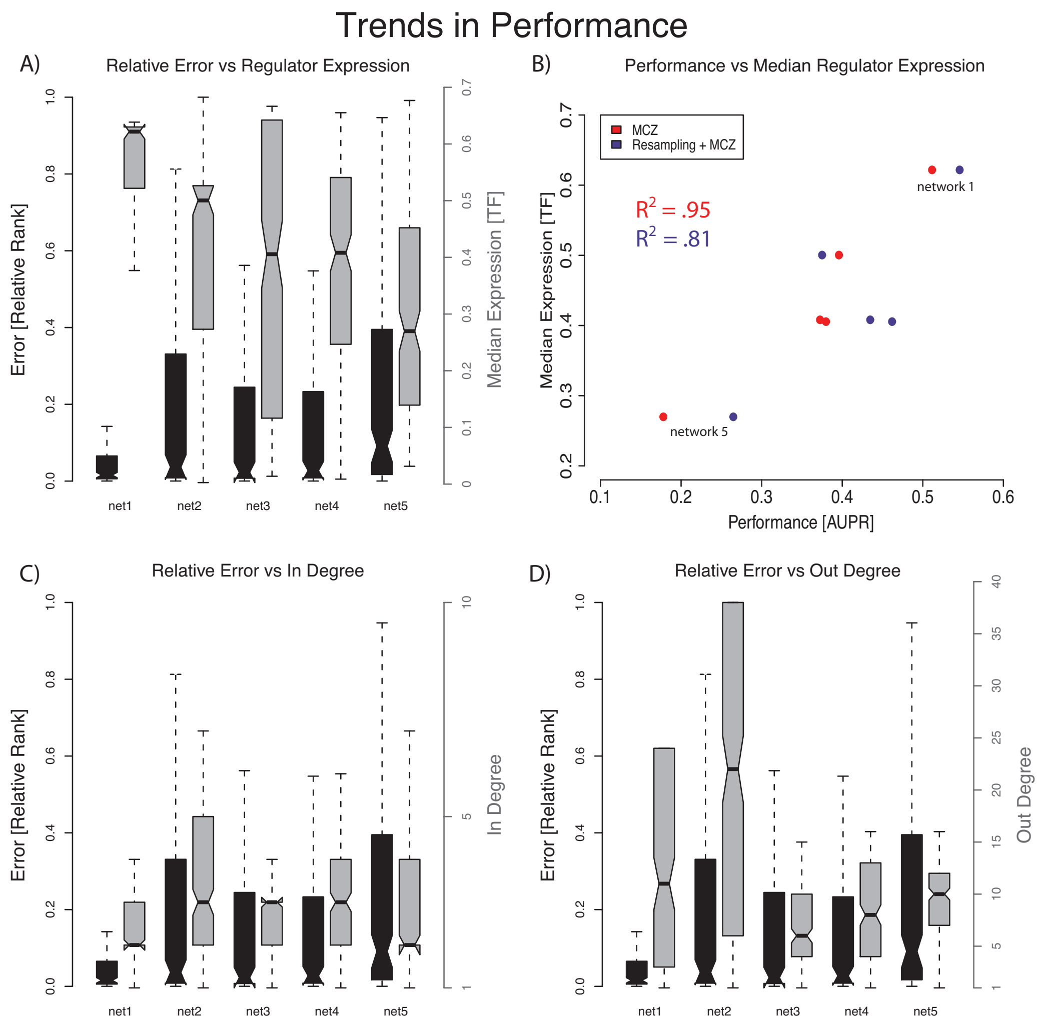

Figure 2. Trends in performance over the five networks For panels A,C,D we consider only the performance of MCZ, and use relative rank as an estimate of error. We compute relative rank in the following way. Denote by L the total number of possible regulatory interactions, and by l the rank that was given to each regulatory interaction, $x_j \rightarrow x_i$. The relative rank of $x_j \rightarrow x_i$ is defined to be $\frac{l}{L}$. Error distributions of the predictions for the five networks are shown as black boxplots in panels A,C,D. Distributions of median expression of the regulators, in-degree of the regulators, and out-degree of the regulators are shown as grey boxplots in panels A,C,D, respectively. **A**) Relative rank (Error) in network prediction increases as the median expression of the regulators decreases. **C**) There is no apparent relationship between relative rank (Error) and the in-degree of the regulators. **D**) there is no relationship between relative rank (Error) and out-degree of the regulators. **B**) we show the relationship between median expression of the regulators and the performance in predicting topology, in terms of AUPR, across all five networks. A correlation of ($R^2 = .95$) exists between MCZ-derived predictions and AUPR (shown in red), while there is a smaller correlation of ($R^2 = .81$) between resampling-based predictions (Resampling+MCZ) and AUPR (shown in purple).

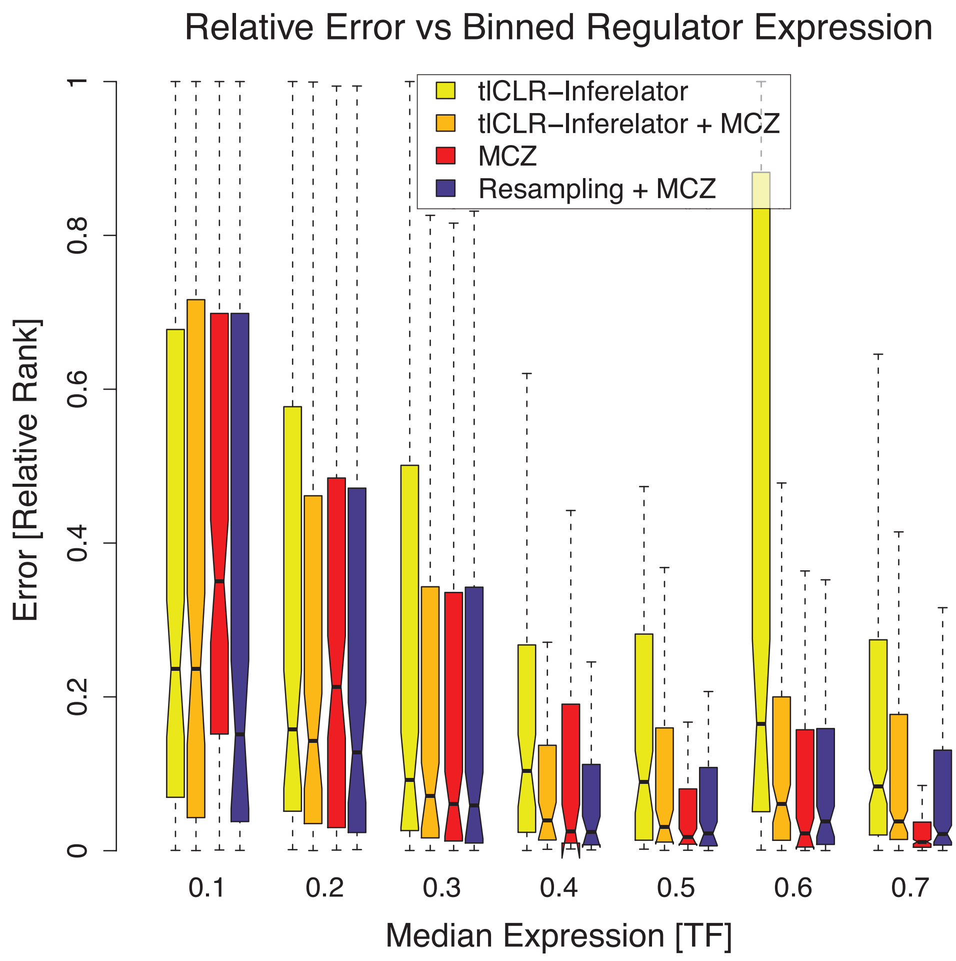

Figure 3. Error as a function of binned median expression for all regulatory interactions We further investigate the relationship between the median expression of the regulators and our performance in predicting topology. We use relative rank as an estimate of error (as in Figure 2). We bin the regulators for all five networks based on their median expression. We show the distribution of relative ranks (Error) for each method in each bin of regulator expression. We see that all of the methods that incorporate the predictions of the inference pipeline (tlCLR-Inferelator, tlCLR-Inferelator+MCZ, Resampling+MCZ) outperform MCZ for regulators with low median expression (0-.2).

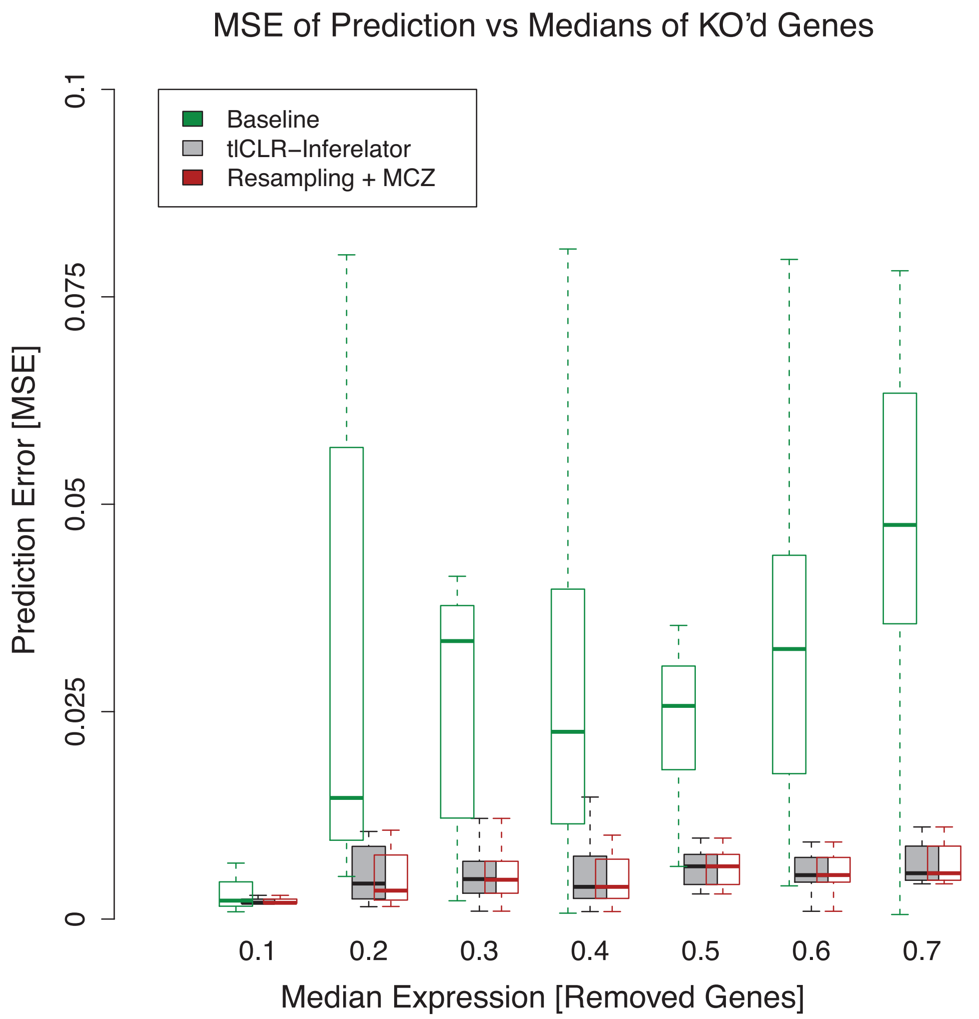

Figure 4. Performance on double knockout prediction We assess the accuracy of predicting the system's response to the simultaneous removal (knockout) of two genes x_i, x_j. In total, there were one-hundred pairs of genes that were knocked out. We bin these pairs of genes based on the average of their respective median expression in the single-gene knockout data. We compare the error of our prediction to the error made by using the wild-type expression, $x^{wt'}$, as a prediction (baseline, show in green). We evaluate the error of a prediction by mean squared error (MSE). We show the error distributions using parameters calculated by the inference pipeline (tlCLR-Inferelator, gray boxplot) are smaller than the error distributions of the baseline. Additionally, the error distributions using parameters calculated by the resampling approach (Resampling+MCZ, red boxplots) are essentially identical to the error distribution obtained by tlCLR-Inferelator derived parameters.

Table 1. Error in predicting double knockouts for each network

	baseline	x^0 =w.t.	tlCLR-Inferelator	Resampling+MCZ
Network 1	.04346	.03487	.00484	.00435
Network 2	.02909	.02570	.00757	.00696
Network 3	.01782	.01492	.00591	.00585
Network 4	.02064	.01970	.00562	.00564
Network 5	.04362	.04089	.00578	.00551

Here we present the error in double knockout predictions (i.e. predicting the response of the network to the removal of two genes). For each network twenty predictions were made. We present our average performance for each network (evaluated by mean squared error). Column 1 ("baseline") shows the average error if use the wild-type as the prediction. Column 2 ("x^0 =w.t.") shows the average error made by tlCLR-Inferelator derived parameters when the initial conditions are based on wild-type. Column 3 ("tlCLR-Inferelator") shows the average error made by tlCLR-Inferelator derived parameters when the initial condition vector is chosen from the knockout dataset (16). Column 4 ("Resampling+MCZ") shows the average error made by Resampling+MCZ (the parameters derived from the ensemble) using initial conditions from the knockout dataset.