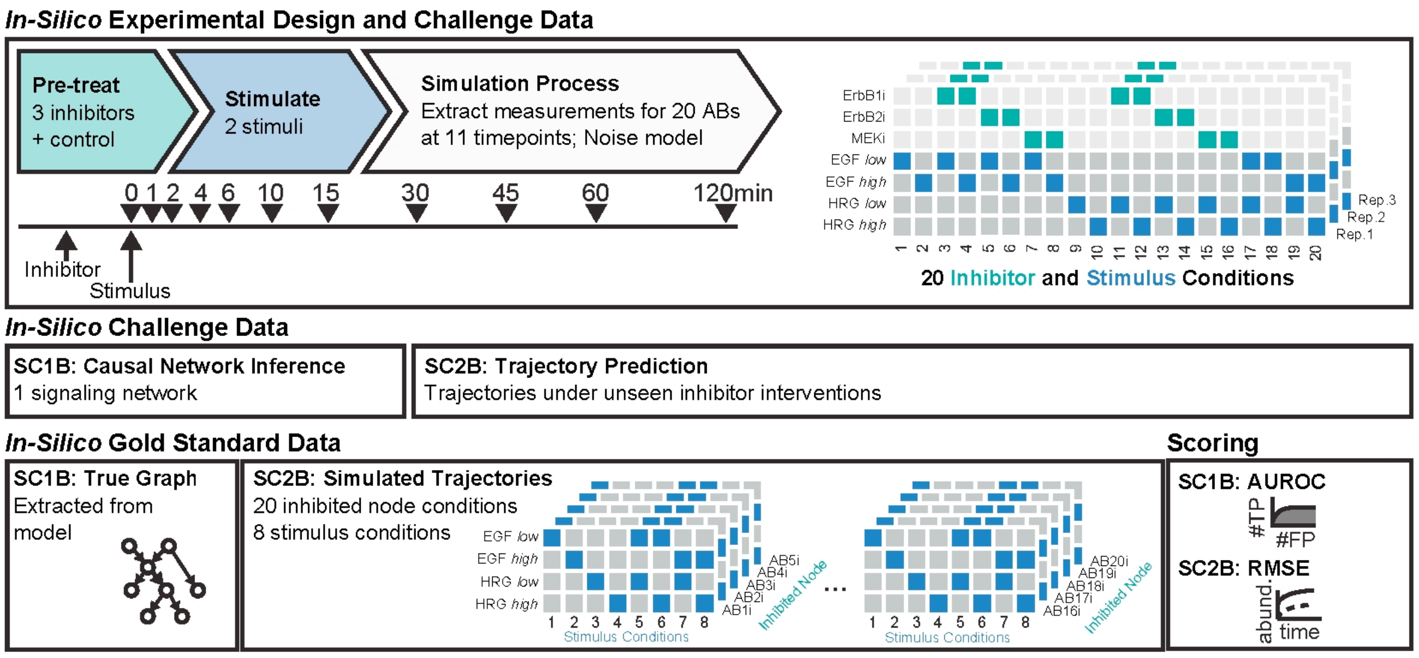
Nat. Methods; doi:10.1038/nmeth.3773 (22 February 2016)

Inferring causal molecular networks: empirical assessment through a community-based effort

Steven M Hill, Laura M Heiser, Thomas Cokelaer, Michael Unger, Nicole K Nesser, Daniel E Carlin, Yang Zhang, Artem Sokolov, Evan O Paull, Chris K Wong, Kiley Graim, Adrian Bivol, Haizhou Wang, Fan Zhu, Bahman Afsari, Ludmila V Danilova, Alexander V Favorov, Wai Shing Lee, Dane Taylor, Chenyue W Hu, Byron L Long, David P Noren, Alexander J Bisberg, The HPN-DREAM Consortium, Gordon B Mills, Joe W Gray, Michael Kellen, Thea Norman, Stephen Friend, Amina A Qutub, Elana J Fertig, Yuanfang Guan, Mingzhou Song, Joshua M Stuart, Paul T Spellman, Heinz Koeppl, Gustavo Stolovitzky, Julio Saez-Rodriguez & Sach Mukherjee

In the version of this supplementary file originally posted online, Supplementary Notes 1–11 were missing. These sections are now included as of 2 March 2016.

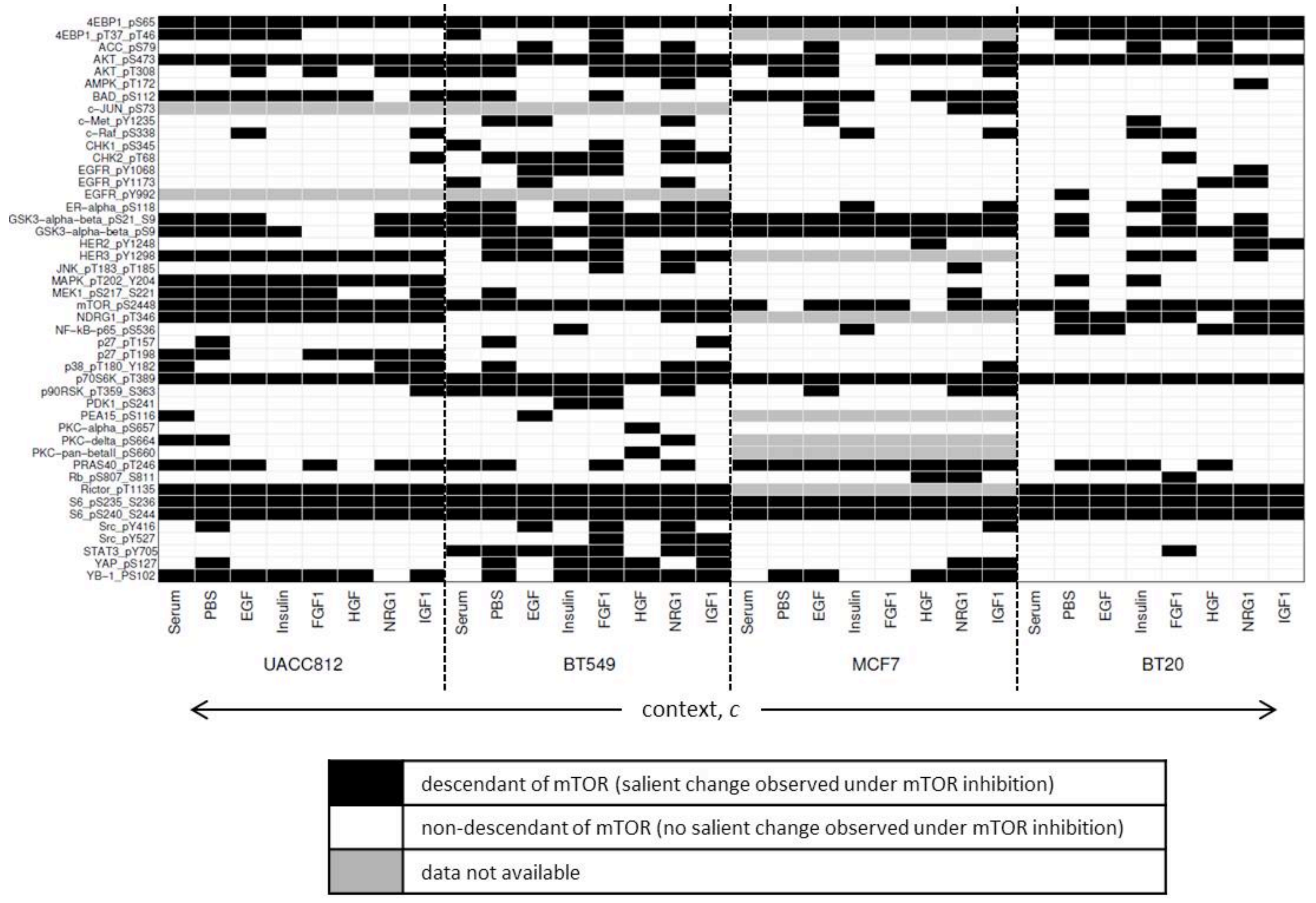


Supplementary Figure 1

The HPN-DREAM network inference challenge: overview of *in silico* data tasks.

Data were generated from a nonlinear dynamical model of the ErbB signaling pathway (Chen *et al*., 2009). Training data consisted of time-courses for 20 network nodes under three inhibitors targeting specific nodes, or no inhibitor, and under two ligand stimuli, applied individually and in combination at two concentrations. In total there were 20 different (inhibitor, stimulus) conditions as shown (top right). Time-courses comprised 11 time points and three technical replicates were provided. Node names were anonymized to prevent use of biological prior information. The sub-challenge 1 *in silico* data task (SC1B) asked participants to infer a single directed, weighted network using the training data. The aim of the sub-challenge 2 *in silico* data task (SC2B) was to predict stimulus-specific time-courses under unseen interventions. For SC1B, submissions were assessed against a gold-standard network extracted from the data-generating model, with agreement quantified using AUROC score. For SC2B, predicted time-courses were assessed using held-out test data obtained under *in silico* inhibition of each network node in turn, with prediction accuracy quantified using root mean square error (RMSE). See Online Methods for further details of the *in silico* data tasks.

Chen, W.W. *et al*. Input-output behavior of ErbB signaling pathways as revealed by a mass action model trained against dynamic data. *Mol. Syst. Biol.* **5**, 239 (2009).

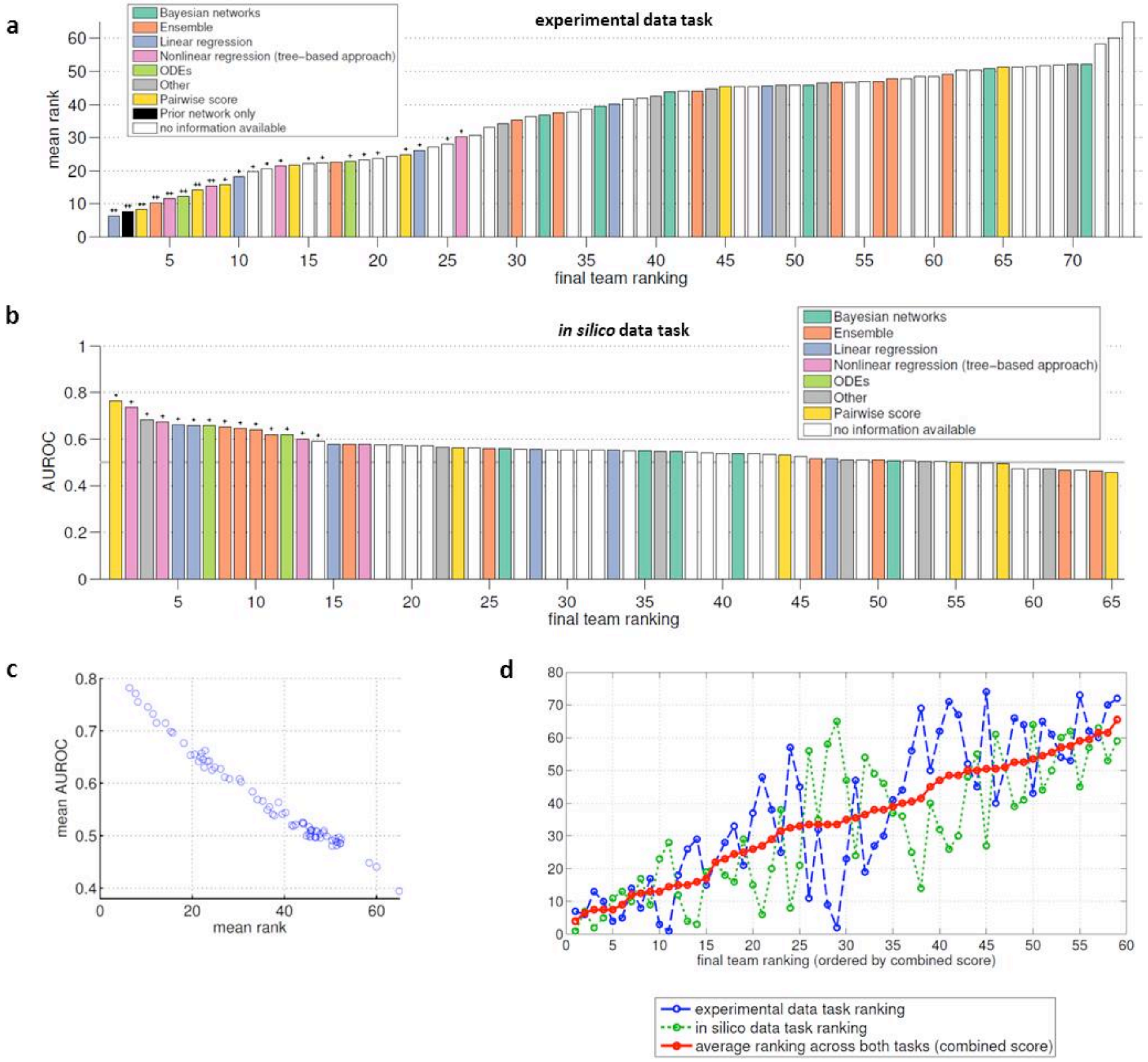


Supplementary Figure 2

Context-specific ‘gold-standard’ causal descendant sets for the network inference sub-challenge experimental data task (SC1A).

Context-specific networks submitted to SC1A were assessed using held-out test data, obtained under inhibition of mTOR. Each column in the heatmap indicates, for a given *(cell line, stimulus)* context *c*, the phosphoproteins that showed salient changes under mTOR inhibition relative to DMSO control (black cells) and those that did not (white cells). Such changes were determined from the test data using a procedure centered around a paired *t*-test. Phosphoproteins that show salient changes can be regarded as descendants of mTOR in the underlying causal signaling network. Columns therefore represent context-specific experimentally-determined sets of causal descendants of mTOR, D_c^{GS}, and were used as a ‘gold standard’ to assess inferred context-specific networks. Further details regarding the determination of the gold-standard descendant sets and the scoring procedure can be found in Online Methods. Missing data is indicated by gray cells (some phosphoprotein antibodies were only present in the (training and test) data for a subset of cell lines). Based on a figure in Hill, Nesser *et al*. (2016).

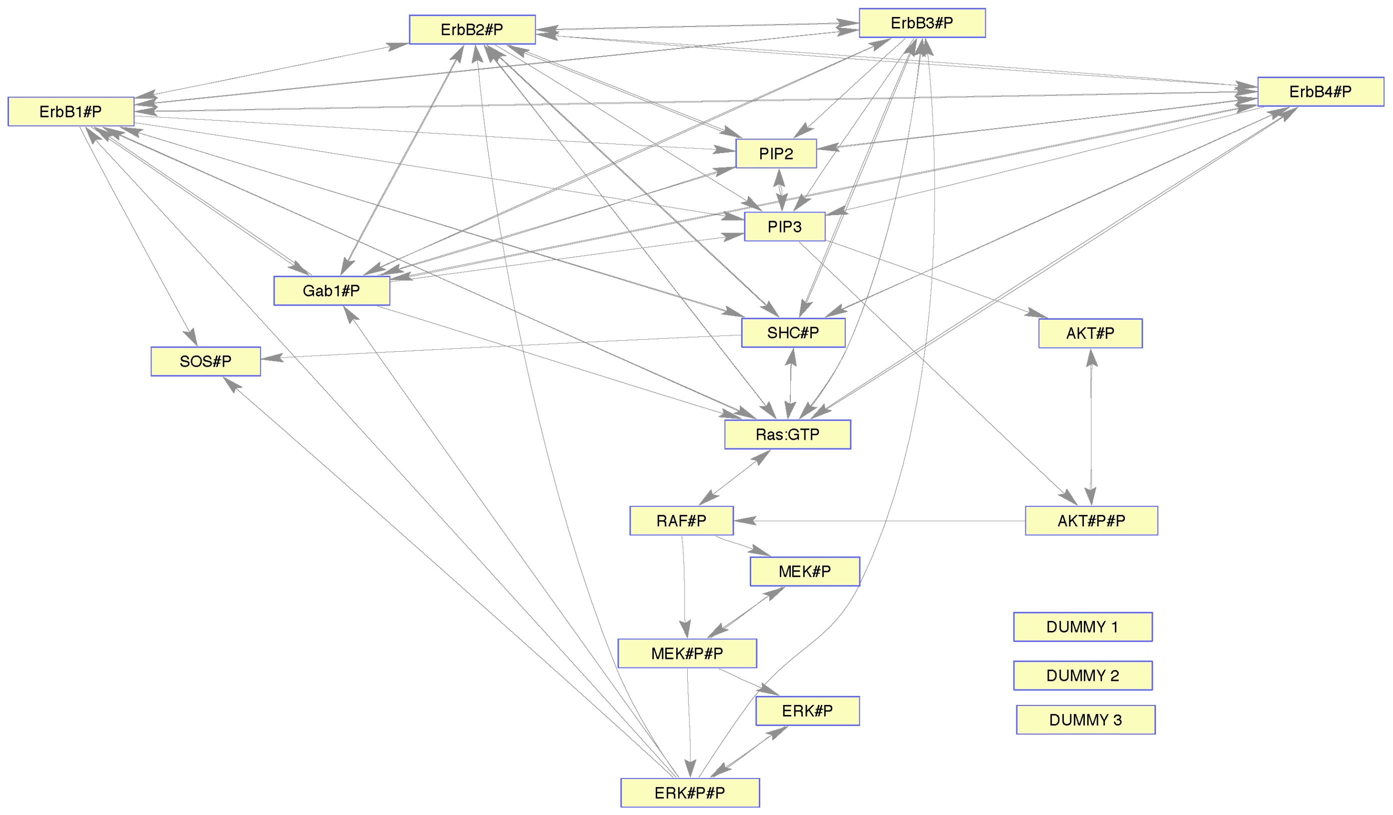
Hill, S.M., Nesser, N.K. *et al*. Context-specificity in causal signaling networks revealed by phosphoprotein profiling. *bioRxiv* doi:10.1101/039636 (2016).



Supplementary Figure 3

Network inference sub-challenge (SC1) final team scores and rankings.

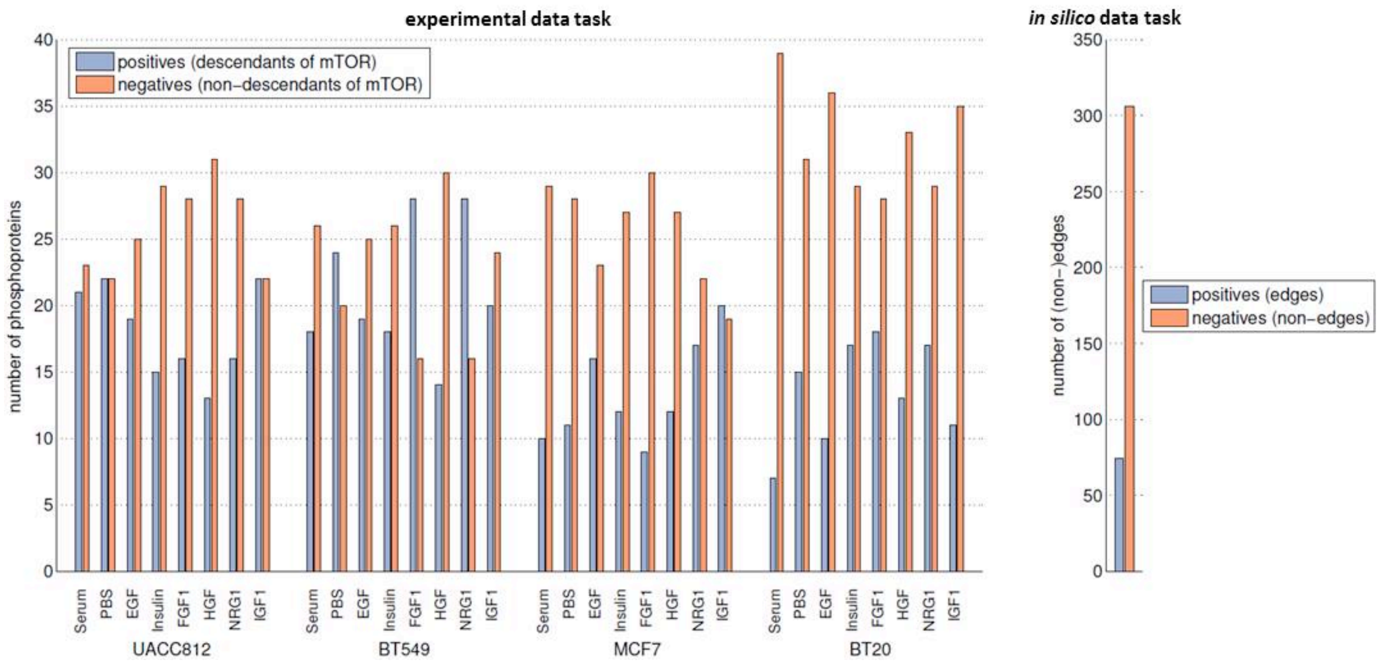
(**a**) Mean rank scores for the 74 teams that participated in the experimental data task (SC1A). Mean rank scores were used to obtain final team rankings. For the 40 teams that provided information regarding their approach, bar color indicates method type (see also **Fig. 3e**, **Supplementary Table 2** and **Supplementary Note 5**). Stars above bars indicate teams with statistically significant AUROC scores (FDR < 0.05) in at least 50% of *(cell line, stimulus)* contexts (2 stars) or at least 25% of contexts (1 star) (multiple testing correction performed within each context with respect to number of teams). (**b**) AUROC scores for the 65 teams that participated in the *in silico* data task (SC1B). AUROC scores were used to obtain final team rankings. As in **a**, color indicates method type (see also **Fig. 3f**, **Supplementary Table 2** and **Supplementary Note 5**). Stars above bars indicate statistically significant AUROC scores (FDR < 0.05). (**c**) Comparison of mean rank and mean AUROC scores for SC1A. (**d**) Final ranks for SC1A (dashed blue line) and SC1B (dotted green line) were averaged to obtain a combined score (solid red line) for the 59 teams that participated in both tasks. Teams ordered by combined score (see "SC1A/B combined final rank" column in **Supplementary Table 2**). See Online Methods for full details of scoring for SC1.



Supplementary Figure 4

Gold-standard causal network for the network inference sub-challenge *in silico* data task (SC1B).

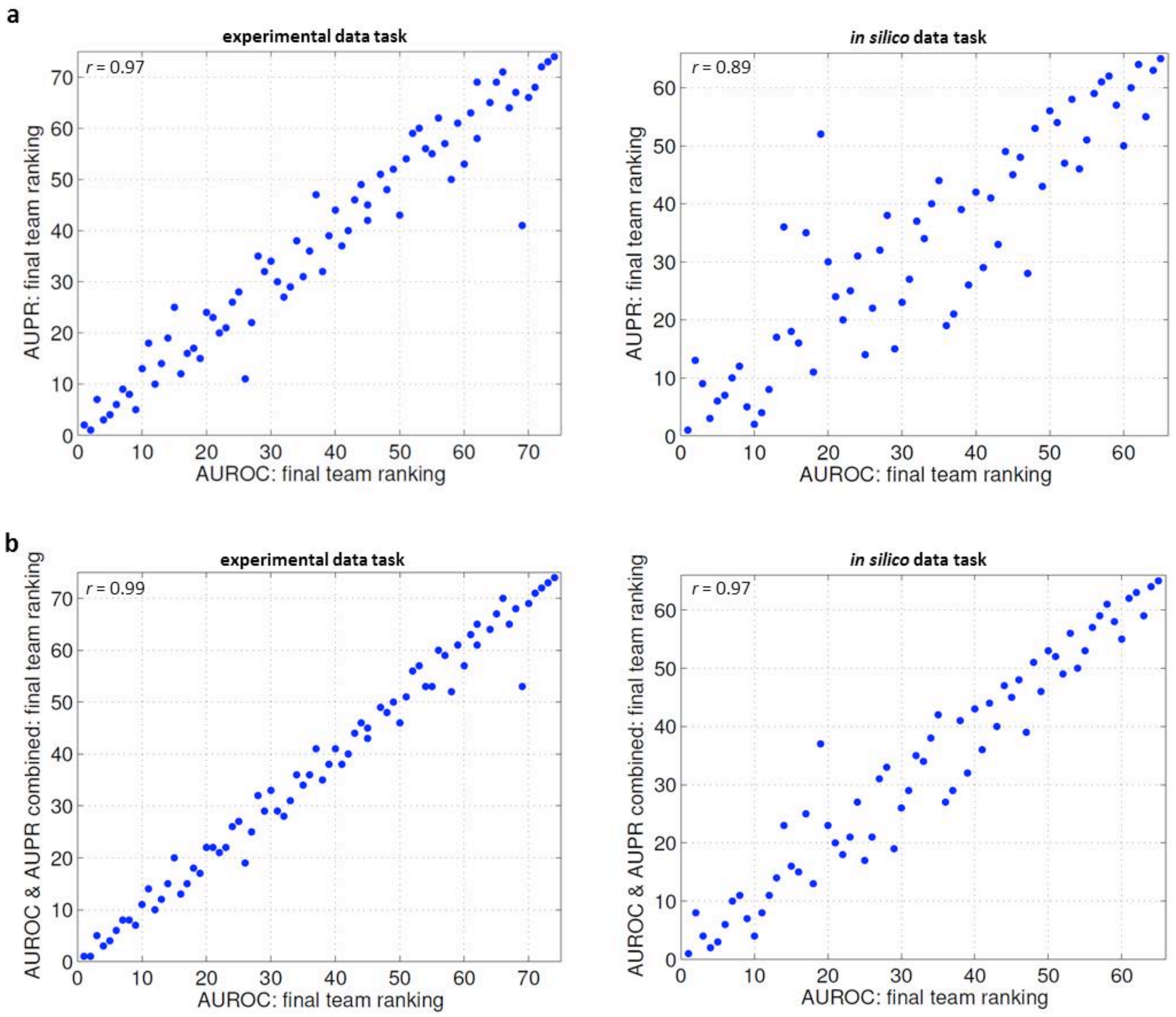
The gold-standard network, used to assess networks submitted to SC1B, was obtained from a data-generating dynamical model of the ErbB signaling pathway. Derivation of the network was non-trivial due to variables appearing in complexes within the model and full details can be found in **Supplementary Note 8**. Three unconnected dummy nodes were incorporated in the model and node names were anonymized in the training data.



Supplementary Figure 5

Balance of positives and negatives in the gold-standards for the network inference sub-challenge (SC1).

The gold standard for the experimental data task (SC1A) comprised sets of descendants of mTOR for each *(cell line, stimulus)* context, experimentally-determined using the held-out test data. Shown (left) are the number of positives and negatives for each context; that is, the number of phosphoproteins that are descendants of mTOR according to the test data (positives) and the number that are non-descendants of mTOR (negatives). For the *in silico* data task (SC1B), the gold-standard consisted of the data-generating network. Shown (right) are the number of edges in this network (positives) and the number of non-edges (negatives).

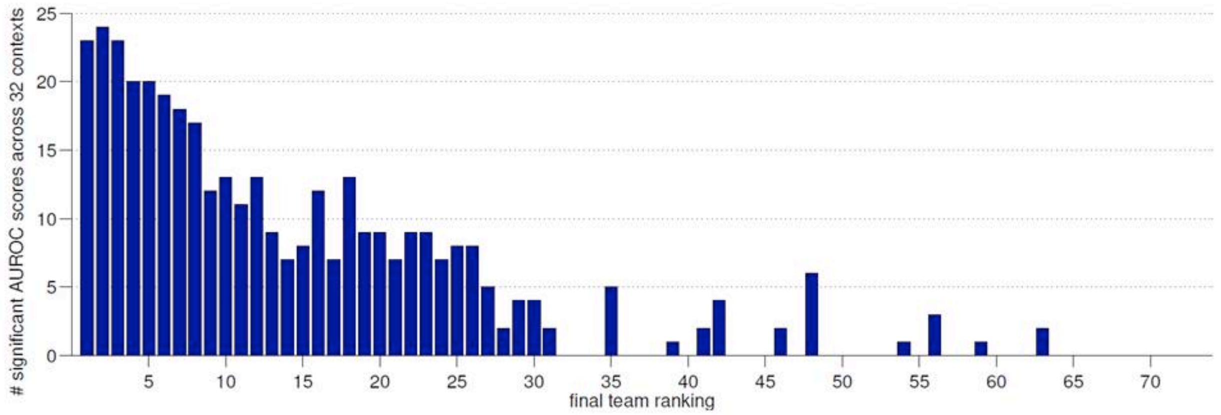


Supplementary Figure 6

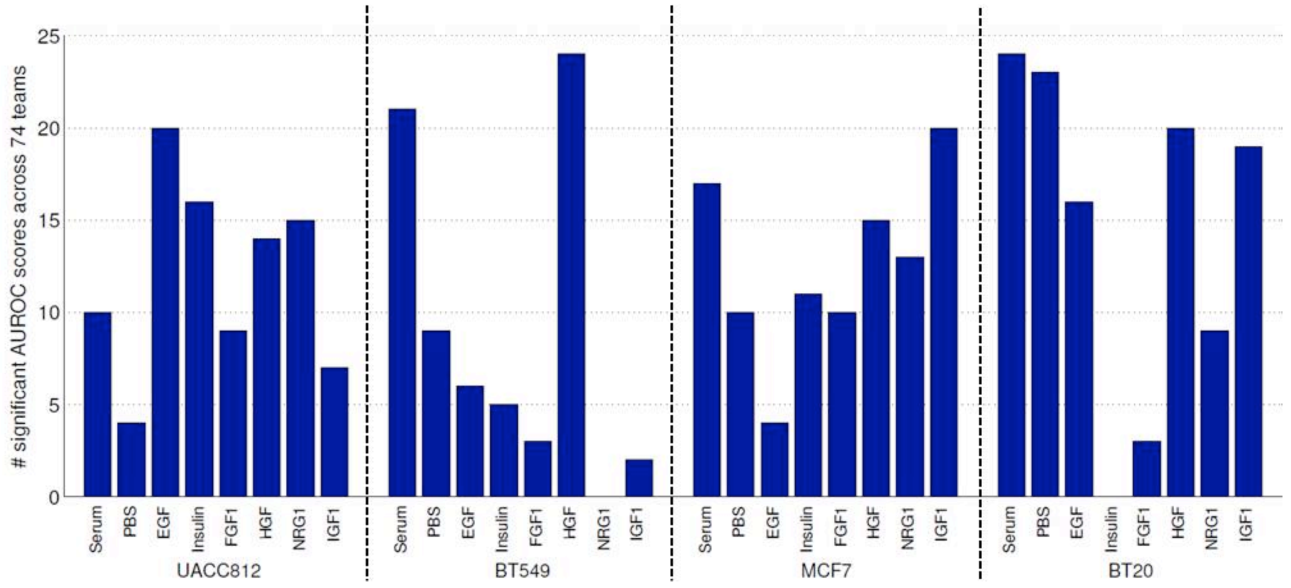
Comparison of AUROC with an alternative scoring metric, AUPR, for the network inference sub-challenge (SC1).

(**a**) Alternative team rankings were calculated by replacing AUROC with AUPR (area under the precision-recall curve) in the scoring procedure. The alternative rankings were compared with the original AUROC-based rankings for both the experimental data task (SC1A; left) and *in silico* data task (SC1B; right). (**b**) A further alternative ranking, combining both AUROC and AUPR, was obtained by ranking teams based on an average of final rank under AUROC and final rank under AUPR, and was compared with the original AUROC-based rankings.

a



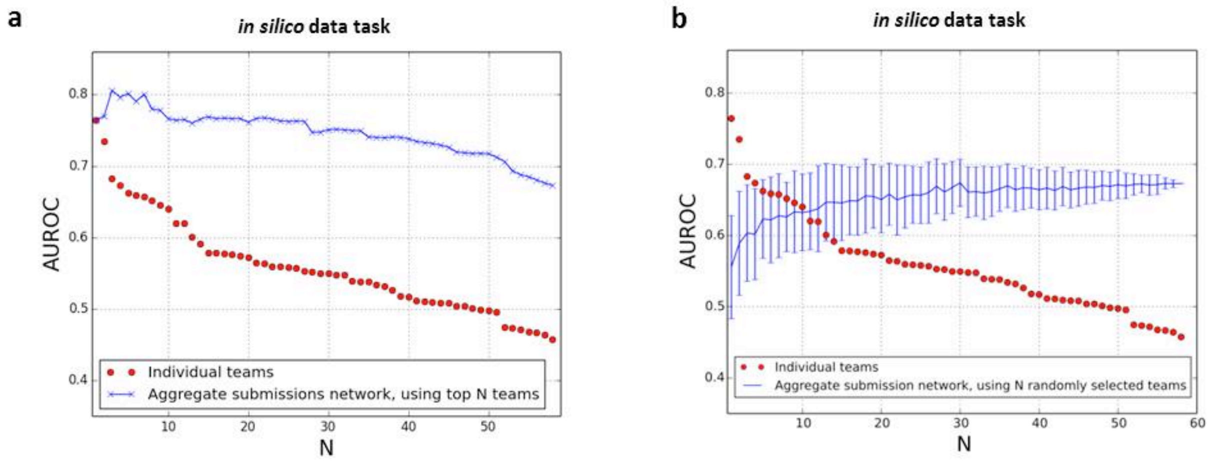
b



Supplementary Figure 7

Statistical significance of AUROC scores for the network inference sub-challenge experimental data task (SC1A).

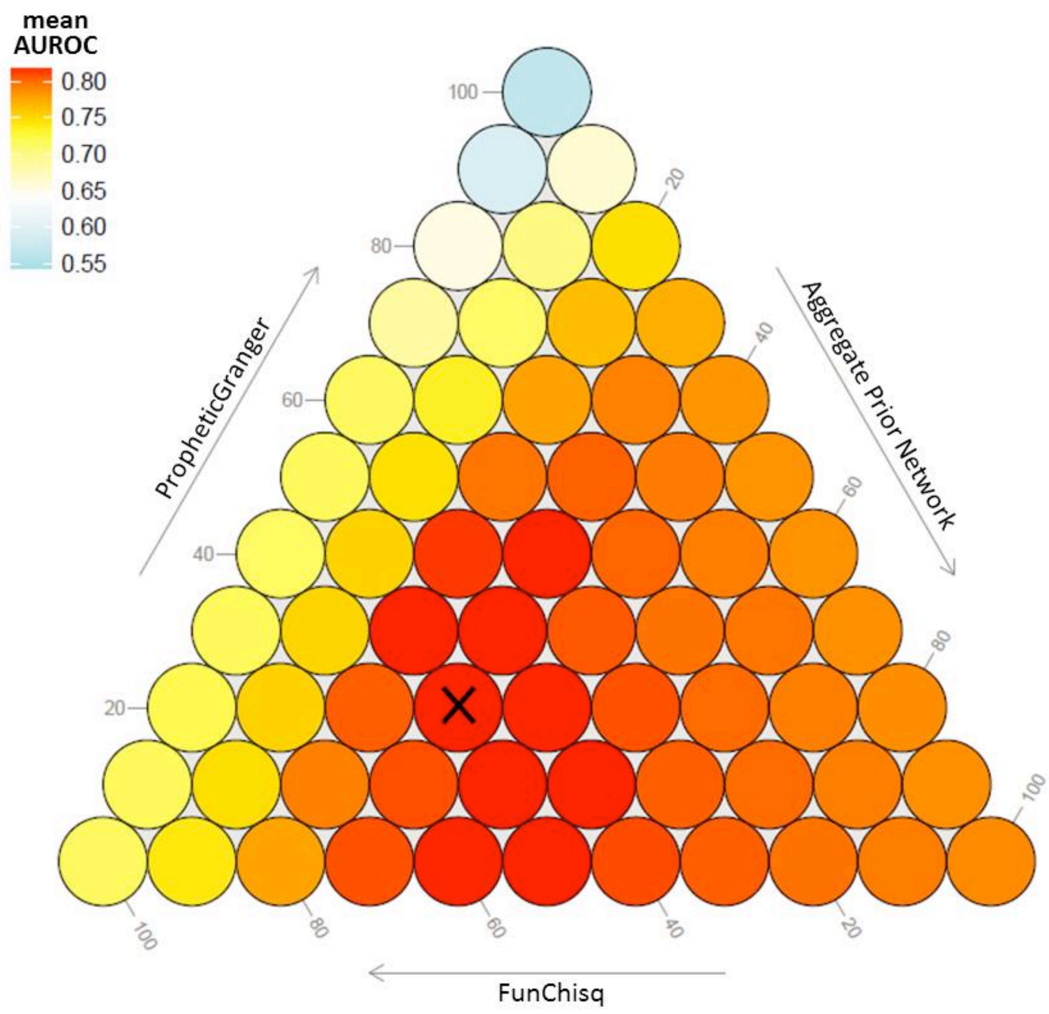
For each *(cell line, stimulus)* context, a null distribution over AUROC was generated and used to calculate an FDR-adjusted *P* value for each team (Online Methods). (**a**) The number of significant (FDR < 0.05) AUROC scores obtained by each team across the 32 contexts (multiple testing correction performed within each context with respect to number of teams). Teams are ordered according to their final ranking in SC1A (based on mean rank score). (**b**) For each context, the number of teams (out of a total of 74) that obtained significant AUROC scores. For two regimes *(BT549, NRG1)* and *(BT20, Insulin)*, no teams obtained a significant AUROC score. These two regimes were disregarded in the scoring process.



Supplementary Figure 8

Crowdsourced analysis for the network inference sub-challenge *in silico* data task (SC1B).

(**a**) Aggregate submission networks were formed by integrating predicted networks across the top *N* teams (as given by final team rankings), with *N* varied between 1 (top performer only) and all teams (after removal of correlated submissions; **Supplementary Note 10**). Integration was done by averaging predicted edge weights (Online Methods). The blue line shows performance (AUROC) of the aggregate submission networks. Individual team scores are also depicted (red circles). (**b**) Predicted networks were integrated for subsets of *N* teams, selected at random. The blue line shows mean performance of the aggregate submission networks, calculated over 100 random subsets of teams (error bars indicate s.d.). Crowdsourced analysis for the experimental data network inference task is shown in **Figure 3c,d**.

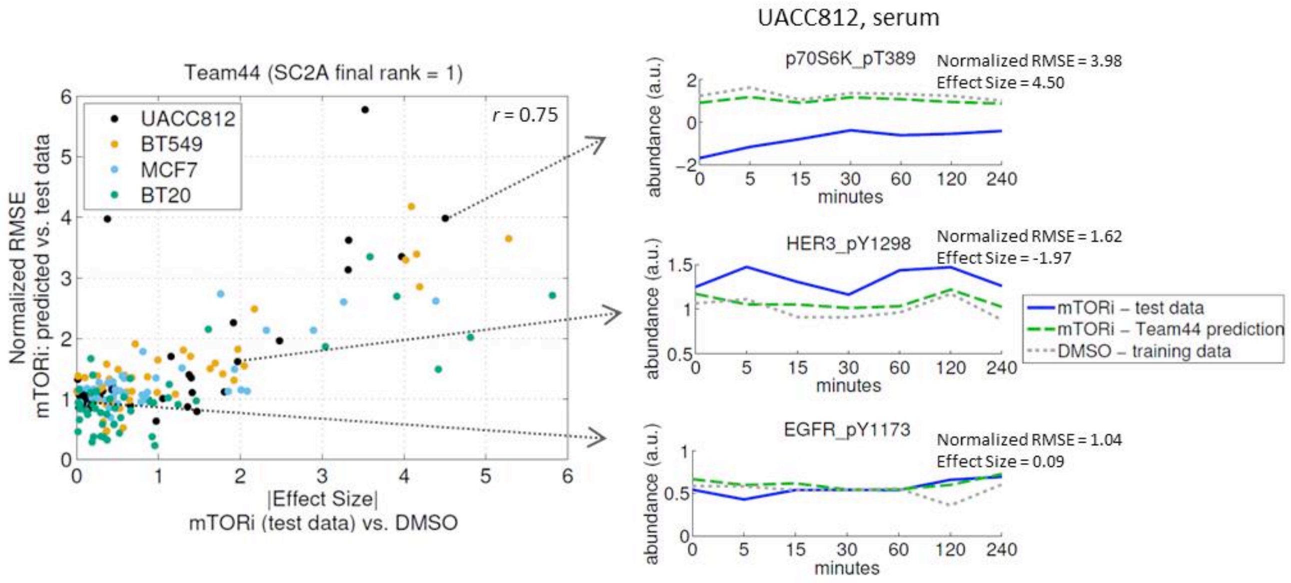


Supplementary Figure 9

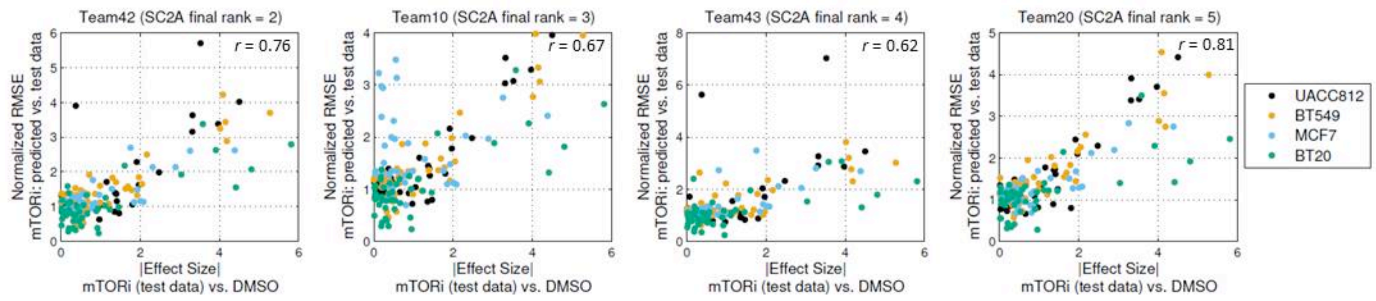
Weighted combinations of two top performing approaches and aggregate prior network for the network inference sub-challenge experimental data task (SC1A).

An extension of **Figure 4b** to show three-way combinations of (i) PropheticGranger – top performer for the experimental data task when combined with a prior network (here, the method is used without the prior network); (ii) FunChisq – top performer for the *in silico* data task and most consistent performer across both data types; and (iii) an aggregate prior network formed by integrating prior networks used by participants (Online Methods). The three approaches were combined by taking weighted averages of predicted edge scores for each *(cell line, stimulus)* context and performance assessed using mean AUROC. For example, the best performance (mean AUROC = 0.82) was achieved by combining 20% PropheticGranger, 50% FunChisq and 30% aggregate prior network, and is highlighted with an "X". See **Supplementary Note 1** for full details of the PropheticGranger and FunChisq approaches.

a



b

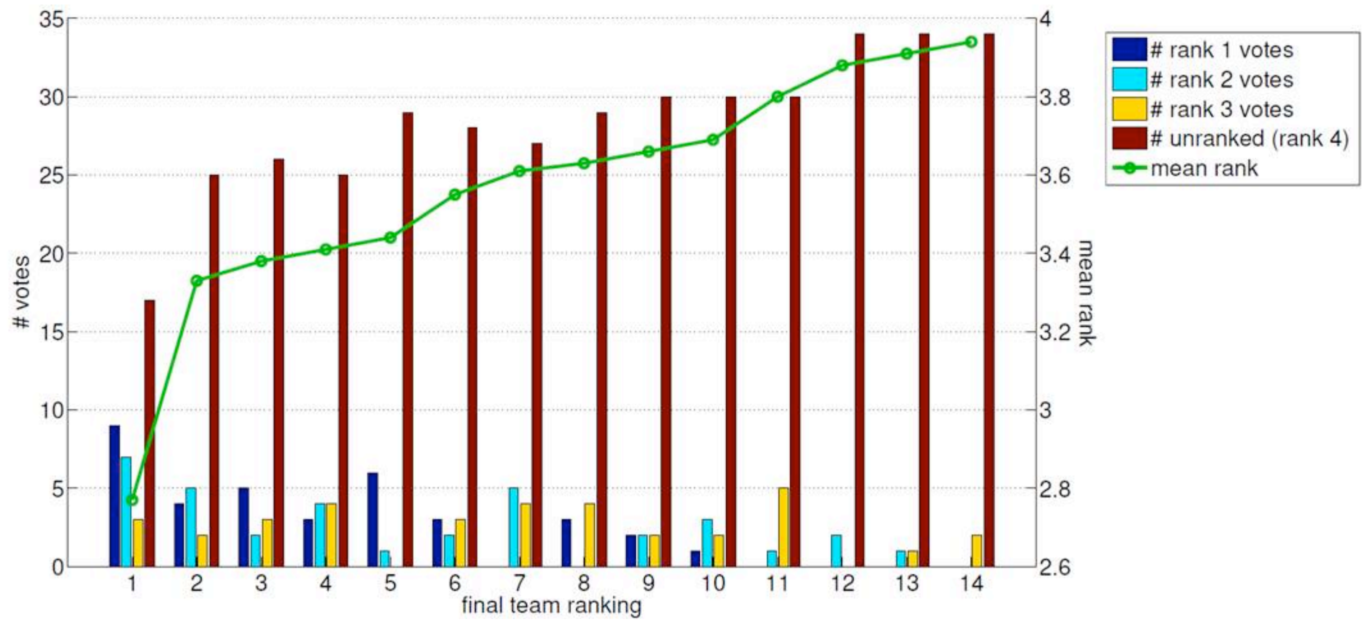


Supplementary Figure 10

Time-course prediction sub-challenge experimental data task (SC2A): phosphoproteins showing the largest changes under mTOR inhbition are predicted with least accuracy.

SC2A tasked participants with predicting phosphoprotein time-courses for each *(cell line, stimulus)* context under an unseen intervention (mTOR inhibition - mTORi). Submitted predictions were assessed against held-out test data obtained under mTORi. For each team, root mean square error (RMSE) scores were calculated for each *(cell line, phosphoprotein)* pair (see **Supplementary Note 6**). (**a**) Left: for each *(cell line, phosphoprotein)* pair, normalized RMSE[1] for the top-ranked team (Team44) vs. absolute effect size. The effect size for a given *(cell line, phosphoprotein)* pair is a measure of the magnitude of abundance change under mTORi relative to DMSO control[2]. Note that this measure is based on the mTORi test data and is independent of team predictions. The strong positive correlation indicates that phosphoproteins showing little or no change under mTORi were predicted relatively well but phosphoproteins that showed large changes under mTORi were predicted badly. Right: examples of time-courses underlying the scatter plot (left). Shown are abundances of three phosphoproteins for cell line UACC812 under DMSO control and under mTORi, as predicted by Team44 and test data values. Note that normalized RMSE and effect size values are calculated across all stimuli, but only serum stimulus time-courses are shown here. (**b**) Scatter plots as in **a** for teams ranked 2 to 5 in SC2A. These results highlight the challenging nature of predicting protein abundance under unseen interventions but also point to a shortcoming of the RMSE score used here, namely that it does not sufficiently emphasize ability to predict proteins that change under intervention. For a future challenge, a modified metric that focuses on those proteins might therefore be useful.

[1]To ensure comparability across cell lines and phosphoproteins, each RMSE score was normalized by the standard deviation of the test data used in the RMSE calculation.
[2]Effect size is defined as the mean difference in phosphoprotein abundance between DMSO control and mTORi, normalized by the standard deviation of the differences. Means and standard deviations are calculated across all time points and stimuli for the given cell line.

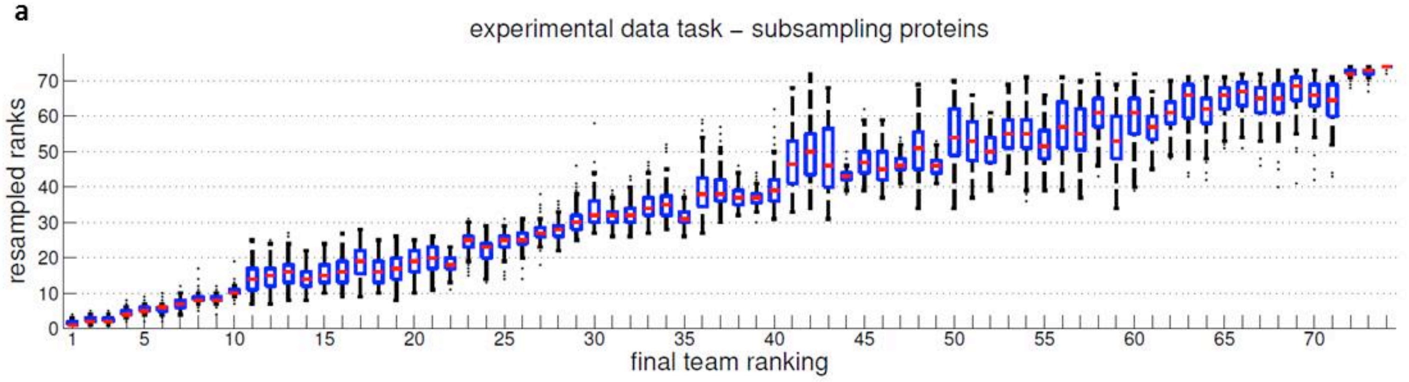


Supplementary Figure 11

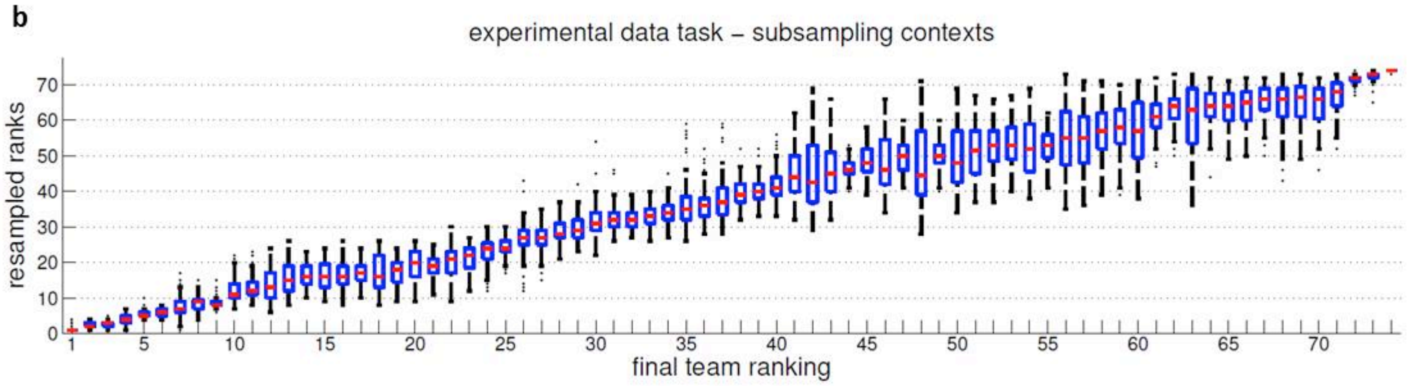
Visualization sub-challenge (SC3) voting results and rankings.

14 teams made submissions to the visualization sub-challenge. HPN-DREAM challenge participants were asked to select and rank (from 1 to 3) their three favorite submissions. The remaining unranked submissions were then assigned a rank of 4. Thirty-six participants participated in the voting process and the number of votes of each rank type is shown (bar plot, left axis). Final team ranks were based on the mean rank across the 36 votes (green line, right axis).

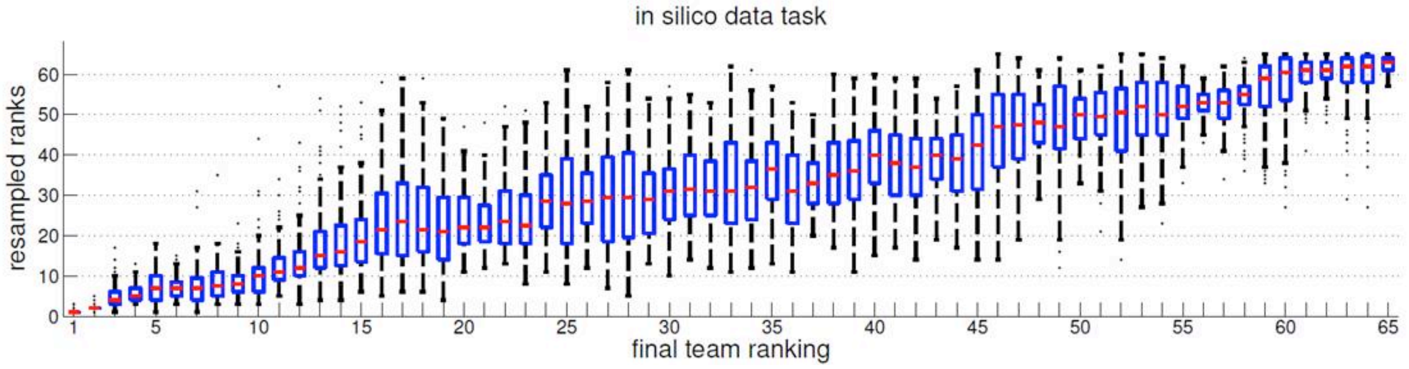
a



b



c



Supplementary Figure 12

Robustness of rankings for the network inference sub-challenge (SC1).

The test data was subsampled to assess robustness of rankings (Online Methods). Box plots show team ranks over 100 subsampling iterations, with 50% of the test data left out at each iteration. (**a**) Experimental data task - subsampling performed by removing 50% of phosphoproteins when assessing descendant sets for each *(cell line, stimulus)* context. (**b**) Experimental data task – subsampling performed by removing 50% of contexts from the scoring process. (**c**) *In silico* data task – subsampling performed by considering only 50% of edges/non-edges in the gold-standard network. For all box plots, the central line indicates the median, and the box edges denote the 25th and 75th percentiles. Whiskers extend to 1.5 times the interquartile range from the box hinge. Data points beyond the whiskers are regarded as outliers and are plotted individually.

Supplementary Information

Inferring causal molecular networks: empirical assessment through a community-based effort

Steven M. Hill[1,*], Laura M. Heiser[2,3,4,*], Thomas Cokelaer[5], Michael Unger[6], Nicole K. Nesser[7], Daniel E. Carlin[8], Yang Zhang[9,25], Artem Sokolov[8], Evan O. Paull[8], Chris K. Wong[8], Kiley Graim[8], Adrian Bivol[8], Haizhou Wang[9,25], Fan Zhu[10], Bahman Afsari[11], Ludmila V. Danilova[11,12], Alexander V. Favorov[11,12,13], Wai Shing Lee[11], Dane Taylor[14,15], Chenyue W. Hu[16], Byron L. Long[16], David P. Noren[16], Alexander J. Bisberg[16], HPN-DREAM Consortium, Gordon B. Mills[17], Joe W. Gray[2,3,4], Michael Kellen[18], Thea Norman[18], Stephen Friend[18], Amina A. Qutub[16], Elana J. Fertig[11], Yuanfang Guan[10,19,20], Mingzhou Song[9], Joshua M. Stuart[8], Paul T. Spellman[7], Heinz Koeppl[6,25], Gustavo Stolovitzky[21,^], Julio Saez-Rodriguez[5,22,^], Sach Mukherjee[1,23,24,25,^]

1. MRC Biostatistics Unit, Cambridge Institute of Public Health, Cambridge, UK
2. Department of Biomedical Engineering, Oregon Health and Science University, Portland, Oregon, USA
3. Center for Spatial Systems Biomedicine, Oregon Health and Science University, Portland, Oregon, USA
4. Knight Cancer Institute, Oregon Health and Science University, Portland, Oregon, USA
5. European Molecular Biology Laboratory, European Bioinformatics Institute (EMBL-EBI), Wellcome Trust Genome Campus, Hinxton, UK
6. Automatic Control Laboratory and Institute of Biochemistry, ETH Zurich, Zurich, Switzerland
7. Department of Molecular and Medical Genetics, Oregon Health and Science University, Portland, Oregon, USA
8. Biomolecular Engineering, UC Santa Cruz, Santa Cruz, California, USA
9. Department of Computer Science, New Mexico State University, Las Cruces, New Mexico, USA
10. Department of Computational Medicine and Bioinformatics, University of Michigan, Ann Arbor, Michigan, USA
11. Department of Oncology, Division of Biostatistics and Bioinformatics, Sidney Kimmel Comprehensive Cancer Center, Johns Hopkins University, Baltimore, Maryland, USA
12. Laboratory of Systems Biology and Computational Genetics, Vavilov Institute of General Genetics, Russian Academy of Sciences, Moscow, Russia
13. Laboratory of Bioinformatics, Research Institute of Genetics and Selection of Industrial Microorganisms, Moscow, Russia
14. Statistical and Applied Mathematical Sciences Institute, Research Triangle Park, North Carolina, USA
15. Department of Mathematics, University of North Carolina, Chapel Hill, North Carolina, USA
16. Department of Bioengineering, Rice University, Houston, Texas, USA
17. Department of Systems Biology, MD Anderson Cancer Center, Houston, Texas, USA
18. Sage Bionetworks, Seattle, Washington, USA
19. Department of Internal Medicine, University of Michigan, Ann Arbor, Michigan, USA
20. Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, Michigan, USA
21. IBM Translational Systems Biology and Nanobiotechnology, Yorktown Heights, New York, USA
22. RWTH-Aachen University Hospital, Joint Research Centre for Computational Biomedicine (JRC-COMBINE), Aachen, Germany
23. School of Clinical Medicine, University of Cambridge, Cambridge, UK
24. German Centre for Neurodegenerative Diseases (DZNE), Bonn, Germany

25. Present addresses: SimQuest Inc, Boston, Massachusetts, USA (H.W.); Amyris Inc, Emeryville, California, USA (Y.Z.); Department of Electrical Engineering and Information Technology, Technische Universitaet Darmstadt, Darmstadt, Germany (H.K.); German Centre for Neurodegenerative Diseases, Bonn, Germany (S.M.).

* equal contributions

^ corresponding

Further supplementary material, including all training and blinded test data, participant submissions and scores, participant prior networks, participant code, crowdsourced aggregate networks, and code for scoring submissions can be accessed through Synapse at https://www.synapse.org/HPN_DREAM_Network_Challenge, under the section "HPN-DREAM Community Resource".

Table of Contents

Supplementary Note 1: HPN-DREAM Network Inference Sub-challenge (SC1) Methods

Team submission files, code and prior networks (where available; see Supplementary Table 2) can be found on Synapse at https://www.synapse.org/HPN_DREAM_Network_Challenge, under the section "HPN-DREAM Community Resource".

SC1 Network Inference: Team1 – PropheticGranger with heat diffusion prior

Summary

"PropheticGranger": our method is an extension of L1-penalized Granger causality constructed specifically to consider "future data", combined with a prior derived from known biological pathways (this prior knowledge network was also submitted as a stand-alone prediction; see Team2 method).

Introduction

Our approach has two aspects: the prior based on curated, publicly available biological pathways that made no use of the HPN DREAM8 data, and the computational approach (Prophetic Granger Causality) that did use the HPN DREAM8 data. For the *in silico* sub-challenge, only the computational part of the method was used.

Motivating the use of the prior is the observation that methods have an improved ability when they make use external data, as was demonstrated particularly well in the DREAM7 Breast Cancer Prognosis Challenge (Margolin, et al., 2013; Cheng, et al., 2013). We chose Pathway Commons (Cerami et al., 2011) as our external source of data due to its comprehensive coverage of the biological pathways implicated in breast cancer. We reasoned that proteins are more likely to be perturbed by the inhibition of other proteins nearby in the graph. To characterize the notion of network distance, we applied heat diffusion, which has been shown to produce biologically meaningful metrics that take into account multiple paths between graph nodes (Qi et al., 2008).

We developed a method based on L1-penalized Granger causality to infer protein-protein regulatory relationships from time series data like the kind provided by the HPN challenge. Granger causality is a well-established method for extracting causal information from time series data in general, and has found use in systems biology (Shojaie and Michailidis, 2010). The L1-penalized version is especially relevant because it can account for irregular time series where the data is spaced at increasing intervals (Bahadori and Liu 2012) such as the HPN DREAM8 data. Granger causality works by regressing onto the "past data" of a particular focus protein and finds any exogenous variables that account for variance beyond what is accounted for by the autoregression terms of the protein itself. Proteins represented among the exogenous variables that provide explanatory power beyond the autoregressive terms are assumed to contribute in a causal way to the response of the focus protein.

We extended the L1-penalized Granger algorithm to also consider future time points, a method we refer to as “prophetic Granger causality” (PGC). PGC uses all available data (i.e. both past and future) for each regression task, and declares the direction of causation based on the temporal ordering between variables. To identify variables that provide explanatory power beyond autoregression, we set the L1 penalty coefficient to the smallest value that produced an assignment of zero weight to all autoregressive terms. Any remaining terms with non-zero weights among the exogenous variables were then predicted as having a causal relationship to the response variable.

Methods

A prior network was constructed from Pathway Commons using the method described in detail in the Team2 write-up. In brief, the relevant causal links were extracted via a simulated heat diffusion process over the Pathway Commons interaction network.

The only preprocessing performed was to address missing values and data replicates: missing data was omitted from the PGC regression formulations and the median across replicates was used to summarize each probe.

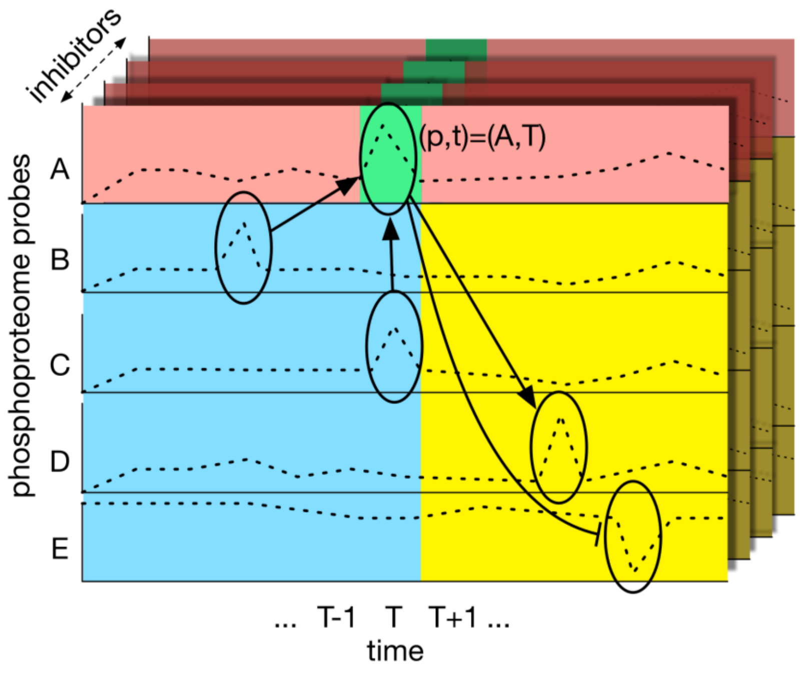


Figure 1. Illustration of the Prophetic Granger causality method. The level of a target phosphoprotein at each time point (green) is considered as a linear regression of all other time points and phosphoproteins. The L1 penalty provides a sparse solution, and is chosen such that the target phosphoprotein’s own past and future contributions (red) are zero, accounting for autoregression. Remaining non-zero regression coefficients for other phosphoproteins suggest causality. Phosphoproteins associated with the past or concurrent time points (blue) are interpreted as causally related to the target phosphoprotein. Conversely, the direction of causality is reversed for phosphoproteins associated with the future time points (yellow). The different inhibitor conditions are treated as different examples of the regression task. This process was repeated for each time point and for each phosphoprotein acting as a target. The resulting regression solutions were then combined into a single connectivity matrix

Prophetic Granger Causality takes as input a dataset with five attributes: time *t*, cell line *l*, condition *c*, inhibitor *i*, and phosphoprotein *p*. An independent network is derived for each (cell line, stimulus)-pair, encoded as a directed weighted *p* x *p* connectivity matrix $C_{l,c}$ with real values on [-1,1], where a value of -1 is the most inhibitory and 1 is the most excitatory connection influence.

For a given stimulus-cell line pair, let *D* be the 3D data cube indexed by the triplet *(i,t,p)*. We formulate a regression task for each time-probe pair *(t,p)*, treating each inhibitory intervention as a separate example in that task. Specifically, for a given inhibitor *i*, our formulation subdivides *D* into three parts: the response $y_i = D_{i,t,p}$; an autoregression explanatory vector A_i made up of $D_{i,t',p}$, for all *t'* not equal to t*;* and an exogenous explanatory matrix X_i made up of $D_{i,*,p'}$, for all probes *p'* not equal to *p* and all time points (see Figure 1).

Following standard regression, we seek to express *y* as a linear combination of *A* and *X*:

$$\hat{y}_i = \alpha^T A_i + \beta^T X_i + \beta_0 = \sum_j \alpha_j (A_i)_j + \sum_k \beta_k (X_i)_k + \beta_0,$$

where we concatenated the columns of *X* into a single vector, for notational convenience.

We solve this regression problem by formulating an L1-regularized least-squares fit:

$$\min_{\alpha,\beta} \sum_i (y_i - \hat{y}_i)^2 + \lambda \left(\sum_j |\alpha_j| + \sum_k |\beta_k| \right)$$

This regression problem is solved by gradient descent (Friedman et al., 2010). The value of the meta-parameter $\boldsymbol{\lambda}$ is chosen such that all of the autoregression terms, *α*, are zero. The smallest such value is given by:

$$\lambda = \max_j \left| \frac{1}{n} \sum_{i=1}^{n} (A_i)_j \left(y_i - y_i^{(j)} \right) \right|$$

After training a model with the above $\boldsymbol{\lambda}$ value, any non-zero weight *β* in the model for (p,t) is interpreted as a causal relationship between the response phosphoprotein *p* at time *t* and the explanatory phosphoprotein *p*' at time *t*' that corresponds to that β. In the absence of such weights (i.e., when all weights in the model are zero), the method concludes that there is no evidence for causality based on the given regression task.

The final connectivity matrix *C* is constructed by normalizing and summing all non-zero coefficients from all regression tasks. To normalize, we divided each weight by the sum of the absolute values of all weights in the same regression task, effectively giving all tasks the same voting power towards the final connectivity matrix. The directionality of each weight's contribution toward *C* was established by the temporal ordering between the response phosphoprotein and the corresponding explanatory phosphoprotein. Specifically, let $C_{p,p'}$ denote the belief that phosphoprotein *p'* causes a response in phosphoprotein *p*. For the regression task on target phosphoprotein *p* at time *t*, if there is a non-zero *β* term corresponding to explanatory phosphoprotein *p'* at time *t'*, we update *C* with the following rule:

$$C_{p',p} \leftarrow C_{p',p} + \frac{\beta_j}{\sum_k |\beta_k|} \text{ if } t \geq t'$$

$$C_{p,p'} \leftarrow C_{p,p'} + \frac{\beta_j}{\sum_k |\beta_k|} \text{ otherwise}$$

For the purpose of the HPN DREAM8 challenge, we reported the absolute value of the final connectivity matrix since the evaluation of challenge submissions made no distinction between inhibitor and promoter links. For the *in silico* part of the contest, *C* was submitted as the result. In the experimental portion, the submission was combined with the biological prior by a simple arithmetic mean, after both matrices were scaled to [0,1] via a division by the largest entry.

Discussion
Note that the heat kernel does not use the training data in its calculation. Nonetheless, this matrix by itself achieved second place in the experimental network inference challenge (see entry for Team2). The performance of this entry in the DREAM challenge can be seen as a confirmation that the contest recapitulates known biology, and is an important reminder that prior biological knowledge should always be taken into account in practical applications of machine learning to biology.

L1-penalized Granger combined with the prior was attempted in the development phase, but did not perform as well as the prophetic version of the algorithm. This result highlights the utility of using all available data for the regression task, while allowing the direction of causality to be determined by the temporal relationship between observations. We note that the power of the individual regression tasks is weak, since each is driven by only four inhibitor treatments. Nevertheless, this is balanced by the large number of regression tasks, leading to an effective ensemble of weak learners.

References

1. Cerami, Ethan G., et al. "Pathway Commons, a web resource for biological pathway data." *Nucleic Acids Research* 39.suppl 1 (2011): D685-D690.
2. Bahadori, Mohammad Taha, and Yan Liu. "Granger Causality Analysis in Irregular Time Series." *SDM*. 2012.
3. Friedman, Jerome, Trevor Hastie, and Rob Tibshirani. "Regularization paths for generalized linear models via coordinate descent." *Journal of statistical software* 33.1 (2010): 1.
4. Qi Y, Suhail Y, Lin YY, Boeke JD, Bader JS "Finding friends and enemies in an enemies-only network: a graph diffusion kernel for predicting novel genetic interactions and co-complex membership from yeast genetic interactions." *Genome research* 18.12 (2008): 1991-2004.
5. Shojaie A, Michailidis G. “Discovering graphical Granger causality using the truncating lasso penalty.” *Bioinformatics* (2010) 26 (18): 517-523.
6. Margolin et.al. “Systematic Analysis of Challenge-Driven Improvements in Molecular Prognostic Models for Breast Cancer.” *Science Translational Medicine* (17 April 2013) 5 (181):181.
7. Cheng W, Yang TO, Anastassiou D. “Development of a Prognostic Model for Breast Cancer Survival in an Open Challenge Environment.” *Science Translational Medicine* (2013) 5 (181):181.

SC1 Network Inference: Team2

Summary

For experimental data, a biological prior is created by applying a simulated heat diffusion process to the constituent pathways from Pathway Commons. For *in silico* data, the network inference method ARACNE is used.

Introduction

There is a wealth of freely available biological pathway knowledge. Our goal in the HPN-DREAM challenge was to leverage the prior knowledge contained within the Pathway Commons resource to augment the methods for our other challenge submissions. The process has 3 main steps: 1) selection of relevant pathways from Pathway Commons; 2) application of heat diffusion to the selected pathways; 3) combination of relevant submatrices from the resulting diffusion kernels. Each of these steps is described in more detail below.

Method: Sub-challenge 1A

1) Selection of Relevant Pathways from Pathway Commons

To produce a biological pathway prior for the DREAM8 challenge, we started with 495 pathways from Pathway Commons version 3 (see https://www.synapse.org/#!Synapse:syn5588699). These pathways were downloaded from pathwaycommons.org in BioPax format. DREAM8 organizers provide a file that describes the antibodies used in the RPPA experiments. We used the antibody target information to identify individual pathways that contain at least two proteins represented on the RPPA platform in the HPN DREAM8 data. The proteins were not required to be direct neighbours in the pathways. We identified 263 DREAM8-relevant pathways that satisfied this criterion. Each pathway was reduced to a simple, undirected graph with nodes representing proteins and edges representing regulatory interactions. For proteins that form a complex, each of the constituent subunits was assigned an edge to any target of the complex.

2) Heat Diffusion on Selected Pathways

The main assumption behind the method to construct the prior is that proteins close together on a network are more likely to interact in the context of the HPN DREAM8 experiments, compared to arbitrary protein pairs. To this end, heat diffusion was utilized to quantify the closeness of proteins on a network. To compute a quantitative measure of distance between any two proteins in a graph, we calculated a single heat kernel for that graph. Let L_p be the Laplacian of the graph derived from pathway p. Then the heat kernel corresponding to a 0.1 time unit diffusion, as suggested by Qi et al. (2008) was calculated as

$$H_p = \mathrm{e}^{\left(-L_p * 0.1\right)}$$

where e^A is the matrix exponential of the matrix A. To generate a single biological prior matrix B from the individual pathway matrices H_p, we performed the following entry-wise update:

```
$B$<-0
for each pathway $p$
    for $j,k$ in interrogated proteins
        if $H_p(j,k)$ exists and $j \neq k$
        $B(j,k)$ <- $H_p(j,k)$
    end
end
```

The diagonal entries *B*(*j,j*) are maintained at zero, precluding self-links. The same biological prior information was used in cases where multiple probes assayed the same protein. The resulting prior information matrix *B* was not specific to either the cell line or stimulus condition.

Note that the code above contains a programming error in line five that we have purposefully reproduced here. As written, the final step of the code overwrites an entry in the accumulated *B* matrix. The final score for the (*j,k*) interaction is then set to the value of an arbitrary pathway-specific network containing the (*j,k*) edge that is encountered last by the algorithm. While the official version of the code used for the HPN DREAM8 submission used this faulty implementation, we have since corrected the issue and found an improvement in accuracy (0.783 compared to 0.771) when B is set to the average of the H_p matrices.

We applied the heat diffusion process to each of the 263 DREAM8-relevant pathways. The approach does not consider the directionality of edges in a graph, making use of the unsigned adjacency matrix *A*. For a given pathway, *A[i,j]* and *A[j,i]* are set to 1 whenever an edge is present between nodes *i* and *j*, and 0 otherwise. Let *D* be a diagonal matrix with *D[i,i]* equal to the number of genes that gene *i* interacts with (i.e., the degree of node i). The associated heat diffusion kernel is then given by the matrix exponential of *–t*(D-A)*, where *D-A* is known as the Laplacian matrix of *A* and *t* is a “time step” parameter. In all of our calculations, we set the value of t to 0.1, which has been shown useful in the context of protein-protein network discovery (Vandin 2012, Paull 2013).

3) Combine Relevant Submatrices from Diffusion Kernels
We extracted submatrices from each diffusion kernel by keeping only rows and columns that corresponded to antibody-targeted proteins. The extracted submatrices were then combined according to the superimposition pseudocode above, creating one matrix containing only antibody-targeted proteins. Protein names were then mapped back to their corresponding antibody names after reading off of the summary matrix. To convert the combined matrix of heat diffusion scores to confidence values, we divided by the largest score found in the entire matrix, converting all values to lie in the [0,1] range.

Method: Sub-challenge 1B
This submission was created using ARACNE, as described in the entry for Team25, under the “ARACNE gene network reconstruction” section.

Discussion
The biological prior was submitted by itself to the sub-challenge 1A and ranked as the 2nd best method in the final scoring. The top-scoring submission, made by Team1 (“Prophetic Granger”), used a causal inference algorithm combined with the biological prior by an equally-weighted averaging between the two approaches.

References

1. Evan O. Paull, Daniel E. Carlin, Mario Niepel, Peter K. Sorger, David Haussler and Joshua M. Stuart (2013) Discovering causal pathways linking genomic events to transcriptional states using Tied Diffusion Through Interacting Events (TieDIE). *Bioinformatics* 29:2757-2764
2. Vandin, F., et al. (2012) Discovery of mutated subnetworks associated with clinical data in cancer. *Pacific Symposium on Biocomputing*, 55-66.
3. Margolin, Adam A., et al. "ARACNE: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context." *BMC Bioinformatics* 7.Suppl 1 (2006): S7.

SC1 Network Inference: Team3

Summary

Edges are removed from a literature-based network by considering time-lagged correlation between phosphoprotein pairs, fold-changes in abundance through time and results from the time-course prediction challenge

Methods

We devised an algorithm to infer fold changes of the phosphoprotein levels over time comparing each time point relative to t_0. Species, which display high qualitative correlation with respect to amplitude and duration of the response over all experimental conditions considering each stimuli and inhibitor individually, were linked via edges. More precisely, when comparing two proteins we considered the changes from t to t + 1 for each individual inhibitor, stimulus, and cell line combination. Their directionality was determined from the speed of the dynamical behavior represented by the delay in responding to the stimulus and the steepness of the curve. In addition, information about ligand and inhibitor targets was used to determine the intersection between signaling pathways, especially in the case of SC1B. A threshold of 20% fold change with respect to the initial time point was applied. Changes in the protein abundance below that value were considered noise and not taken into account for potential edges, that is: not considering the species' behaviour as relevant. We determined this threshold by estimating fluctuations in those cell lines with duplicate time-courses. The effect of ligands and inhibitors was not explicitly modeled for network inference but their qualitative effect on phosphoprotein dynamics was taken into account when we set up the model.

For the initial model in SC1A, we relied rather heavily on information provided in literature and the online resource KEGG to ensure that our primary network would be as correct as possible. We reasoned that this initial model would contain all potential edges allowing us to refine the network by removing or reducing edges, for which we saw no evidence. Even at this stage we were able to individualize the networks to the cell lines based on literature research.

The initial model for SC1A was tailored to the cell lines and stimuli according to the results obtained from our analysis resulting in an individual network per cell line and stimulus. An edge reported in literature, for which we could not determine any correlation in the dynamic behavior was removed from the network. Additionally, regulatory feedbacks via AKT and MEK, that had not been part of the initial model, were later included based on the inhibitor data.

Refinement of network using time course prediction results

Putative interactions and non-validated edges, based on literature and thresholding, were included in the parameter estimation for sub-challenge 2 (time course prediction) as a phosphorylation reaction catalyzed by the source of the edge. The edge pA $\rightarrow$ pB would then be represented as:

$$\frac{dpB}{dt} = k_1 * [B] * [pA]$$

In order not to introduce too much of a bias, we set the initial possible interval for the kinetic parameter k_1 to include several orders of magnitude, ranging from $0.001s^{-1}$ to $100s^{-1}$. These ODE-based models gave us information about the ability of edges to reproduce the dynamical behavior seen in the data. If the parameter k_1 for an edge received a value close to zero, we considered this edge to be eliminated from the network, thus further refining our networks to the individual cell lines and stimuli.

This algorithm was implemented in R with a custom-written script and required only a few minutes of computation time on a personal computer.

SC1 Network Inference: Team4

Summary

An ensemble approach combining prior knowledge with results from L1-penalized Granger causality and the GENIE3 algorithm (the prior knowledge network was also submitted as a stand-alone prediction; see Team2 method).

Sub-challenge 1A: The mean of L1-penalized Granger causality (non-prophetic version; see Team1), GENIE3 (see Team17) and a prior network (see Team2) were taken for each of the inferred phosphoprotein links.

Sub-challenge 1B: The mean of L1-penalized Granger causality (non-prophetic version; see Team1) and GENIE3 (see Team17) were taken for each of the inferred phosphoprotein links.

SC1 Network Inference: Team5

Summary

A random forest classifier, with a literature-derived network used as a "gold standard", predicts existence of edges from several measures of pair-wise association, calculated using multiple sources of data.

Introduction

Our approach uses a random forest classifier to infer casual networks. The instances (or observations) to be classified are all possible pairs of nodes (phosphoproteins) and the classes (or outcomes) are binary, indicating the presence/absence of an edge between the node pairs. The attributes for each node pair are several measures of association derived from the experimental proteomics data or *in silico* data provided with the challenge (EXP) and from prior knowledge (PK): literature-derived attributes and knowledge-based inferred functional relations and signalling networks. Class labels were derived using a manually curated “gold-standard” network.

Methods

We trained random forest (RF) classifiers using a manually curated network (MCN) and several attributes for each pair of phosphoproteins (from now on referred to as “proteins”) in sub-challenges 1A and 1B.

Manually curated network. The MCN consisted of 33 proteins and 38 edges and was formed using information in Uniprot-Swissprot for the 48 proteins in the sub-challenge 1A.

RF attributes. We built an array of scores describing the potential causality between each pair of proteins. Scores are derived from the experimental or *in silico* data provided with the challenge (EXP) and from prior knowledge (PK), as described below. For the *in silico* sub-challenge 1B, where the nodes were anonymized, we used only EXP scores.

EXP attributes. We used protein levels reported in sub-challenges 1A and 1B (our analyses used only the ‘main’ datasets for sub-challenge 1A). First, we calculated the mean and standard deviation of protein levels for each combination of protein, cell-line and inhibitor at time 0 (no stimulus). We used these values to transform all the time-courses (for the corresponding combination of protein, cell-line and inhibitor) to z-scores. EXP attributes can be divided into two categories: (1a) continuous functions, and (1b) discrete scores. In the 1a category we used different correlation functions (Pearson, Spearman, and cross-correlation using Fast Fourier Transform and Wiener filtering of noise). For each protein, cell-line and inhibitor condition, we constructed a vector containing the normalized protein values at all time-points and all stimulus conditions. We used these vectors to calculate the correlation values between any given pair of proteins (X and Y), either at the same time point (comparing X(t) with Y(t)) or consecutive time points (comparing X(t) with Y(t+1)). In the category 1b we discretized the protein z-scores into binary values (0 or 1) using different thresholds (from 0.5 to 3.5, increasing by 0.5). Pairs of proteins were scored using several statistical measures for the comparison of their vectors: Probability of being related according to a Chi-squared test; Ratio between equal and different values in the vector; Mutual Information; and Direct information (Morcos et al., 2011). This results in, for each pair of phosphoproteins and for each association measure, several association scores corresponding to specific cell-line and inhibitor conditions. For each pair of phosphoproteins, these association measures were summarized with their centrality measures (mean and median) and their extreme values (maximum and minimum).

PK attributes. Five approaches were used to predict a relationship between pairs of proteins in the experimental network inference sub-challenge, none of which used the proteomics time-course data provided with the challenge: 3a) *Known protein-protein interactions*. We used BIANA (Garcia-Garcia et al., 2010) (http://sbi.imim.es/BIANA.php) to build a protein-protein interaction network with the proteins of the sub-challenge. We computed scores describing if there are known direct and/or indirect interactions between two proteins reported by any experimental method. 3b) *Functional relations*. We built a network as before, but we used the combined score provided by the database of STRING. 3c) *iLoops predictions*. Prediction of protein-protein interactions based on local structural features (Planas-Iglesias et al., 2013) (http://sbi.imim.es/iLoopsServer/). 3d) *Functional similarity*. We computed the putative functional relationship based on GO as described by Wang et al., 2007. 3e) *Phosphorylation event predictions*. We used the *iGPS* software to predict potential kinase-substrate pairs (Song et al., 2012). The five approaches predicted a relationship with some score of reliability. We used these scores as attributes in the RF classifier.

Random forest classifier. The attributes were used to train RF classifiers with WEKA (http://www.cs.waikato.ac.nz/ml/weka/). The edges contained in the MCN were used as a positive training class (presence of an edge) while a negative class (absence of an edge) of the same size was built with random connections between protein pairs (excluding those in the MCN). To avoid biases in the negative sample selection, 500 different RF classifiers were trained. We tested the classification performance using a 10-fold cross-validation. Redundant pairs with identical attributes were not found. Edge scores between all protein pairs were obtained by computing the mean of all RF classifiers. We allowed the RF classifier to override the MCN edges, thus allowing low scores for some MCN edges.

This resulted in a "scored global network prediction". To convert this "global network" to the 32 specific sub-networks (for each cell line and stimulus) required for the sub-challenge 1A, we applied a penalization factor to remove irrelevant edges depending on the cell-line and stimulus condition. Protein pairs in which both proteins had a level of variation greater than 1.5 standard deviations were not penalized (penalty factor of 1.0). In contrast, the penalty factor ranged from 1.0 to 0.5, proportional to the lower level of variation showed by any protein in each protein pair (a level of variation close to 0 corresponds a penalty factor of 0.5, while a level of variation close to 1.5 standard deviations corresponds to a penalty factor of 1.0).

Edge directionality. In order to distinguish A→B from B→A, thus to infer the direction of causal relationships, additional RF classifiers (RF_D) were trained using only EXP attributes. 10-fold cross-validation was then applied to a dataset consisting of MCN correctly directed pairs as a positive class and the MCN pairs with inverted direction as a negative class. Edge scores in which the incorrect direction was predicted were penalized by 0.75.

Results and discussion

We proposed an approach that combines the data from different conditional networks with knowledge-based and computational inferences to reproduce a global prediction of the causal network. Our method lacks a validation approach to transform the predicted global network to the 32 specific sub-networks required by sub-challenge 1A. Furthermore, the lack of a negative causality standard (i.e. negative feed-back loops) hampered our ability to set a correct direction for edges in some strong bi-directional cliques present in our models. In a previous leaderboard round we applied just the random forest approach (without the "filters" to decipher the directionality of the edges). For sub-challenge 1B this achieved an AUROC of 0.68 compared with the final submission score of 0.60.

References

1. Morcos F. et al. Direct-coupling analysis of residue coevolution captures native contacts across many protein families. *Proc. Natl. Acad. Sci. USA*. 2011 Dec 6;108(49).
2. Garcia-Garcia J, et al. Biana: a software framework for compiling biological interactions and analyzing networks. *BMC Bioinformatics*. 2010;11:56.
3. Planas-Iglesias J, et al. Understanding protein-protein interactions using local structural features. *J Mol. Biol*. 2013 Apr 12;425(7):1210-24.
4. Wang JZ, et al. A new method to measure the semantic similarity of GO terms. *Bioinformatics*. 2007;23:1274–1281.
5. Song C, et al. Systematic analysis of protein phosphorylation networks from phosphoproteomic data. *Mol. Cell Proteomics*. 2012 Oct;11(10):1070-83.

SC1 Network Inference: Team6

Summary

Network topology and parameters are optimized by minimizing an objective function based on a linear Ordinary Differential Equation (ODE) model, using a greedy search starting from a knowledge-based network topology.

Introduction

A signaling network can be described by a system of ODEs, whose parameters can be used to describe its dynamic behavior, once the interacting proteins and the type of these interactions are determined. An integrative approach is proposed to identify context-specific network models characterized by a set of parameters describing relationships between proteins, together with their degradations and synthesis processes. For each cell line/stimulus, the method uses the time-course data information and applies a greedy approach to infer the interactions among proteins, starting from a prior network derived from public databases.

Methods

The algorithm was applied to all cell lines, stimuli, time points and inhibitors, focusing only on the main datasets of the sub-challenge. When available, replicated measurements collected in the same experimental condition and time were mediated. Data were normalized, for each protein, by its maximum concentration across all the experimental conditions.

The approach is described according to the following steps:
Prior knowledge network. A prior network topology was defined using the following resources to retrieve specific information about protein-protein interactions: KEGG pathways (Kanehisa et al., 2014), Uniprot (The UniProt Consortium, 2014) and Phospho.ELM (Dinkel et al., 2010).

Module-network inference and parameter estimation. For each protein, regulators were defined accordingly to the prior knowledge network topology and the dynamic data were modeled using linear ODEs:

$$\dot{Q}_p(t) = -k_{deg}\, Q_p(t) + \sum_{i=1}^{q} k_{i,p}\, y_{i,p}(t) \; - \sum_{i=1}^{r} k_{i,n}\, y_{i,n}(t) \; + k_{in} + k_{synth}$$

Where *Q_p* is the concentration of the protein *p*, *k_deg* is the degradation rate, *k_in* is the constant input, *k_synth* is the synthesis rate, *k_(1,p),...,k_(q,p)* are the rates of the regulatory proteins that act positively on the protein *p*, *k_(1,n),...,k_(r,n)* are the rates of the regulatory proteins that act negatively on the protein *p* and *y_i* are the regulatory inputs from other proteins. Protein expression was modeled independently for each cell line and stimulus. Weighted Least Squares (WLS) were used for the parameter identification to simultaneously minimize an objective function for each inhibitory condition, using the standard deviations of the measurement error as weights. Indeed, the standard deviations appear to be essentially constant for each protein, but vary across different inhibitor conditions. Considering only the edges determined by the prior knowledge network and starting from one edge at a time, an approach that iteratively adds the most significant edge according to the F-test results was used to find the best subset of prior network edges that explain the data. The test was applied to the residual sum of squares (RSS), using a threshold on p-values equal to 5%. The initial values of the parameters were fixed within a biological range consistent with the literature: [0-20] min^{-1}.

Global network inference. A global network able to explain the data with the minimum number of edges was identified. Starting from the network obtained by assembling each protein modules previously identified, WLS and F-test were used to remove unnecessary edges. Results obtained from the identification of the protein modules were used as initial values of the parameters for the global model.

Ranking. For each configuration (protein, regulators) selected by the F-test, a confidence based on the log(RSS), normalized in a 0.9-1 range, was associated to the corresponding edges. A confidence equal to 0.5 was assigned to the other edges previously removed but belonging to the initial prior network.

In silico data. Protein dynamics were equally modelled with a linear system of ODEs. However, to generate temporal profiles, we integrated each equation of the system with a piecewise analytical solution, assuming constant regulators in each time step. This allowed us to independently search for the best set of regulators for each protein, starting with a model comprising all possible regulators and then iteratively removing the least useful regulator according to the F-test. Edges in the final network were ranked in decreasing order of the F-test p-value. For each configuration (protein, regulators), the fit to temporal data was carried out by minimizing the RSS through an evolutionary algorithm, Covariance Matrix Adaptation - Evolution Strategies (CMA-ES; Hansen et al., Evolutionary Computation 2003). The effect of the inhibitors was modelled by clamping protein activity to a minimal value for the entire experiment, while the stimuli were simulated by adding a constant term (1 for high stimuli, 0.01 for low stimuli) to the differential equation corresponding to the stimulated proteins.

The algorithms implementing the methods were written in both R and MATLAB language, requiring additional packages R.matlab, Statistical Toolbox, Control System Toolbox, Optimization Toolbox and the MATLAB implementation of the CMA-ES algorithm (https://www.lri.fr/~hansen/cmaes_inmatlab.html#matlab).

Discussion

The implemented model-based method using linear ODE system provides an accurate description of protein interaction networks in a time-dependent manner, which describes the most important dynamic features during both transient and steady state conditions. Moreover, the application of the F-test is able to decrease the number of redundant network edges, thus providing more accurate parameter estimations.

References

1. Kanehisa M, Goto S, Sato Y, Kawashima M, Furumichi M, Tanabe M (2014) Data, information, knowledge and principle: back to metabolism in KEGG. *Nucleic Acids Res.*, 42 (Database issue):D199-205.
2. The UniProt Consortium (2014) Activities at the Universal Protein Resource (UniProt). *Nucleic Acids Res.* 42 (Database issue): D191–D198.
3. Dinkel H, Chica C, Via A, Gould CM, Jensen LJ, Gibson TJ, Diella F (2010) Phospho.ELM: a database of phosphorylation sites—updated 2011. *Nucleic Acids Res*. 39 (Database issue):D261-7.

SC1 Network Inference: Team7 - FunChisq

Summary

"FunChisq": a novel functional chi-square test to infer causal network topology based on nonparametric functional dependency from data discretized by optimal k-means clustering for each variable.

Introduction

Previous statistics employed in network inference such as correlation coefficients, mutual information, Pearson's chi-square of association (Song et al., 2009), can detect linear or non-linear associations, but not interaction directionality. Other regression methods, including those nonparametric methods using splines, can detect directionality but must assume a functional form in advance. To overcome the disadvantage of both types of approach, we developed a novel functional chi-square test called FunChisq (Zhang and Song, 2013) to infer the directionality of interactions by functional dependency without assuming parametric functional forms for interactions.

Methods

Data sets used. For the sub-challenge 1A (SC1A, experimental data collected from four breast cancer cell lines), we used only the main dataset. For the sub-challenge 1B (SC1B; *in-silico* data), we used the entire data set.

Prior knowledge. We did not use prior knowledge regarding network topology when performing network inference for SC1A. Due to anonymity of node identities, prior knowledge could not be used for SC1B.

Pre-processing. Data for each protein were first quantized using R package Ckmeans.1d.dp (Wang and Song, 2011) with the number of quantization levels set to equal the number of components in Gaussian mixture models estimated by optimizing the Bayesian information criterion using R package mclust (Fraley 1999). When only one Gaussian component was selected by mclust, we changed it to three to capture low, intermediate, and high levels. For the SC1A experimental datasets, the actual number of quantization levels ranged between 2 and 6. For SC1B *in silico* dataset, the actual number of quantization levels ranged between 2 and 5. No additional pre-processing such as log transform was performed.

Network inference by FunChisq. We developed a functional chi-square test to detect causal interactions by nonparametric functional dependency (Zhang and Song, 2013). We use the principle that a causal interaction is a mathematical function from cause to effect (Simon and Rescher, 1966). In the functional chi-square test, each interaction is assessed for its functional strength as evidence for causality. Temporal information was not used, though potentially it could strengthen network inference. Interactions were ranked by normalized chi-squares from FunChisq. Although many-to-one functional dependency is a user option of the software, we examined only one-to-one interactions in both SC1A and SC1B. Normalized chi-square scores of all possible directional pair-wise interactions were computed. We used the same program and parameters for both the experimental data (SC1A) and the in silico data (SC1B).

Post-processing. As required for challenge result submission, the list of all interactions were sorted in decreasing order by the chi-square scores, which were further normalized to [0,1] by dividing distance from the minimum score by the range of all scores. Let X be the ranked list of initial scores (the normalized chi-squares). We linearly mapped X to [0,1] by $X' = (X - \min(X))/(\max(X) - \min(X))$ to obtain the final scores that we submitted.

Implementation. The FunChisq code was written in *C++.* The pre- and post-processing code was implemented in R. We uploaded our software and R scripts to Synapse: syn2450780 reproduces results for the SC1A experimental dataset and syn2450779 reproduces our submission for the SC1B *in silico* dataset. An R package FunChisq implementing the basic functional chi-square test is freely available from Comprehensive R Archive Network (CRAN).

Discussion

Before the final submission, we made only one leader board submission in the 7th week (also the last chance to receive feedback before final submission). We had first used FunChisq to eliminate any interaction with a greater p-value (less significant) than its reverse. Then we ranked interactions by normalized Pearson's chi-squares of independence. The only leader board feedback we received for our initial submission reported promising ranking for SC1A but mediocre ranking for SC1B, suggesting there might exist many bi-directional interactions and eliminating weaker ones could have led to many false negatives. Towards the time of final submission, we decided to rely entirely on FunChisq to rank interactions without using the standard Pearson's chi-square – the best performing approach as suggested by our evaluation of several possible alternative methods (all chi-square based) on DREAM5 network challenge data sets. There was no tuning of parameters, i.e., the maximum number of quantization levels to search for in pre-processing was all set to 9. This is not a sensitive parameter as long as it is large enough, because the automatically selected quantization levels were not greater than 5/6 in SC1A/1B.

We only considered one-to-one interaction in generating the results for both network inference sub-challenges, though many-to-one interactions are implemented in the FunChisq software and FunChisq could also model combinatorial effects. This choice was made because in our past experience with DREAM5 network inference data sets, combinatorial effects could not be effectively identified. We ponder that the experimental perturbations might be insufficient to infer interactions involving more than two genes.

After the benchmark software tool for SC1A was released, we found that the AUROC of FunChisq rose to the second after we tried a different way to use the data, while keeping the software setup same as before otherwise. Specifically, we inferred a single network for each cell line by merging the data from all 8 stimuli. Then, for each cell line, we submitted the same identical result 8 times (once for each stimuli) to the SC1A benchmark tool. The resulting AUROC of FunChisq increased from 0.7148 to 0.7724. This is a very important result as it suggests that even without using any prior, combining data from the same underlying network under different stimuli could dramatically improve network inference performance.

References

1. Song, M., C. K. Lewis, E. R. Lance, E. J. Chesler, R. K. Yordanova, M. A. Langston, K. H. Lodowski, and S. E. Bergeson. Reconstructing generalized logical networks of transcriptional regulation in mouse brain from temporal gene expression data *EURASIP Journal on Bioinformatics and Systems Biology*, vol. 2009, Article ID 545176, 13 pages, doi:10.1155/2009/545176
2. Fraley, C. (1999). MCLUST: Software for Model-Based Cluster Analysis. *Journal of Classification*, vol.16, pages 297–306. doi:10.1007/s003579900058
3. Wang, H. and M. Song. (2011) Ckmeans.1d.dp: Optimal k-means clustering in one dimension by dynamic programming. *The R Journal*, vol. 3, no. 2, pages 29-33.
4. Zhang, Y., Song, M. (2013) Deciphering interactions in causal networks without parametric assumptions. *arXiv*:1311.2707.
5. Simon, H. A. and Rescher, N. (1966). Cause and counterfactual. *Philosophy of Science*, vol. 33, no. 4, pages 323–340.

SC1 Network Inference: Team8

Summary

Gradient tree boosting regression and a Markov assumption are used to model the time series data, with network connectivity derived using a combination of the most frequently selected features over the boosting rounds and information from a prior knowledge network

Introduction

Our approach uses a traditional dynamic Bayesian network model, augmented to include gradient boosting regression to fit the response of each phosphoprotein. The salient features of this algorithm are (a) it learns a strong regression function from an ensemble of weaker regression trees, and (b) it re-weights the training data at each iteration to emphasize hitherto poorly modeled examples (Hastie et al., 2004). After the training phase the parents and edge weights for any particular node in the network can be recovered from the relative frequencies of the other nodes incorporated in the boosted model. The underlying methodology is based on an ensemble regression approach that can capture nonlinear interactions among covariates. Our approach was validated on known literature models (Xu et al., 2010) of signalling networks before applying to the challenge data.

Methods

Dynamic Bayesian networks ‘unroll’ Bayesian networks by making the Markovian assumption that phosphoprotein activity at each time point depends only on the values at the previous time point. We use a gradient boosting approach to fit a model that predicts activity for each phosphoprotein individually using the values of all phosphoproteins at the previous time point. The predictions are then pooled and used as input for the next time step. Our method was identical for both the *in silico* and experimental components of the sub-challenge, with the exception of the biologically informed prior which enhanced performance on the experimental data. The prior was manually curated by mining the literature on RTK signalling and incorporated as an initial feature weighting for each protein.

Several pre-processing steps were performed prior to learning the model. Each experimental condition was mean-centered. Missing time points were mean-imputed to facilitate comparison between experimental conditions. Multiple replicates were averaged together to produce one time series for each experimental condition. Only the main data was used, not the full data. We explored training our model on both the response level and response rate, and found response level to have better performance. Parameters, including maximum tree depth, number of boosting rounds, and learning rate were set using a grid search and evaluated under cross-validation.

All feature weights are rescaled to the interval [0, 1] and are used to populate the adjacency matrix. In this approach, therefore, if antibody X was repeatedly used to build the predictive function for antibody Y, then we assume that X is a parent of Y in the network. This workflow is illustrated in figure 1 below for the case of fitting a boosted model to target species MAPK_pT202_Y204.

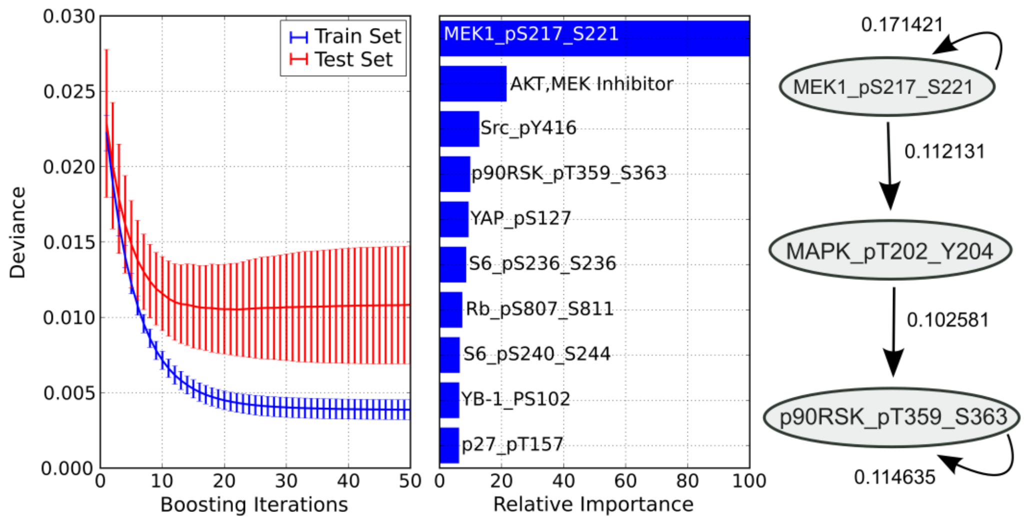


Figure 1. An example of the output of our algorithm, fitting a boosted model to target species MAPK_pT202_Y204. On the left is the train and test set performance from cross-validation folds on the provided training data. The center shows feature importances, measured as the number of times a given feature appears in the collection of weak learners. AKT, MEK Inhibitor is the categorical variable accounting for the presence or absence of the inhibitor. On the right is an extracted sub-module of our network, where edge weights are taken from feature weights in the model training step. In this case we reproduce known patterns in MAPK signalling.

Inhibitors were modeled using a perfect fixed-effects model (Spencer et al., 2012). The stimulus was not explicitly modeled. We dealt with stimuli in two ways: by grouping datasets across stimuli, and by training independent models for each stimulus. We found training separate models for each stimulus performed better than grouping across stimuli.

Our algorithm was implemented in Python, using the scikit-learn library for model development and testing. The pandas library was used for data manipulation and preprocessing.

Discussion
The gradient tree boosting approach performs well on both *in silico* and breast cancer RPPA data, particularly when augmented with a small number of canonical edges in the form of a biological prior. We expect ensemble models such as gradient boosting regression to gain increasing traction in this field due to their advantages over traditional regression approaches, including robustness to overfitting and ability to identify nonlinear effects.

References

1. T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, volume 1. 2004.
2. Simon E F Spencer, Steven M Hill, and Sach Mukherjee (2015) Inferring network structure from interventional time-course experiments. *Annals of Applied Statistics* 9: 507-524.
3. Tian-Rui Xu, Vladislav Vyshemirsky, Amelie Gormand, Alex von Kriegsheim, Mark Girolami, George S Baillie, Dominic Ketley, Allan J Dunlop, Graeme Milligan, Miles D Houslay, and Walter Kolch. Inferring signaling pathway topologies from multiple perturbation measurements of specific biochemical species. *Science Signaling*, 3(113):ra20, January 2010.

SC1 Network Inference: Team9

Summary
Network structure was predicted by integrating a prior knowledge network with results of time-lagged correlation analysis on pair-wise phosphoprotein abundances.

Introduction
We used a simple approach to build a causal network that explains the different interactions that occur between the proteins in sub-challenge 1. The approach is based on a time-lagged correlation method that incorporates prior knowledge from the KEGG database. The motivation behind using a simple correlation approach lies in its ability to capture the relationships that occur between the proteins across time.

Methods
A time-lagged correlation method was used to solve sub-challenge 1. In addition, prior knowledge regarding the network was extracted from a public database and used to build the model. Time-lagged correlations between all possible pairs of phosphoproteins were examined. In order to avoid an unreasonably long lag time, the maximum lag time interval for the experimental data spans 3 time points[1]**.** The correlation $\boldsymbol{R_{ij}}(\boldsymbol{\tau})$ between protein *i* and protein *j* with a time lag $\boldsymbol{\tau}$ is calculated as follows:

$$\boldsymbol{R_{ij}}(\boldsymbol{\tau}) = \frac{S_{ij}(\tau)}{\sqrt{S_{ii}(0)S_{jj}(0)}}$$

where

$$S_{ij}(\tau) = \sum_{t=0}^{N-\tau} (X_i(t) - \bar{X}_i)(X_j(t+\tau) - \bar{X}_j),$$

$\bar{X}_i$ is the averaged expression value of protein *i* across all time points and *N* is the number of time points. The value of $\boldsymbol{\tau}$ that produces the maximum value for $\boldsymbol{R_{ij}(\tau)}$ is used as the lagged time for protein *i* and protein *j* [1].
The threshold for the correlation score, which determines which edges should be considered, is tuned to ensure a manageable number of edges are identified. The threshold value of 0.7 obtained the highest rank, according to the leader board.

The KEGG database[2] was used to provide the external prior information for the time-lagged correlation model. For sub-challenge 1A, all KEGG maps that cover the proteins from the "Main" dataset were extracted. Since the data contain the expression level for phosphoproteins, the protein-protein relationships and the phosphorylation and de-phosphorylation interactions identified by the KEGG database were used. All the maps obtained from the KEGG database were merged, and a prior network was generated. If an edge is found between two proteins, and the type of the edge is a protein-protein interaction, a phosphorylation/de-phosphorylation, or an inhibition/activation interaction, an edge is created between the target phosphoproteins. The inhibition/activation information is integrated in the network through edge values of either +1 or -1. The network inferred by the time-lagged correlation analysis is compared with the prior network inferred from the KEGG database. If there is an edge conflict with the prior network, the edge is deleted. For example, if the prior network has an edge between proteins A and B representing an inhibiting interaction, while in the predicted network the edge between A and B is an activation edge, then the edge is deleted from the final network.

In the experimental sub-challenge, only the "Main" dataset is used. The same methodology for inferring networks is used in the experimental and *in silico* challenges. Moreover, no further processing or cross-validation schemes were performed. We implemented the time-lagged correlation equation from [1], as well as the pipeline that generates the final network files, with custom-written scripts in Perl. We did not use other tools or packages.

Discussion

During the challenge we follow two different paths to solve the problem. It is found that a simple solution based on correlations between the phosphoproteins produces better results compared to a more complex approach based on Dynamic Bayesian Networks (DBNs)[3]. Additionally, prior knowledge provided through the KEGG database proved to be very useful in inferring the network. One important conclusion is that simple approaches should always be considered first, as they can often produce very good results compared to more complex approaches. Future work includes improving the approach to model the effects of interventions in building the signalling networks with several nodes being inhibited, and utilizing cross validation to determine the threshold for the correlation method. Additionally, we plan to revisit the DBN approach.

References

1. Schmitt, W. A., Raab, R. M. & Stephanopoulos, G. Elucidation of gene interaction networks through time-lagged correlation analysis of transcriptional data. *Genome Res.* **14,** 1654–63 (2004).
2. Kanehisa, M. & Goto, S. KEGG: kyoto encyclopedia of genes and genomes. *Nucleic Acids Res.* **28,** 27–30 (2000).

3. Sima, C., Hua, J. & Jung, S. Inference of Gene Regulatory Networks Using Time-Series Data: A Survey. *Curr. Genomics* **10,** 416–429 (2009).

SC1 Network Inference: Team10

Summary

We developed a *de novo* regression approach using a stationary Markov assumption and truncated singular value decomposition to predict network structure. No prior biological knowledge was used.

Introduction

We have developed a prediction algorithm for protein phosphorylation network inference using truncated singular value decomposition (SVD) Our method is based on a stationary Markov assumption and uses a regression method comparable to Lasso regression.

Methods

We make a first-order Markov and stationarity assumption [1] that: every variable at a given time point t_i only depends on variables at the previous time point t_{i-1}. Furthermore, we assume that the value of a variable will not change without the influence of any other variables (including self-regulation). Equation 1 is the foundation of our method:

$$E_k(t_{i+1}) = E_k(t_i) + \sum_{\alpha=1}^{n}(E_\alpha(t_i) \times R^0_{\alpha,k} \times KD_\alpha) + \epsilon \qquad (\forall t \in [t_1..t_{T-1}],\ k \in [1..N]) \qquad (1)$$

Where $E_k(t_i)$ represents the phosphorylation value for protein k at the i_{th} time point, $R_{\alpha,k}$ is the relationship factor indicating how much protein α will affect protein k, N is the number of proteins and ε is the error factor. KD_α is the knockdown factor, set to 0 if α is targeted by an inhibitor, or to 1 otherwise. We replace $R^0_{\alpha,k} \times KD_\alpha$ with $R_{\alpha,k}$ in this notation.
Combining Equation 1 for all proteins, all time points and all inhibitors, we can form the following equation:

$$\begin{pmatrix} R_{1,1} & R_{2,1} & \cdots & R_{N,1} \\ R_{1,2} & R_{2,2} & \cdots & R_{N,2} \\ \vdots & \vdots & \ddots & \vdots \\ R_{1,N} & R_{2,N} & \cdots & R_{N,N} \end{pmatrix} \begin{pmatrix} E_1(t_1) & \cdots & E_1(t_{T-1}) \\ E_2(t_1) & \cdots & E_2(t_{T-1}) \\ \vdots & \ddots & \vdots \\ E_N(t_1) & \cdots & E_N(t_{T-1}) \end{pmatrix} = \begin{pmatrix} (E_1(t_2) - E_1(t_1)) & \cdots & (E_1(t_T) - E_1(t_{T-1})) \\ (E_2(t_2) - E_2(t_1)) & \cdots & (E_2(t_T) - E_2(t_{T-1})) \\ \vdots & \ddots & \vdots \\ (E_N(t_2) - E_N(t_1)) & \cdots & (E_N(t_T) - E_N(t_{T-1})) \end{pmatrix} \qquad (2)$$

For the convenience of description, we assign an abbreviation for each matrix in Equation 1:

$$RT = (T^+ - T) \qquad (3)$$

where each element in T is the observed value for a protein at a given time point, while corresponding element in T^+ is the observed value for the same protein at the next time point. Both T and T^+ can be obtained from the input data and R is the unknown relationship factor matrix we are interested in inferring. The problem can be solved by:

$$R = (T^+ - T)\ \ T^{-1} \qquad (4)$$

Because T is not a square matrix, it is not invertible. T^{-1} in Equation 4 is the pseudo-inverse of matrix T. There are numerous pseudo-inverse methods, such as singular value decomposition (SVD) [2], QR method, L1-regulation and L2-regulation. In order to handle the noise ε mentioned in Equation 1, we used truncated SVD [3], a variant of SVD, in our approach.
The above procedure is repeated to infer a network (or relationship factor matrix) for each cell line/stimulus pair

Discussion
We used the same method for sub-challenge 1 and sub-challenge 2. Its performance was mediocre in sub-challenge 1 (ranked 10 in SC1A (experimental), ranked 5 in SC1B (*in silico*), and ranked 3 for combined SC1A and 1B). However, it performed remarkably well in SC2, predicting network responses upon drug intervention, which was the winning method of this sub-challenge (ranked 2 in SC2A (experimental), ranked 3 in SC2B (*in* silico), and first for combined SC2A and 2B). We do not yet understand the difference of performance in SC1 and SC2, in which the latter was directly built on top of the former. One possibility is that in SC1 the gold standard of evaluation includes incomplete knowledge of signalling interactions in databases, which we did not use in our de novo model. This information could potentially be used to improve our model. On the contrast, SC2 involves only experimental measurement of phosphorylation levels, and rely on no prior knowledge in evaluation, to which our *de novo* model has an advantage in predicting the outcomes.

References

1. Friedman, N., K. Murphy, and S. Russell. Learning the structure of dynamic probabilistic networks. *Proceedings of the Fourteenth conference on uncertainty in artificial intelligence*. 1998: Morgan Kaufmann Publishers Inc.
2. Golub, G.H. and C. Reinsch, Singular value decomposition and least squares solutions. *Numerische Mathematik*, 1970. **14**(5): p. 403-420.
3. Henry, E. and J. Hofrichter, Singular value decomposition: application to analysis of experimental data. *Essential Numerical Computer Methods*, 2010. **210**: p. 81-138.

SC1 Network Inference: Team11

Summary
Boruta, a wrapper feature selection method, utilizing random ferns classifier as an importance source, is used to perform feature selection for each node in the network.

Introduction
In this approach, network inference was treated as a series of regression problems. Specifically, the abundance of each node was modelled as a function of other nodes' abundances, and the independent predictors of time and applied inhibitor. In this view, the application of a feature selection method to each such regression problem should reveal the source nodes of edges targeting corresponding target nodes. This approach will find all correlated edges, in addition to casual edges. Causal edges were identified as those where the target node abundance was found to depend on the presence of an inhibitor. These causal edges were reported in the final result.

Methods
The full data set was used in the analyses, and pre-processed in the following way. First, the data from all treatments were combined and then used to create a set for each stimulus-cell line pair in the experimental dataset, as well as a single set for the *in-silico* data. Second, time point 0 was randomly down-sampled to avoid over-representation. For the experimental networks, the set of independent predictors consisted of time and inhibitor features, while for the *in-silico* network it also included stimulus features. After pre-processing, the following algorithm was used for network inference. For each node of the network, we create a regression problem where node abundance is modeled as a function of all other nodes' abundances and independent predictors: time and applied inhibitor, and, for the *in silico* network, stimulus features. Next, a feature selection method is applied to this problem and returns a list of

selected predictors. If this list does not contain the inhibitor feature, it is discarded and the algorithm proceeds to the next node. If the inhibitor is contained within the selected feature list, edges from the selected nodes to the currently investigated node are added to the inferred network.

The feature selection was performed with the Boruta algorithm [1] utilizing a random ferns classifier as an importance source. The random ferns classifier was used because of its computational efficiency [2,3]. Because this algorithm is classification-only, regression problems generated by the network inference method were dynamically converted into classification problems by quantizing the decision into several equally sized bins.

The entire method has 3 parameters: number of ferns (N), the ferns' depth (D), and the number of bins used to convert the problem from regression to classification before applying random ferns (nQ). The selection of N becomes irrelevant if it is sufficiently large, so a fixed value of 10,000 was used. D and nQ control the complexity of interactions that can be modelled by a fern forest, and thus the recall of the method; however setting them too high may overload the model with noise and produce many false positives. To this end, for D and nQ, the method was applied for all combinations of D=3, 5, 7 and nQ=3, 5, 7. The method has a stochastic component, and therefore a run for each parameter pair was additionally repeated 20 times to explore the prediction space for these parameters. For the final submission, an ensemble approach was utilised: specifically, all obtained sets of edges were simply joined, and the number of times each edge was detected was used as its confidence score. This approach did not rely on biological knowledge or manual tuning. The analysis was implemented in R [4] using the Boruta and rFerns CRAN packages.

Discussion

This method proved to be quite effective considering that it did not rely on the use of prior information, and made no assumptions about the data. There are two limitations to this approach. First, random ferns requires that the data be quantized, which likely leads to loss of information. The algorithm would certainly benefit from using a method that can assess attribute importance by directly performing regression. The second limitation is that Boruta is an all-relevant feature selection method and therefore, in an ideal case, is expected to find at least a Markov blanket of a certain node X, that contains three types of nodes: 1) the nodes influencing it, 2) nodes influenced by X, and 3) other nodes that influence the nodes influenced by X. In practical terms, this means a significant portion of the edges detected may be false positives. To improve upon this, the proposed methodology could be followed by some refinement procedure that would further reduce the set of edges, or it could be used as a member of an ensemble of other methods.

References

1. Kursa, MB, and Rudnicki, WR. “Feature Selection with the Boruta Package.” *Journal of Statistical Software* 36, no. 11 (2010).
2. Kursa, MB. “rFerns: An Implementation of the Random Ferns Method for General-Purpose Machine Learning.” *Journal of Statistical Software* 61, no. 10 (2014). http://arxiv.org/abs/1202.1121
3. Kursa MB. “Robustness of Random Forest-based gene selection methods. *BMC Bioinformatics* 15:8 (2014).
4. R Core Team. “R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria (2013). http://www.R-project.org.

SC1 Network Inference: Team12

Summary

Network inference based on (static and time-lagged) mutual information and Bayesian model averaging for linear regression (the latter was applied to *in silico* data only).

Introduction

With time series data, it is possible to do both dynamic network inference and static network inference. The question is how to combine the two types of inference to achieve a better prediction. Our observations in a number of artificial datasets indicate that dynamic inference tends to achieve a high AUPR (area under precision-recall curve) while static inference tends to achieve a high AUROC (area under ROC curve). If we use the static inference to filter out the least likely links, and use the dynamic inference to rank the remaining links, the performance could be improved.

Another problem concerning this challenge is how to combine different datasets. There are two simple ways to do this. One is to do network inference for each dataset, and then combine the networks inferred together to generate the final network. Another way is to combine the datasets first, and then do a single network inference to generate the final network. The drawback of the first tactic is that if the datasets are too small, each inference is unlikely to generate meaningful result. While the shortcoming of the second tactic is that it tends to miss interactions that exist only in some of the datasets. In order to solve this problem, we developed an approach which first generates a number of “pseudo datasets”, and next do inference on each pseudo dataset, and then combine the inferred networks together.

In addition, we also applied our newly developed network inference method, BMALR (Bayesian Model Averaging for Linear Regression), on the *in silico* dataset. BMALR applies a new close form solution for calculating the posterior probability of each edge, and it works well on DREAM4 and DREAM5 *in silico* data.

Methods

Experimental network inference challenge

In the first step, we generated a scored draft network (denoted as Net_draft) for each cell line using an approach that is based on time-lagged mutual information. We first obtained a network (denoted as Net_tlCLR) based on time-lagged CLR (context likelihood relatedness) [1-2]. In addition, we generated a second network (denoted as Net_tlMIg) based on the time-lagged mutual information inference. We took around 5 time series as a group, and selected 16 different groups. For each group, a time-lagged mutual information inference was applied. The ranked scores in all the groups are summed up to generate a score network, Net_tlMIg. Finally, we generated a draft network (Net_draft) by assigning the score of each link as the maximum score of the link in the ranked Net_tlCLR and ranked Net_tlMIg.

In the second step, we generated a stimulus network (denoted as Net_stimu_stMI) for each stimulus using static mutual information inference method, which assumes the data at the time points of all the time series are in steady state.

In the last step, we generated the final network matrix by summing up the ranked draft network (Net_draft) and the ranked stimulus network (Net_stimu_stMI). The Net_draft was assigned with higher weight than Net_stimu_stMI.

In silico network inference challenge
In the first step, we generated a network by dynamic learning (denoted as Net_dyn). To do this, we obtained a network (denoted as Net_tlMI) using time-lagged mutual information. In addition, we generated another network (denoted as Net_dynBMALR) using Bayesian model averaging for linear regression, which was developed by our team before this challenge. Finally, we generated the dynamic learned network (Net_dyn) by assigning the score of each link as the maximum score of the link in the ranked Net_tlMI and ranked Net_dynBMALR.

In the second step, we generated a network for filtration (denoted as Net_filtr). Here, we obtained a network (denoted as Net_stMI) by static mutual information inference method, which assumes the data at the time points of all the time series are in steady state. In addition, we generated another network (denoted at Net_MIg) based on both static and time-lagged mutual information inference. In the generation of NetMIg network, we took around 5 time series as a group, and selected 16 different groups. For each group, static mutual information inference and time-lagged mutual information inference were applied. The ranked scores (including both static inference and time-lagged inference) in all the groups are summed up to generate a score network, which is called "Net_MIg". Finally, we generated the network for filtration (Net_filtr) by assigning the score of each link as the maximum score of the link in the ranked Net_stMI and ranked Net_MIg.

In the last step, we generated the final network matrix in the following way. Based on the scores in Net_dyn and Net_filtr, we used a cut-off to define "strong links" and "weak links". Net_dyn assigns the scores of “strong links”. Net_filtr ranks the “weak links”, and the new scores of the "weak links" are ensured to be smaller than the scores of "strong links".

Discussion
There are two crucial problems for improving network inference. One is how to combine the results from different methods, and the other is how to combine the results from different datasets. In this challenge (including Leaderboard phase), we tried several combinations of different methods. Actually, our final submission turned out to be worse than our best attempt in the Leaderboard phase. For the experimental challenge, the difference between our best attempt in the Leaderboard phase and our final submission is that in our best attempt we did not use time-lagged CLR, but only use the group-based time-lagged mutual information. This indicates that the arbitrary combination tends to compromise the results, and can make the final performance worse. For the in silico challenge, our best attempt uses only time-lagged CLR. Our guess is that this *in silico* data is so non-linear that it is unfavourable to linear regression based method, including our BMALR.

References

1. Greenfield, A., et al., DREAM4: Combining genetic and dynamic information to identify biological networks and dynamical models. *PloS one*, 2010. **5**(10): p. e13397.
2. Faith, J.J., et al., Large-scale mapping and validation of Escherichia coli transcriptional regulation from a compendium of expression profiles. *PLoS biology*, 2007. **5**(1): p. e8.

SC1 Network Inference: Team13

Summary

A generalized outranking approach to aggregate data-driven predictions from multiple network inference methods with prior biological interaction data.

Introduction
With our ensemble-based network inference we aim at a more reliable gene regulatory network prediction by (i) profiting from existing knowledge and (ii) compensating the limitations associated with single inference methods while benefiting from their individual strengths. In contrast to existing ensemble-based network inference methods (Marbach et al., 2012, Hase et al. 2013) we here (i) incorporate information from the STRING database in addition to the data-driven predictions, and (ii) use a generalized outranking approach, based on Farah and Vanderpooten (2007), to aggregate primary rankings.

Methods
General approach
Our approach, illustrated in Figure 1, can be divided into two parts: 1) A variety of existing network inference methods, including correlation-based, mutual information-based, and regression-based approaches, were used to obtain data-driven network predictions (*i.e.* primary rankings). We incorporated predictions of GENIE3, TIGRESS, CLR, MRNETB, mutual information, partial correlation, and Pearson correlation *(*Marbach et al. 2012*).* For the latter three we performed a subsequent column-wise Z-score scaling to favor certain directionality of the edges. We additionally constructed a knowledge-driven primary ranking of known protein interactions to incorporate prior biological knowledge into our network inference. This ranking was based on a score combining all types of evidence obtained from the STRING database (Franceschini et al. 2013). 2) The primary rankings were aggregated to obtain a final ranking. We used a generalization of the outranking approach described in Farah and Vanderpooten (2007) to determine the top 500 edges in the final ranking. The remaining positions were populated by a weighted Borda count. For the *in silico* network inference challenge (Part B) the knowledge-driven primary rankings were excluded from the aggregation.

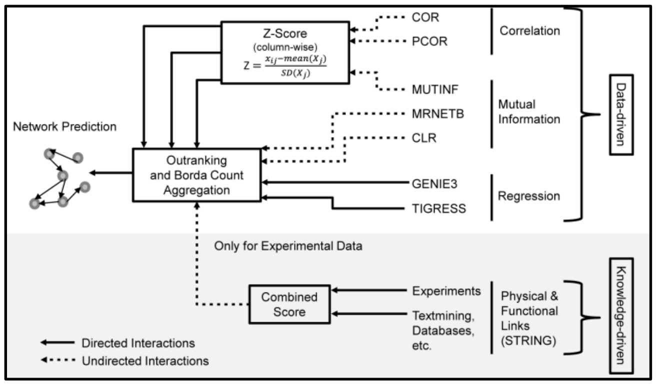


Figure 1: Unsupervised ensemble-based network inference incorporating prior knowledge

Implementations and parameter settings
We integrated all data given in the “Main” datasets without any normalization, log-transformation, or discretization and analyzed it as one sample. The data-driven predictions of existing network inference methods were obtained from R implementations of the respective algorithms with default parameter settings. The partial correlation was calculated with the “corpcor” package. MRNETB and CLR were run using the “minet” package. Predictions of GENIE3 and TIGRESS were run by utilizing R as a GenePattern client. The outranking and Borda count approach were implemented in Python. The parameters for the outranking (Farah and Vanderpooten 2007) were set as s_p=1 (preference threshold), s_v=4 (veto threshold), c_{min}=3 (minimal concordance coalition size) and d_{max}=2 (maximal discordance coalition size).

The ranking weights *w* = *(w*$_{agreement}$, *w*$_{disagreement}$, *w*$_{Borda)}$ were set to (1,1,1) and (3,1,2) for data-driven primary rankings and knowledge-driven primary ranking, respectively.

References

1. Farah, Mohamed, and Daniel Vanderpooten. An outranking approach for rank aggregation in information retrieval. *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2007.
2. Marbach, Daniel, et al. Wisdom of crowds for robust gene network inference. *Nature Methods* (2012).
3. Franceschini, Andrea, et al. STRING v9. 1: protein-protein interaction networks, with increased coverage and integration. *Nucleic Acids Research* 41.D1 (2013): D808-D815.
4. Hase T. et al. Harnessing Diversity towards the Reconstructing of Large Scale Gene Regulatory Networks. *PLoS computational biology* 9.11 (2013): e1003361

SC1 Network Inference: Team14

Summary

Inferelator, an ODE-based method with model selection, is combined with prior information to infer network structure.

Introduction

We used our method called the Inferelator [Greenfield et al., 2013], which was originally developed to learn the structure of global gene regulatory networks. In order to build a mechanistic model of phosphorylation, we create time-lagged response and design variables, such that the abundance of a protein is time-lagged with respect to the abundance of potential targets.

Methods

For both sub-challenges, we used the latest version of the Inferelator [Greenfield et al., 2013], a method developed to learn the structure of global gene regulatory networks. The method was slightly modified to take as input the phosphoprotein dataset and manually curated signaling priors. In order to build a mechanistic model of phosphorylation, we create time-lagged response and design variables, such that the abundance of a protein is time-lagged with respect to the abundance of potential targets. We use a bootstrapping approach and resample (with replacement) the response and design matrices, running model selection (Bayesian Best Subset Regression) for each resample. This procedure is repeated 50 times, generating an ensemble of networks, which we rank combine into one final network.

We used all cell lines, stimuli, time points and inhibitors, but split the data into 32 chunks. In sub-challenge A, for each of the 32 networks, we used only data from the specific cell line under the specific stimulus. We used only the 'Main' dataset. The only further data processing done was to average replicates. We browsed the KEGG website, read various literature, and did Google image searches to find known interactions among the proteins in the 'Main' dataset. That information was encoded in a matrix that was part of the Inferelator input.

The method is implemented in R and details are given in Greenfield et al. [2013]. The following modifications to the published method were implemented:

1. Support for negative priors – the structure prior can contain a -1 denoting that an edge is very unlikely to be in the final model. By default all edges that are facing the opposite direction of the positive priors are set to -1.

2. We have implemented a heuristic to remove potential shortcut errors from the consensus network. In the final network, if all three edges A->B, B->C and A->C are present, the last one is removed if the first two are jointly present in more bootstraps.

Our algorithm has the following tuning parameters:

1. τ, the parameter defining the time shift between design and response. This was set to 5 and 1 for sub-challenges A and B respectively.
2. The value of g, the weight given to the three types of interactions in the prior. The three types of prior interactions are: -1 (should not be in model), 0 (no prior information), 1 (should be in model). Based on the desired sparsity of the final networks, and our belief in the completeness of the network structure prior, we chose g = 0.01, 0.1, 1 for sub-challenge A, and g = 1, 1, 1, for sub-challenge B.

The actions of the inhibitors were modelled by setting the abundance of their targets to the minimum value in the entire input data set. Protein abundance is used as proxy for regulator activity.

Processing the provided data and generating results for both sub-challenges took 457 minutes CPU-time. On a Linux machine with an Intel Core i7-870 CPU (4 cores clocked at 2.93 GHz) this was equal to 79 minutes wall clock time.

References

1. Alex Greenfield, Christoph Hafemeister, and Richard Bonneau. Robust data-driven incorporation of prior knowledge into the inference of dynamic regulatory networks. *Bioinformatics*, 29(8):1060–7, April 2013.

SC1 Network Inference: Team16

Summary

An adaptive lasso regression model was used to infer connections between proteins.

Introduction

Inferring causal phosphoprotein signalling networks is very important to understand underlying biological mechanisms and identify key molecules that could be targeted therapeutically. Here we studied this problem based on the assumption that the abundance of individual phosphoproteins directly affects abundance of other phosphoproteins in a linear manner. To predict the abundance of the target phosphoprotein, we formulated this problem as a feature selection problem. In other words, for each given target phosphoprotein, we consider how to determine the subset of phosphoproteins that best predict target phosphoprotein abundance across the different conditions. To solve this problem, we have adopted an adaptive lasso regression model [1,2] to infer the regulatory network of each protein with respect to other proteins using time-course data with perturbations. The adaptive lasso regression method derives a necessary condition to confirm the selected variables to be consistent.

Methods

Data pre-processing

We used only the “Main” dataset in the following analysis. For the experimental data, we removed the data at zero time point due to too much missing data, and we combined all data of other time points together based on Formula (1). For the *in silico* data, there were 3 replicates for each of the 20 conditions under specific stimulus and treatments. We used a simple averaging approach to merge the replicates, and combined data from all time points based on Formula (1). Moreover, we scaled the data to make the abundance level of each phosphoprotein have zero mean and unit variance.

Adaptive Lasso regression model
Assume there are *K* phosphoproteins measured at *T* time points in *L* environments or conditions. Let

$$P^{-i} = (P^1, ..., P^{i-1}, P^{i+1}, ..., P^K),$$
$$P^i = (P^i(t_1^1), ..., P^i(t_T^1), P^i(t_1^2), ..., P^i(t_T^2), P^i(t_1^L), ..., P^i(t_T^L))^T, \quad (1)$$

which represents the expression of protein *i* in all *L* environments.

The problem can be formulated into the following format:

$$\min \sum_{i=1}^{k} \left\| P^i - P^{-i}\beta_i \right\|^2 + \lambda \sum_{i=1}^{k}\sum_{j=1}^{k-1} w_{ij} \cdot \left|\beta_{ij}\right|, \quad (2)$$
$$w_{ij} = \frac{1}{\left|\hat{\beta}_{ij}\right|}, \hat{\beta}_{ij} = \beta_{ij}(ols),$$
$$P^i : LT \times 1, P^{-i} : LT \times (k-1), \beta_i : (k-1) \times 1, \beta_i = (\beta_{1i}, \beta_{2i}, ..., \beta_{(k-1)i})^T,$$

where $\beta_{ij}(ols)$ is the ordinary least squares solution. We conducted the prediction for each phosphoprotein separately and predicted its direct regulators from all other phosphoproteins.

Algorithm
We used the R package 'msgps' to solve this problem (2). We applied the same method for inferring the underlying networks in the experimental and *in silico* sub-challenges, respectively.

Parameter selection
We chose the generalized cross validation (GCV) method to select the parameters in our model [3].

Discussion
Our approach was mainly based on the abundance level of a phosphoprotein affecting that of other phosphoproteins in a linear manner. It is simple, feasible, and applicable to even more complex situations. However, it doesn't consider non-linear relationships among molecules. Further investigation on this method and its variants is needed based on more biological knowledge and experimental observations.

References

1. Zou H. (2006) The adaptive lasso and its oracle properties. *Journal of the American Statistical Association*, 101(476): 1418-1429.
2. Huang J., Ma S., Zhang C.H. (2008) Adaptive Lasso for sparse high-dimensional regression models. *Statistica Sinica*, 18(4): 1603.
3. Hirose, K., Tateishi, S. and Konishi, S. (2011) Efficient algorithm to select tuning parameters in sparse regression modeling with regularization. *arXiv*:1109.2411v2.

SC1 Network Inference: Team17

Summary

The GENIE3 algorithm was used to infer weights and directions for regulatory edges between phosphoproteins, with incorporation of a prior knowledge network (this prior knowledge network was also submitted as a stand-alone prediction; see Team2 method).

Introduction

The GENIE3 algorithm (Huynh-Thu, 2010) was the top performer of the DREAM4 multifactorial sub-challenge, where it was used to infer a general regulatory network over a set of diverse perturbations. While the DREAM8 challenge presented a significantly more complex prediction task, with time series and inhibitor data available to predict causality. We reasoned that the random forest framework used by GENIE3 might allow it to perform well. To adapt this algorithm to the DREAM8 network inference sub-challenges, we combined data for all treatments and time points for each experimental condition into a single multifactorial data matrix that was then used as input to the GENIE3 algorithm. Compared with the canonical strategy of looking at only past time points to build predictors of a single node's activity, this strategy allowed us to maximize the density of training data available to the algorithm by using both past and future time point values for each regression task.

Methods

We used the GENIE3 algorithm to infer weights and directions for regulatory edges between antibodies. For each target protein, this algorithm uses a tree-based regression framework with the randomForest package in R (A. Liaw and M. Wiener, 2002) to build an ensemble of 1000 trees that explained the variance of the phosphoprotein with a set of possible regulators (other proteins in the trees). Each candidate regulator was scored according to the improvement in predictive power it provided when applied as an explanatory variable to a target protein's expression. Two 'variable importance' measures were provided by the randomForest package (A. Liaw and M. Wiener, 2002). One measure is the reduction in accuracy when a candidate is permuted within the dataset. The second measure is the decrease in the residual sum-of-squares for a particular candidate node. In both cases, an aggregate measure was obtained by averaging over all trees. In the former case, the withheld (out of bag) data was used to assess the accuracy before and after permuting each candidate regulatory node. Because of the lack of compelling evidence suggesting that the performance of one of these particular importance measures is superior for the DREAM8 dataset, we ran the algorithm twice, once for each measure, and averaged the scores from each run into a combined set of edge predictions.

Causal interactions between a putative regulator and a target are inferred from the variable importance scores given to each (regulator, target)-pair. For this study, rather than averaging the values of probes that map to multiple proteins, we considered each probe to be a distinct phosphoprotein, allowing the method to consider each active site individually. Each list of potential regulators for a given phosphoprotein was weighted by the variance reduction it provided (on variance-normalized data) and these lists were merged for all phosphoproteins with the GENIE3 method to produce a weighted list of predicted interactions.

For each experimental cell line/stimulus condition *l*, data for all treatments *i* and time points *t* were combined into a single multifactorial data matrix that was then used as input to the GENIE3 algorithm. The regression problem for each target probe *p* and cell line/stimulus condition *l* then reduced to minimizing the squared error loss over all treatment and timepoint pairs *(i,t)*:

$$\sum_{(i,t)} (x^p_{(i,t)} - f_p(x^{-p}_{(i,t)}))^2$$,

where $x^p(i,t)$ is the target phosphoprotein value for timepoint *t* and treatment *i*, $x^{-p}(i,t)$ are the values for potential regulators at (*t,i*) (all phosphoproteins, excluding *p*), and $f_p(y)$ is the function of the data to be optimized by our regression model.

Regression trees solve this problem by recursively splitting the learning sample with binary tests, each based on one input variable (phosphoprotein) in x^{-p}, trying to reduce the variance of the output variable/phosphoprotein x^p (Huynh-Thu, 2010; A. Liaw and M. Wiener, 2002). Votes from an ensemble of trees, each built from a bootstrapped sample of the data, is then averaged to produce the final predictions.

For the final submissions for the 1A experimental challenge, predicted edge values were averaged with the values derived from the biological prior (see Team2 entry). Because we observed that the prior greatly improved the performance of this entry during the course of the contest, we used a 60:40 split (prior:Genie3) to weight the average. The final submission for 1B consisted of the Genie3 solution on its own without a contribution from the prior.

Discussion

Though not competitive with the top performing methods, the adapted GENIE3 approach performed well on this challenge and better than many methods specifically tailored for this time-series data. This supports the hypothesis that generic regression methods may outperform more specific models, particularly if they're able to use a greater amount of training data.

References

1. Huynh-Thu VA, Irrthum A, Wehenkel L, Geurts P (2010). Inferring Regulatory Networks from Expression Data Using Tree-Based Methods. *PLoS ONE* 5(9): e12776. doi:10.1371/journal.pone.0012776
2. Liaw A and Wiener M (2002). Classification and Regression by randomForest. *R News* 2(3), 18--22.
3. Paull EO, Carlin DE, Niepel M, Sorger PK, Haussler D and Stuart JM (2013). Discovering causal pathways linking genomic events to transcriptional states using Tied Diffusion Through Interacting Events (TieDIE). *Bioinformatics*: doi: 10.1093/bioinformatics/btt471
4. Vandin, F., et al. (2012) Discovery of mutated subnetworks associated with clinical data in cancer, Pacific Symposium on Biocomputing. *Pacific Symposium on Biocomputing*, 55-66.

SC1 Network Inference: Team18

Summary

Network inference is formulated as a linear programming problem designed specifically for perturbation time series data, and models signaling as information flow.

Introduction

Our model lpNet-dyn is based on linear programming, designed for perturbation time-series data. It assumes the signal to be an information flow that is propagated along the network, starting at the source nodes and ending at the sink nodes. It can be seen as an inverse max-flow/min-cut problem, as we aim to reconstruct the underlying network given the flow of the network after observing different cuts (perturbations) [1]. The formulation of the problem as an LP in combination with the discretization of the data allows for heuristic inference of quasi-optimal networks in quasi-polynomial time.

Methods

Our method is based on linear programming, and was designed to take advantage of perturbation time-series data. It is an extension of the lpNet model previously developed by Knapp and Kaderali [1], which used perturbation steady-state data for the inference of signaling networks.

In lpNet-dyn we assume the signal to propagate through the network as an information flow. It starts at the source nodes and ends at the sink nodes. The signal can be positive (activating) or negative (deactivating) and its propagation is interrupted by perturbations such as knock-down, knock-out, or overexpression experiments.

Consider a network with n nodes, K perturbation experiments, and the values of each node are measured at T different time points. Then $x_{ikt} \in \mathbb{R}_0^+ \forall\{i,k,t\}$ denotes the value of node $i \in \{1,\dots,n\}$ in perturbation experiment $k \in \{1,\dots,K\}$ at time point $t \in \{1,\dots,T\}$. These values are stored in an observation matrix X. We consider a node to be inactive if its value x_{ikt} is less than its threshold δ_i and consider it to be active if $x_{ikt} \geq \delta_i$. The value of δ_i is defined by the user, usually with consideration of the data.

To specify whether a node was silenced in a given perturbation experiment we use an activation matrix $B \in \{0,1\}$, where $b_{ik} = 0$ if node *i* has been silenced in perturbation experiment *k*, and $b_{ik} = 1$ otherwise.

The network is modeled as a graph $G = (V, W)$, where the nodes v_i represent the proteins, and the edges $w_{ij} \in W$ with $w_{ij} \in \mathbb{R}_0^+ \forall\{i,j\}$ represent the influence of node i over node j. If $w_{ij} > 0$ node i is activating node j, if $w_{ij} < 0$ node *i* inhibits node j, and if $w_{ij} = 0$ node i does not influence node j.

We assume the expression of a given protein to be the sum of its intrinsic activity, w_i^0, plus the sum of the incoming edges multiplied by the node values of the corresponding parents at the previous time point $t-1$:

$$w_i^0 + \sum_{j \neq i} w_{ji} x_{jkt-1}$$

We call this value the *activity* of a node/protein. Furthermore, we assume that only an active node, $x_{jkt-1} \geq \delta_j$, at *t*-1 can influence other nodes in the network at *t*.
Assuming the network is sparse we minimize the edge weights w_{ji} in the following Linear Programming (LP) problem:

$$min_{\{w_{ji}^+, w_{ji}^-, w_i^0, \xi_{lt}\}} \left(\sum_{i,j} \left(w_{ji}^+ + w_{ji}^- \right) + \sum_i w_i^0 + \frac{1}{\lambda} \sum_{lt} \xi_{lt} \right) \quad (1)$$

s.t.:

if $x_{ikt} \geq \delta_i \wedge b_{ik} = 1 \wedge x_{jkt-1} \geq \delta_j \wedge b_{jk} = 1$:

$$w_i^0 + \sum_{j \neq i} \left(w_{ji}^+ - w_{ji}^- \right) x_{jkt-1} \geq \delta_i \quad (2)$$

if $x_{ikt} < \delta_i \wedge b_{ik} = 1 \wedge x_{jkt-1} \geq \delta_j \wedge b_{jk} = 1$:

$$w_i^0 + \sum_{j \neq i} \left(w_{ji}^+ - w_{ji}^-\right) x_{jkt-1} \leq 0 + \xi_{lt} \tag{3}$$

if $x_{jkt-1} < \delta_j \vee b_{jk} = 0$:

$$x_{jkt-1} = 0 \tag{4}$$

if $i \in V \setminus S$:
$$\sum_{j \in V, j \neq i} \left(w_{ji}^+ + w_{ji}^-\right) \geq \delta_i \tag{5}$$

if $i \in V \setminus F$:
$$\sum_{j \in V, j \neq i} \left(w_{ji}^+ + w_{ji}^-\right) \geq \delta_i \tag{6}$$

$$w_{ji}^+, w_{ji}^-, w_i^0, \xi_{lt} \geq 0 \tag{7}$$

where the objective function 1 minimizes the sum of the absolute values of the edge weights $w_{ji} = w_{ji}^+ + w_{ji}^-$, baseline activities w_i^0, and slack variables $\xi_{lt} \in \mathbb{R}_0^+ \forall \{l, t\}$. The minimization is done while satisfying constraints 2-7, where constraint 7 is the intrinsic nonnegativity constraint in the LP formalism. Constraints 2-3 implement the basic assumption that if a protein i is active, its *activity* should be greater than the respective threshold δ_i, and if protein i is inactive then its *activity* should be zero. Still, a slack variable ξ_{lt} is inserted in constraint 3 to account for noise in the data, which can lead to incompatible constraints that render the LP problem unfeasible. By checking whether the parent node j is active and not silenced in constraint 4, we implement the assumption that only active parent nodes $x_{jkt-1} \geq \delta_j$ at time point $t-1$ can influence other nodes at t. The number of slack variables ξ_{lt} introduced is $= |\{x_{ikt} : x_{ikt} < \delta_i, \forall \{i, k, t\}\}|$, i.e., it is equal to the number of inactive nodes in the network. The introduction of slack variables is penalized in the objective function according to the value of parameter λ. The best value for λ is determined using a k-fold cross validation (CV). We calculate the range of possible values for λ going from 0 to $\Xi = L \times \sigma^2(x_{ikt})$, where $\sigma^2(x_{ikt})$ is the variance of the observations x_{ikt}, and Ξ is the worst case scenario, in which all the introduced slack variables are unequal to zero. We repeat the CV a given number of times and compute the Mean Squared Error (MSE) each time. We then use the set of networks corresponding to the λ value that results in the minimum MSE. To sum up the networks inferred in the different CV steps in a single network, we divide the number of times an edge $w_{ji} > 0$ or $w_{ji} < 0$ is inferred by the number of inferred networks, and set the edge sign accordingly. If an equal number of negative and positive edges are inferred, its value in the final network is set to 0.

Constraints 5 and 6 are a way to include prior knowledge about the network to be inferred. If the source nodes of the network, corresponding to receptor proteins in the cell membrane, are known in advance these can be included, and constraint 5 will force all other nodes to have at least one incoming connection. On the other hand, if the end nodes are known in advance - these can be, for instance, transcription factors - constraint 6 will force all other nodes to have at least one outgoing connection. For further details please refer to section 2.2-2.3 from [2].

The algorithm has 5 parameters, four of which are determined from the data and one (λ) whose value is estimated using cross validation. The latter has already been explained. The parameters to be determined from the data are: a) a threshold per node δ_i, which distinguishes its active state from its inactive state. This value is set as the value of the node at the first time point when no inhibitors are used; b) the average value of each node when it is active, which is calculated by taking the average value of all values of the node that are equal or greater than its threshold; c) the average value of each node when it is inactive, which is calculated by taking the average value of all values of the node that are less than its threshold; d) and e) the standard deviation values of each node when it is active and inactive, respectively. These values are usually a small percentage of the average values in b) and c).

Parameters b)-e) are only necessary to predict the values of removed entries from the observation matrix X in the cross validation step [1]. We used 50-fold cross validation. We randomly chose the entries that have been removed from the observation matrix, and we predict their values using a normal distribution with a very low standard deviation value (0.01).

We modeled the actions of the inhibitors by considering the targeted proteins to be silent, i.e., once the signal reaches these proteins it cannot be propagated through them and $b_{ik} = 0$. We do not model the stimuli actions.

The same methodologies for inferring the underlying networks in the experimental sub-challenge and the *in silico* sub-challenge were applied, except for a) the modeling of the stimuli actions, b) the definition of the threshold value δ_i, c) data normalization.

Computation time and implementation
The model was programmed in R, and it is now implemented as a Bioconductor package, lpNet (http://bioconductor.org/packages/release/bioc/html/lpNet.html). To solve the linear programming problem, we used the R package lpSolve v. 5.6.6 (http://cran.r-project.org/web/packages/lpSolve/index.html). We used a computation cluster (sun grid engine version 6.2u5) with an average node architecture equal to an Intel Xeon 2GHz, 4GB RAM running a 64bit Linux-based operating system for the experimental sub-challenge. Depending on the network to be inferred, i.e. cell line + stimulus, the algorithm took between 3 and 16 days to infer each network.

Data
For the experimental dataset we used only the main datasets, and for each combination cell line + stimulus we used only the data referring to that cell line and stimulus, considering all time points and inhibitors. We added two nodes, FGFR1 and FGFR3, whose values were set as NA, and normalized the data, i.e., for each cell line + stimulus, each node value was divided by the maximum value of that node across all perturbation experiments and all time points, after removing the outliers. For the experimental dataset we specified the source nodes as being the following source nodes: c-Met_pY1235, EGFR_pY1068, EGFR_pY1173, EGFR_pY992, HER2_pY1248, HER3_pY1298, FGFR1, FGFR3. The reason is that these proteins are known to be receptors in the cell's membrane.

Conclusion
Both during the algorithm development and the challenge we concluded that the value of the threshold that distinguishes an active node from an inactive one is fundamental for the method's performance. Plus, one way to change the method's execution time is to change k's value in the k-fold CV step, the higher k's value, the higher the execution time. However, k's value needs to be defined in proportion to the observation matrix dimension ($n \times K \times T$), since if it is too small a big percentage of the matrix entries are removed in the CV step and the results may not be reproducible between different executions of the model.

References

1. Knapp, Bettina and Kaderali, Lars. Reconstruction of cellular signal transduction networks using perturbation assays and linear programming. *PLoS ONE*, 8(7):e69220, 07 2013.
2. Matos, Marta. Network inference: extension of a linear programming model for time-series data. Master's thesis. University of Minho (2013). http://repositorium.sdum.uminho.pt/handle/1822/27867

SC1 Network Inference: Team19

Summary

Edges contained in prior knowledge networks are selected and ranked using Pearson correlation scores between protein pairs and feature weights from a gradient boosting regression approach.

Introduction

Our method takes the edges from a previously constructed kinase network (Newman et al.) and, for each edge, calculates Spearman correlation between protein abundances for the pair of proteins that constitute the edge. These correlation scores are then used to rank edges. Additional edges, contained in the BioGrid database, are added in to our inferred network, weighted using a gradient boosting regression approach; a machine learning approach that has been proven to accurately reverse-engineer transcriptional regulatory networks from expression data used in the DREAM3, DREAM4, and DREAM5 challenges

Methods

In the pre-processing phase we log-transformed the data for both SC1A and SC1B.

Experimental network inference challenge

For each cell line-stimulus pair we performed the following procedure:

In the first phase we created two groups of data: "all data" where we took all the data for all inhibitor conditions and "no inhibition data" where we only took the non-inhibited (DMSO) condition data. For both groups we calculated a Spearman correlation matrix for the proteins contained in the "full" dataset (i.e. we used the optional extra time points and antibodies for this analysis).

In the second phase we intersected the above correlation matrix with the Newman et al. kinase network. We added the highest scoring edges to our final solution:

a) edges from kinase network with p-value of Spearman correlation coefficient from "all data" less than or equal to 0.001 were ranked by their correlation coefficient and scaled to (1,0.85) range,
b) edges from kinase network with p-value of Spearman correlation coefficient from "all data" less than or equal to 0.005 were ranked by their correlation coefficient and scaled to (0.85-0.7) range,
c) edges from kinase network with p-value of Spearman correlation coefficient from "no inhibition data" less than or equal to 0.001 were ranked by their correlation coefficient and scaled to (0.7-0.6) range,

d) edges from kinase network with p-value of Spearman correlation coefficient from “no inhibition data” less than or equal to 0.01 were ranked by their correlation coefficient and scaled to (0.6-0.5) range,
e) edges from kinase network with p-value of Spearman correlation coefficient from “no inhibition data” less than or equal to 0.05 were ranked by their correlation coefficient and scaled to (0.5-0.4) range.

In the third phase we bounded the above edges ranked (1-0.4) to the proteins contained in the “main” dataset (i.e. phosphoproteins). We only allowed edges that:

a) had a direct link in the “main” set,
b) had a 1-hop link in the “full” set, but no 1-hop link in the “main” set.

In the last phase we added the edges from a broader set of interactions reported in BioGrid database. We took “all data” from “main” set of proteins. We limited the possible edges to the ones found in Newman et al. kinase network and BioGrid (i.e. we limited the search domain to the edges known from the literature, but at possibly low confidence). We reasoned that these edges need a high support from the data in order to compensate for their low confidence in the literature. In order to assess how well these edges were supported in data we decomposed the network inference problem into a set of variable selection problems (one for each node). We solved each of them using the relative importance of features from the ENNET gradient boosting regression procedure. We used the default values of parameters of the algorithm. We scaled the edges to (0.4-0) range.

In silico network inference challenge
We took all data available. We decomposed the network inference problem into a set of variable selection problems (one for each node). We solved each of them using the relative importance of features from the ENNET gradient boosting regression procedure. We used the default values of parameters of the algorithm. We scaled the edges to (1-0) range.

Discussion

We learned that the machine learning-based methods must always be supported not only by the experimental data but also by existing biological knowledge in the network inference problem. In three distinct rounds we tried the following approaches for SC1A: report only the edges from the Newman et al. kinase network and BioGrid network, report only the edges indicated by the machine learning algorithm without use of any prior information (like in SC1B), and report the edges combined in the first and the second approach. Out of these three, the last approach gave the best result. However, we would need to analyze the gold standard networks to assess which interactions were especially easy/hard to find by our method.

References

1. Newman, R. H. *et al*. Construction of human activity-based phosphorylation networks. *Molecular Systems Biology* 9.1 (2013).
2. Stark, C. *et al*. BioGRID: a general repository for interaction datasets. *Nucleic Acids Research* 34.suppl 1: D535-D539 (2006).
3. Slawek, J. and Arodz, T. ENNET: inferring large gene regulatory networks from expression data using gradient boosting. *BMC Systems Biology* 7:106 (2013).
4. Slawek, J. and Arodz, T.. ADANET: Inferring Gene Regulatory Networks using Ensemble Classifiers. In *Proceedings of the ACM Conference on Bioinformatics, Computational Biology and Biomedicine*, 434-441. (ACM, 2012).

SC1 Network Inference: Team21

Summary

Random forests were combined with mutual information to yield protein networks that were integrated with prior literature and logic-based domain knowledge.

Introduction

The motivation behind our ensemble approach came from our previous work with protein array data, as well as the premise of ‘wisdom of crowds.’ Inspired by the latter, each member of the team initially chose different subclasses of algorithms to focus on. General classes that our computational methods fall under include Ensemble, Regression, and Correlation. While we first used only the unsupervised network inference methods, we soon realized that they were insufficient in capturing the data’s structure and temporal changes. Innovation stemmed from integrating the unsupervised approaches with a logic-based method dependent on the relative time-courses of each protein, which was informed by the inhibition/stimulation conditions. We combined networks resulting from the logic-based approach with existing literature knowledge to define a set of prior-knowledge networks. A second innovation in this approach was how we obtained our literature knowledge, which was identified both manually and by mining publically available databases.

Methods

We used all parts of the data for the experimental sub-challenge including the “Full” dataset, except for its confidence scores. Pre-processing consisted of replacing duplicate data points by their average. Missing data were not replaced or approximated.

We used ensemble, regression, and correlation techniques for modelling. We initially used unsupervised network inference methods. We employed the context likelihood of relatedness algorithm (Faith et al. 2007) in Matlab and the Random Forest algorithm in R.

In our final iteration, we integrated the unsupervised approaches with logic-derived networks and available literature data. The logic-based knowledge, obtained from a method we designed ourselves, depended on the relative time-courses of each protein, informed in part by the inhibition/stimulation conditions.

To supplement the data provided in the challenge, we searched the worldwide web both automatically and manually in order to find relations between proteins in the experimental data set. We wrote scripts in Python and Java to mine protein data from Kegg and STRING (string-db.org). We also relied on general web searches and manual MIMI (mimi.ncbi.org) queries for the proteins listed in the experimental data.

Our final models were created by combining, with equal weight, networks derived from random forest and mutual information algorithms, with networks based on prior knowledge obtained through the literature searches and those derived from our logic-based approach.

Discussion

Our approach to generating protein signaling networks integrated information from prior knowledge, i.e., interactions reported in literature, with a theoretical analysis based on both Random Forest and Mutual Information. In combining multiple theoretical methods, we hoped to incorporate some of the advantages inherent in the “wisdom of crowds” (WOC) approach elucidated in DREAM 5. However, we found that RF alone (score = 2.24) outperformed our integrated approach using both RF and MI (score = 1.84, final submission) on the simulated data (SC1B). It is possible the dynamic nature of this time-series data, in contrast to the DREAM 5 data, may make the effectiveness of WOC sensitive to the method of integration. This may be an interesting topic for future research. In addition, we found the theoretical component alone, without prior knowledge, performed well on the simulated data (SC1B) but was not sufficient to accurately predict networks derived from the experimental data set (SC1A). This highlights the importance of incorporating prior knowledge into the analysis of experimental protein networks.

References

1. Faith JJ, Hayete B, Thaden JT, Mogno I, Wierzbowski J, Cottarel G, Kasif S, Collins JJ, Gardner TS (2007) Large-scale mapping and validation of Escherichia coli transcriptional regulation from a compendium of expression profiles. *PLoS Biol.* **5**:e8.
2. Ho, Tin Kam (1998) "The Random Subspace Method for Constructing Decision Forests". IEEE Transactions on Pattern Analysis and Machine Intelligence **20**: 832–844.
3. Marbach D, Costello JC, Küffner R, Vega NM, Prill RJ, Camacho DM, Allison KR (2012) DREAM5 Consortium, Kellis M, Collins JJ, Stolovitzky G. Wisdom of crowds for robust gene network inference. *Nat Methods* **9**: 796-804.
4. Watkinson J, Liang KC, Wang X, Zheng T, Anastassiou D (2009) Inference of regulatory gene interactions from expression data using three-way mutual information. *Ann N Y Acad Sci* **1158**: 302-313.

SC1 Network Inference: Team22

Summary

Networks are inferred by combining prior information with the output of a lasso regression model, adopted to capture the time-varying effect of protein phosphorylation between adjacent time points.

Introduction

How to infer causal phosphoprotein signalling networks is a challenging task for understanding complex regulatory mechanism in key biological processes. Temporal analysis of time-series data can provide insights into the underlying mechanism of the biological processes. How to consider the effect of time changes on network topology will be valuable to infer the underlying networks. Another effect is that the underlying systems should be well designed and simplified, thus how to infer the sparse structure will be important to infer the networks. We adopted a standard lasso regression model for achieving this task, which was inspired by a very recent study [1].

Methods

Given K proteins with T time frames, we can obtain a K-by-T data matrix $P_{K\times T}$. We first considered analyzing the temporal causal relationships among the K proteins. We adopted the following matrix equation to reflect the causal relationships:

$$P_{t+1} = WP_t$$

where P_t denotes the t-th column of P, and $W_{K\times K}$ is the time-varying coefficient matrix and $W_{i,j}$ depicts how the protein j at time t affects the abundance level of protein i at time t+1. A positive $W_{i,j}$ indicates activation, others it means depression or inhibition. We set the diagonal line of W to zero to remove the self-regulation effects. Accordingly, the assumed lineal regulatory relationships between proteins i at time t+1 and all other proteins at time t could be expressed as:

$$P_{i,t+1} = w_{i,1}P_{1,t} + w_{i,2}P_{2,t} + \cdots + w_{i,i-1}P_{i-1,t} + w_{i,i+1}P_{i+1,t} + \cdots + w_{i,\mathrm{k}}P_{\mathrm{k},t}$$

Mathematically, we can formulate this problem into the following optimization model by considering the L_1-norm penalty:

$$\min \sum_{t=1}^{T-1} \left\| P_{t+1} - WP_t \right\|^2 + \lambda \left\| W \right\|_1$$

The matrix *W* uniquely specifies a regulation network of *K* proteins, where each node is a protein and the edge weights indicate the interaction between each pair of nodes. In addition, the network structure should be sparse. To this end, we added a L_1-norm regularization penalty to ensure the sparsity of the network.

Finally, we can obtain the matrix *W*, where $W_{i,j}$ denotes a directed edge pointing from protein *j* to protein *i*, and the value of $W_{i,j}$ indicates the intensity of their regulatory relationship.

We only used the “Main” dataset and did not employ the extra antibodies and time points in the “Full” dataset. For in silico data, there were 3 copies of each of the 20 conditions under specific stimulus and treatments. We combined all this time-varying profiles together to expand the amount of available data. For the 32 specified networks from the experimental data, we did the same thing with the 4 different treatments. Missing data were excluded from analysis. We constructed the causal relationships by analysing two contiguous timepoints from the same profile. We did log transformation of the in silico data before further calculation. For the experimental sub-challenge, we used the original data. The optimization model can be formulated as a standard lasso model and many advanced solvers can be available. We chose the build-in package of lasso in Matlab 2013a version for calculation.

For the experimental data, we extracted some rough information of the pathways of the proteins from the KEGG dataset and added these prior edges to 32 predicted networks. For example, we noticed an inhibition relationship from AKT to p27 in the PI3K-Akt signalling pathway. Thus we added edges of inhibition from 'AKT_pS473' and 'AKT_pT308' to 'p27_pT157' and 'p27_pT198'.

We did not make use of the network models learned in sub-challenge 1 to sub-challenge 2. The classical parameter selection criteria (e.g. BIC) produced unsatisfactory network results that are either too sparse or lack generic properties of biological networks. Therefore, we conducted a selection to guarantee small MSE and proper network density.

Discussion

We assumed there is a linear temporal regulatory relationship between the phosphorylated proteins and formulated this problem into a standard lasso regression model for inferring signal transduction networks. More detailed investigations to the data generation and underlying biological principles would contribute to improve the performance of this method.

References

1. Zhang, K., Han, J., Groesser, T., Fontenay, G., & Parvin, B. (2012). Inference of Causal Networks from Time-Varying Transcriptome Data via Sparse Coding. *PLoS One*, 7(8), e42306.

SC1 Network Inference: Team24

Summary sentence

Bayesian network structure learning with Hill Climbing (experimental data) or Max-Min Hill Climbing (*in silico* data), combined with model averaging to identify weighted networks.

Introduction

We hypothesize that causal signaling networks can be modeled using causal Bayesian networks. Bayesian networks are increasingly being used for uncertain reasoning and machine learning in many domains, including biomedical informatics [1-3]. In the modeled Bayesian network, each node represents a protein. A benefit of Bayesian networks is that they represent probabilistic relationships, and therefore they can manage the perturbations in biological data. A second strength is that they can model the natural causal relationships in biology. A Bayesian network (BN) consists of (i) a Directed Acyclic Graph (DAG), where the node set consists of random variables and the edges represent relationships among the random variables, and (ii) a conditional probability distribution for each node given its parent nodes.
Here we apply Bayesian network structure learning with model averaging for inferring the underlying networks in the experimental sub-challenge and the *in silico* sub-challenge.

Methods

The concept behind Bayesian network structure learning is to select a unique DAG model from input data. Two types of structure learning algorithm, constraint-based and score-based algorithms [4], have been developed for Bayesian model selection. The score-based Hill-Climbing (HC) algorithm was used for the experimental sub-challenge and the Max-Min Hill-Climbing (MMHC) algorithm was used for the *in* silico sub-challenge. HC is a greedy search on the space of the directed graphs. The optimized implementation uses score caching, score decomposability and score equivalence to reduce the number of duplicated tests [4]. MMHC is a hybrid algorithm that combines the constraint-based Max-Min Parents and Children algorithm to restrict the search space and the Hill-Climbing algorithm to find the optimal network structure in the restricted space [4].

We implemented Bayesian network model averaging to average over different possible DAG models to obtain the final strength of the edges in the network. When averaging the models, we did not include edges with small weights in the averaged network, to ensure that the final network would contain only significant edges.

The key steps of the method follow. First, to construct the Bayesian network, the method computes the strength of all possible edges based on the HC or MMHC algorithm, which was learned from bootstrapped data. For the experimental sub-challenge, the data are the "main" data for each cell-line/stimulus context, and networks for each context were learned independently of each other. For the *in silico* sub-challenge we used the entire dataset. Second, the resultant Bayesian networks of the first step are averaged to get the averaged network. Finally, the strength of the edges present in the averaged network are formatted as SIF and EDA files representing the final causal network.

We implemented this method in R with the package "bnlearn" (developed by Scutari) [4, 5]. "bnlearn" is an R package for learning the graphical structure of Bayesian networks, estimating their parameters and performing inference. This package has implemented the HC and MMHC algorithms, which can be directly used to build the Bayesian network structure and calculate the strengths of the edges in the network. We also used the "averaged.network" in the package for model averaging.

Our method validation is based on the AUROC scores and rankings from the leader board. The algorithm selection of each part of the sub-challenges is also based on the results from the leader board. For example, we have compared different learning algorithms (e.g., grow-shrink and incremental association learning algorithms [4]) in the "bnlearn" R package for the experimental and *in silico* data, and finally we selected HC for experimental data and MMHC for *in silico* data.

Discussion
Bayesian network structure learning has been employed to learn the protein causal network structure from the provided experimental and *in silico* data. Bayesian model averaging was used to get the final strength of the edges in the protein causal network. The approach described here could be expanded to include prior network knowledge, as well as information about the action of inhibitors on particular proteins.

References
1. Kjaerulff UB, Madsen AL. *Bayesian Networks and Influence Diagrams*. NY, NY: Springer; 2010.
2. Neapolitan RE. *Probabilistic Reasoning in Bioinformatics*. Burlington, MA: Morgan Kaufmann; 2009.
3. Jiang X, RE Neapolitan. Mining strict epistatic interactions from high-dimensional datasets: ameliorating the curse of dimensionality; *PLoS ONE* 2012; 7(10):e46771. doi:10.1371/journal.pone.0046771.
4. Scutari M, Learning Bayesian Networks with the bnlearn R Package; *Journal of Statistical Software* 2010; 35(3):1-22. URL: http://www.bnlearn.com/
5. http://www.R-project.org/.

SC1 Network Inference: Team25

Summary
Ensemble network inference approach that incorporated a prior knowledge network with results from L1-penalized Granger causality, GENIE3, and ARACNE (the prior knowledge network was also submitted as a stand-alone prediction; see Team2 method).

Data preprocessing
The competition-provided data was aggregated by taking the median value across replicates and inhibitors, resulting in matrices of proteins (rows) by time points (columns), for each cell line and stimulus pair. These matrices will be referred to as the protein time series matrices.

Experimental network inference sub-challenge
The submission was the result of applying an ensemble that combined multiple inferred protein interaction networks generated by multiple methods. The interaction network was encoded as the square adjacency matrix representing a graph of pairwise interactions (edges) between proteins (nodes). The methods incorporated into the ensemble included L1-penalized Granger causality, GENIE3, ARACNE, and the biological prior (see respective summaries for Team1, Team2, and Team17). Each of these methods produces an inferred interaction matrix. The scores in each matrix were normalized by linearly scaling so that the values obtained were between 0 and 1 before combining. Note this formulation makes no distinction between activating and inhibiting interactions. The ensemble computed the median of the interaction

values (for each pair of proteins) across the 4 methods, to generate the final combined network with scores given by:

$$Score_{Ensemble}(A,B) = Median\{Score_{LASSO-Granger}(A,B), Score_{GENIE3}(A,B), Score_{ARACNE}(A,B), Score_{Biological\ Prior}(A,B)\}$$

where (A,B) represents a pair of proteins.

In silico network inference sub-challenge

This submission was an ensemble created identically as was done for the experimental network inference sub-challenge except that no biological prior was used as part of the ensemble. This ensemble computed the median of the individual methods: LASSO-Granger, GENIE3, and ARACNE.

ARACNE gene network reconstruction

ARACNE (Margolin et al., 2006) was used to generate the symmetric adjacency matrix for the inferred network. ARACNE infers pairwise gene interactions using mutual information and then uses the information-theoretic data processing inequality to restrict to direct interactions. The Mutual Information (MI) between two proteins is estimated using a Gaussian kernel estimator:

$$I(\{x_i\},\{y_i\}) = \frac{1}{M}\sum_i \log \frac{f(x_i, y_i)}{f(x_i)f(y_i)},$$

where x_i, y_i are a series of measurements for two proteins (x and y), with i iterating over the M samples, and f is a Gaussian kernel estimator. The kernel estimator can approximate the joint probability distribution of two variables, $z_i = \{x_i, y_i\}$:

$$f(\vec{z}) = 1/M \sum_i h^{-2} G\left(h^{-1}|\vec{z} - \vec{z}_i|\right),$$

where G is the standard bivariate Normal density function, and h is the kernel width.

We downloaded ARACNE from http://wiki.c2b2.columbia.edu/califanolab/index.php/Software/ARACNE. ARACNE takes in a matrix of genes (rows) by experimental conditions (columns), and outputs a list of inferred interaction values for pairs of genes. To obtain directionality of causal links, we concatenated the proteins-by-time-points matrix with a time-lagged version of itself. Specifically, we row-concatenated the [15min, 4hr] portion of the matrix with the one corresponding to [5min, 2hr]. Application of ARACNE to the time-lagged version of the matrix now allowed to detect dependency between proteins at different time points.

We ran ARACNE on the time-lagged data to obtain scores for forward and backward temporal correlations between proteins in a time series. For each pair of interacting proteins, we kept the directed temporal edge with the largest magnitude (absolute value). Specifically, if $\mathrm{Score}(\mathrm{A_{original}}, \mathrm{B_{lagged}}) > \mathrm{Score}(\mathrm{A_{lagged}}, \mathrm{B_{original}})$, we kept the causal link from A to B. Conversely, if $\mathrm{Score}(\mathrm{A_{original}}, \mathrm{B_{lagged}}) < \mathrm{Score}(\mathrm{A_{lagged}}, \mathrm{B_{original}})$, then we interpret that it is more likely that B is a cause of A.

LASSO-Granger: Granger Causality was applied as in the Prophetic Granger submission by Team1, except that here only time points prior to the variable being regressed were used.
GENIE3: GENIE3 was applied as in the entry for Team17.
Biological prior: The biological prior was obtained as described in the entry for Team2.

References

1. Margolin, Adam A., et al. ARACNE: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. *BMC Bioinformatics* 7.Suppl 1 (2006): S7.

SC1 Network Inference: Team26

Summary

A protein signaling network is inferred by comparing phosphoprotein abundance levels between intervention and no-intervention treatments.

Introduction

This approach is motivated by the working definition of a ‘causal edge' used in this challenge. Specifically, the idea is that a causal edge might be detected by monitoring the change in the child node when the parent node is either intervened or not intervened. Causal edges between nodes can then be predicted by collecting and comparing the protein expression of each node in each treatment (intervention / no-intervention) over time. Finally, these causal edges are scored according to the differences in protein expression, and those above a set threshold are included in the final network.

Methods

We read the training datasets, containing normalized protein abundance measurements for *in silico* and experimental data, into R as data lists. We used the provided *in silico* relationship between stimuli, inhibitors, and their target proteins from the Synapse Wiki to create another R data list. For experimental data we obtained a prior network of relationships from the literature (Rhodes, 2008) (Gilmartin, 2011) (Pardo, 2009). We imputed missing data points in the experimental data set by averaging neighboring data points. We did not perform imputation for cases where missing data were at the first or the last position in the time series, or there was more than one data point absent in a row. Replicates in experimental data were averaged.

We defined parent network nodes as nodes targeted by the treatments. We then used time series data for comparison between stimulus plus inhibitor (intervention) conditions with stimulus only (no intervention) conditions for all other nodes (possible child nodes), to obtain an Area Between Curves (ABC) by integration over all time points. If the parent node was treated with a known inhibitor (less parent) and the ABC of intervention for the child was positive (more child) the relation between the parent and child was defined as an inhibition and if the ABC of intervention for the child was negative, the relation between the nodes was defined as an activation. A normalized ABC for each child was calculated as the absolute value of the fraction of its value and the maximum change ABC occurring in the whole network leading to inhibition and activation scores between 0 and 1. In the final network only those edges were included which scored higher than 0.1. This arbitrary threshold score was decided upon based on observation of the expression graphs plotted for each of the two conditions for a larger number of parent-child edges. Thresholds greater than 0.1 appeared to describe visually different curves.

This method was implemented in R using the “simp” function from the StreamMetabolism R package (Sefick Jr, 2013). All R scripts are available from GitHub at: https://github.com/gungorbudak/netinf-bigcat/

Discussion

In comparison to the other approaches used, this approach has ranked averagely for *in silico* data and low for experimental data in the leaderboard. Here we propose possible improvements to the algorithm. First, we did not check for actual treatment effects on the parents, and instead assumed parents responded as expected. Looking at actual changes might improve the analysis, especially for the experimental data. Second, the statistical evaluation of the time series data could be improved with a test that uses individual time points instead of a threshold value for the whole ABC, or one that uses repeat values of measurements to calculate the ABC repeatedly. Third, we did not evaluate indirect effects. We could in principle extend the approach to either allow any changed node to be a parent for any other (but that would be computationally hard) or only allow parent child relationships that are already known for biological databases like IntAct, or combine the two by allowing primary parent child relationships to be newly found while indirect relationships would only be allowed for known interactions. Finally, the network obtained could be further enriched, with more relations between nodes, by assigning antibodies found to inhibit others as intervening treatment and antibodies found to activate others as non-intervening treatment and repeating the process. This might also result in the discovery of novel relations to be included in the final network.

References

1. Gilmartin, A. G. (2011). GSK1120212 (JTP-74057) is an inhibitor of MEK activity and activation with favorable pharmacokinetic properties for sustained in vivo pathway inhibition. *Clinical cancer research* , 989-1000.
2. Pardo, O. E. (2009). The fibroblast growth factor receptor inhibitor PD173074 blocks small cell lung cancer growth in vitro and in vivo. *Cancer research*, 8645-8651.
3. Rhodes, N. (2008). Characterization of an Akt kinase inhibitor with potent pharmacodynamic and antitumor activity. *Cancer research* , 2366-2374.
4. Sefick Jr, S. A. (2013, 13-November). StreamMetabolism: Stream Metabolism-A package for calculating single station metabolism from diurnal Oxygen curves. From CRAN Package: http://cran.r-project.org/web/packages/StreamMetabolism/index.html

SC1 Network Inference: Team28

Summary

A novel network inference method using Partial Least Squares Regression (PLSR) with the Variable Influence on Projection (VIP) score as a measure of edge confidence

Introduction

Many existing network inference algorithms are computationally expensive and time-consuming. This inefficiency limits the number of nodes that can be feasibly incorporated into a network model. We therefore selected a regression-based approach, which offers simplicity, speed, and scalability. We developed a novel network inference method using Partial Least Squares Regression (PLSR)[1] with the Variable Influence on Projection (VIP) score[2] as a measure of edge confidence. PLSR has been found to give highly accurate results for collinear and complex inputs[1], which are inherent in signaling networks. To our knowledge, this is the first application of PLS-VIP for network inference. We also took the novel approach of combining a static (data explained by the same timepoint) and dynamic network (data explained by the preceding timepoint), to form a consensus model. We deemed this algorithm Dynamic Inference of NEtworks Using Singular values (DIONESUS).

Methods

In order to make an informative comparison among relationships of phosphoproteins, we normalized each sample to an internal reference from the same dataset. In the experimental network inference challenge (SC1A), the inhibitor was added to the cell media before the growth factor, so we considered an additional time point at -1 min to account for the differential effect of each perturbation. All concentrations of phosphorylation for the -1 min time point were assumed to be the same as concentrations in the DMSO treated condition at time point 0. The log2 fold change (L2FC) was calculated with respect to the -1 min time point. As a result, all L2FCs at -1 min were effectively set to zero values and other input data points reflected the corresponding log change above or below this assumed steady state. In the companion *in silico* challenge (SC1B), all L2FCs were made in reference to the first time point (L2FC = 0 at 0 min) for each phosphoprotein. The arithmetic means of the L2FCs were computed over the biological replicates and used as an input to the algorithm.

For SC1A, missing data were replaced with data for the DMSO-only perturbation at that time point. We hypothesized that translational, rather than post-translational, effects would dominate network dynamics at longer time-scales. Therefore, we did not include the 240 min data to infer edges representing information flow between phosphoproteins in the static network. The 240 min data was however included in the dynamic network model. In order to account for the variance added by the perturbations, dummy categorical variables were included for the presence (indicated with a 1) or absence (indicated with a 0) for each growth factor and each inhibitor. All time points were used in Part B.

Inference of Static and Dynamic Networks. The static network method regressed all explanatory variables (L2FC in phosphoproteins and categorical variables) against the phosphoprotein abundances at the same time point. The dynamic network method applied the same regression but for subsequent time points. For the DIONESUS algorithm, the explanatory variables were mean-centered with unit-variance to produce the X matrix. The values corresponding to the node being regressed against were set to zero to remove spurious connections as a result of auto-correlation. For each node, the dependent variables in the y-vector were defined as the z-score of the L2FC for the node's data points. For the dynamic network, the data at time point t was used to explain the data at time point t+1 using the same input matrix, X. We then solved for the vector of coefficients (beta) given the standard regression equation: [y=X*beta].

We solved for beta using the Non-Iterative PArtial Least Squares (NIPALS) algorithm implemented as the nipals() function[3]. All calculations were coded and analyzed using Matlab 2013a (Mathworks). The only tuning parameter was the number of principal components to include in the regression. Using the elbow rule, we employed three principal components as they explained over 90% of the variance without overfitting. The importance of each edge was calculated by z-scoring the VIP score for each regression problem using the vipp() function[3]. To account for experimentally inhibited phosphorylation activity, edge confidences were upweighted by the number of perturbations in which they were inhibited (e.g. if AKT-inhibitor was used in 2/5 of the training data, the edges connecting AKT to its daughter nodes were multiplied by 1+2/5). The edge importances were normalized between 1 and 0 and the final consensus network was formed by taking the arithmetic mean between the static and dynamic networks. We assumed that the scoring criterion would disproportionately weight true positives over true negatives so we chose a highly liberal threshold and included all non-self edges in the final submission regardless of confidence score.

Discussion

The DIONESUS algorithm introduces a novel PLSR-based inference approach that offers ultrafast computation of network structure that may be critical for the reconstruction of large-scale genomic/proteomic/signaling networks, and rapid iterations between predictive modeling and experimental design. The strengths of this method are its speed, simplicity, and accuracy, which are especially notable considering that no cross-validation or prior knowledge was used. In future research, we would like to explore the scalability of each algorithm on a larger, phospho-network from the assay of 91 phosphosites[4].

References

1. Wold, S., Sjostrom, M., and Eriksson, L. (2001). PLS-regression: a basic tool of chemometrics. *Chemometrics and Intelligent Laboratory Systems* 58, 109-130.
2. Kubinyi, Hugo, ed. 3D *Qsar in Drug Design: Theory Methods and Applications. Vol. 1.* Springer, 1993.
3. PLSR and CARS Toolbox: Hongdong Li 2011. /http://code.google.com/p/carspls/
4. Ciaccio, Mark F., et al. Systems analysis of EGF receptor signaling dynamics with microwestern arrays. *Nature Methods* 7.2 (2010): 148-155.

SC1 Network Inference: Team29

Summary

An MCMC-based EM algorithm is used to infer network structure and parameters for a logistic regression model with latent variables representing protein phosphorylation and functional states.

Introduction

We cast the task of inferring causal relationships as a Bayesian network (BN) structure-learning problem. While it is possible to represent the observed phosphorylated proteins as vertices of a BN, this representation has the limitation that phosphorylation of a protein does not always change the function of a protein. To address the task, we introduced a set of latent variables to represent the phosphorylation states of each phospho-protein, such that the observed phosphorylation signals can be modelled as signals contributed by proteins in two populations: phosphorylated proteins and un-phosphorylated proteins. In addition, we introduced a set of latent variables to represent the activation states of each protein, such that the activation state of a protein, e.g. Protein A, can causally affect the phosphorylation state of another protein, Protein B. This setting allows modelling the causal relationship between Protein A and Protein B independent of the phosphorylation states of Protein A, such that we model the situation that the Protein A is phosphorylated by its upstream protein but its function is blocked by a drug.

Methods

Learning BN structure including latent variables is a challenging task. Based on our biological knowledge, we constrain the search space to only find the causal edges between the activation state of a protein and phosphorylation state of other proteins. We developed an MCMC-based EM algorithm to infer the state of latent variables and to estimate the parameter associated with each edge in a framework of logistic regression. This setting enables us to search the structure of a BN through lasso regression between a node representing the phosphorylation state of a protein and candidate kinases in the data. Based on our knowledge regarding certain proteins studied in this project, we topologically sorted the proteins in the data, which further restricted the search space of edges between activation-state nodes and phosphorylation-state nodes to those satisfying the topological relationship. After training, we collapsed the latent variables and translate the network into a network among phosphorylation antibodies.

Discussion

Besides all the computational challenges we have overcome, the advantage of our method is the introduction of activation states of proteins as latent variables, which help us to model the causal relationship between proteins independent of the phosphorylation states. It allows us to capture the change in activation state of a protein resulting from factors other than phosphorylation by its upstream proteins, e.g., the drug-induced block of the function of a phosphorylated protein or a mutation that activate a protein without phosphorylation. We participated in the challenge only during the last week and did not use the leaderboard to fine-tune our method. Refinements to this model may improve performance.

References

1. Qin et al.: Signaling network prediction by the Ontology Fingerprint enhanced Bayesian network. *BMC Systems Biology* 2012 6(Suppl 3):S3.

SC1 Network Inference: Team30

Summary Sentence

Time-lagged correlation scores are calculated between the abundance of a target protein and abundances of proteins in a set of potential regulators

Introduction

Recent advances in the field of regulatory and signaling networks have shed light onto the problem of network inference. We have incorporated several of these ideas into our algorithm: 1) the effect of a species on another is governed by Hill kinetics [*Margolin]*; 2) an individual species is not usually regulated by more than 3 other species [*Liang]*; and 3) inhibition is always stronger than activation [Saadatpour]. Rather than focus on time series data *per se*, we consider that successive measurement will inform causally-related protein pairs. We derive the causality from the correlation of state in the first measurement and a change of the value of a species between the two measurements. The correlation measure was inspired by the mutual information approach of [*Liang*]. Finally, based on the assumption of exclusivity, we separately searched for activators and inhibitors.

Methods

Broadly, our approach to identification of context-specific networks consists of data normalization and calculation of correlations between phosphoproteins. Here, we use the term *species* to depict any component of the system, including the stimuli. We used the “main” dataset, and divided it into independent files, one for each cell line and stimulus. This resulted in 32 input files for SC1A, each corresponding to a single wiring diagram.

Our algorithm consists of the following steps:

1. Normalize the data in such a way that the lowest and highest measured values of a species are assigned 0 and 1, respectively, with intermediate values uniformly projected in the range [0,1].
2. Find all pairs of subsequent measurements for each experiment. In cases where a measurement is missing, the next available time point is considered.
3. For each pair, compute the vector of differences between values of species.
4. For each species, compute an average of all the positive differences (among all the measurement pairs) and compute second vector of values, called *production*, describing the difference of the increase from the average.
5. For each species, compute the cumulative correlation between the aforementioned differences and values of up to three species in a first measurement.

6. Use step (5) to independently determine a set of possible activators and inhibitors. Here, we always choose the combination of regulators with the highest cumulative correlation.

The key features of the approach is the computation in step 5, where we use the logistic function:

*logistic((max(x, max(z, y)) + x*y + x*z + y*z - y*x*z) / 3),*

where *x, y, z* are species of the system (possibly equal) and *logistic* is a standard logistic sigmoid function, which first uniformly maps its argument from [0,1] to [-6,6].

The expression over the arguments reflects that a single occurrence of a highly active regulator is more relevant than a mild occurrence of three of them. The logistic function simulates the sigmoid-like regulation pattern. The result of the logistic call is then multiplied by a “change” factor. When searching for inhibitors, the change factor equals the negation of a difference between measurement times. When searching for activators, we use the production value calculated in step 4. Finally, we remove the stimuli from the resulting graphs for the purpose of the submission. This method does not rely on any prior network knowledge from the literature.

Discussion

The performance of the computation was rather good (around a second for a file). As pointed in the step 6, we searched for a set of activators and inhibitors independently. This allows for higher total amount of regulators, if they have opposing effects, as it would be in the case where all regulators are considered. This is a strong computational boundary as the number of possible combinations to test grows exponentially with the maximal number of regulators considered. Splitting the dataset into 32 files, each having a single stimulus, allowed for simplification of the method. However, we have not adjusted the resulting data to account for the fact that there were collections of 8 datasets that were measured on the same cell line. Arguably, this meant that we discarded some of the prior knowledge, which could otherwise be used for quality adjustments of the results. We have included stimuli as components of the system. This step was unnecessary in hindsight and only resulted in an increase in the computational complexity.

References

1. Liang, S., Fuhrman, S., & Somogyi, R. REVEAL, a general reverse engineering algorithm for inference of genetic network architectures. *Pacific Symposium on Biocomputing*. Vol. 3. No. 18-29. 1998.
2. Margolin, A. A., Nemenman, I., Basso, K., Wiggins, C., Stolovitzky, G., Favera, R. D., & Califano, A.. ARACNE: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. *BMC Bioinformatics*, 7(Suppl 1), S7. 2006.
3. Saadatpour, A., & Albert, R.. Boolean modeling of biological regulatory networks: a methodology tutorial. *Methods*. 2013 Jul 15;62(1):3-12

SC1 Network Inference: Team31

Summary

An ensemble approach that combines prior information with the results from three existing methods: ordinary differential equations, dynamic Bayesian networks, and Bayesian Regression.

Introduction

Various methods have been proposed for inferring molecular networks. We used 3 existing network inference methods and combined them to obtain a causal protein signaling network. The first 2 network inference methods individually use time series measurements to find the posterior probabilities of one node affecting the other. Between these 2 approaches, the first one uses ordinary differential equations (ODEs) and Gaussian processes to come up with the probabilities; the second one uses dynamic Bayesian networks (DBNs) for representing probabilistic relationship among interacting nodes. A third network inference method based on Bayesian regression uses the inhibitor conditions to find the inhibitory and activating relationship among different nodes of the network. The confidence score on each edge of the network was deduced from posterior probabilities found from first two methods and absolute value of the edge scores found from the third method.

Methods

Network inference methods

An ODE and Gaussian process based inference method [1] has been used as our first method to find posterior probabilities of one node of the network affecting the other. The second method [2], which was also used to calculate posterior probabilities, is based on dynamic Bayesian networks. A third method [3] based on a Bayesian regression approach has been used to find the connection direction and scores of the edges of the network.

For the experimental sub-challenge, we separated the “main” dataset into 32 different datasets (combination of 4 cell lines and 8 stimuli) and for the *in silico sub*-challenge, there was only 1 dataset present. Each of the 33 datasets has time series proteomic measurements that have been used as input to the 3 approaches described above to find the network edge parameters.

For first method [1], each of the 33 separated datasets has time series measurements for up to 4 different inhibitors (exactly 3 inhibitors for in-silico data) applied to the cell line. So, we have split each of the 33 dataset into 4 or (less) number of sub-dataset based on number of inhibitors applied. If *n* (<=4) is the number of inhibitors present in the dataset, these *n* sub-datasets are considered as *n* different time-series measurements and put into the algorithm that gives the posterior probabilities of the nodes affecting each other.

For second method [2], if *n* (<=4) is the number of inhibitors present in each of 33 dataset, the *n* sub-datasets are used separately to find posterior probabilities of the nodes affecting each other. All *n* probabilities for a single edge are then averaged out to find the final probability using this approach.

The third method [3] does not care in which time point the data are taken, but in each measurement, one protein has to be inhibited. We’ve created a dummy node and made the expression of it very low to look like it’s been knocked out for all the time point measurements. To confirm closeness to steady state data, we’ve considered taking 2 hour data (in-silico data) and 4 hour data (experimental data) from the main datasets for each of 33 networks as steady state measurements. This approach gives edge scores that contain a confidence measure of how well the regulation is working; also a signed edge score determines if one node is activating or inhibiting the other.

Combined approach

The posterior probabilities found from first 2 approaches and the absolute value of edge scores found from approach 3 are considered and average of these 3 measures are taken as edge confidence score. A threshold of 0.1 was taken to cut some of the connections. If the network has *N* nodes, then the edge score matrix *M* has a dimension of *N*x*N*. Element (*i,j*) represents confidence that node *i* is affecting node *j*. if *M*(*i,j*)>*M*(*j,i*), then *M*(*j,i*) is set as 0. Then the signs of the edge scores are added back in using the confidence scores from approach 3. We've cut some of the edges found from our approach to make the network causal and directed.

Additional details
MATLAB codes were available for all the 3 approaches described. Implementation of those approaches using the experimental data and *in silico* data have been done by us, also the combination of the results from 3 approaches is done by our team.

Prior biological knowledge on protein-protein interaction has been used to see if the connections obtained from our approach actually match with information found in the literature. If anything is found, then the confidence score is boosted up for that edge. For both sub-challenges, if multiple measurements were present for a single time point, we've always taken average of them.

Discussion
We've kept our approach simple by inferring networks using existing methods and combining those methods to calculate confidence scores for edges in causal networks. To maintain simplicity, we did not use stimulus information for *in silico* data. Also, we've used prior biological knowledge on protein-protein interaction to boost up confidence score of an edge.

References

1. Tarmo Aijo and Harri Lahdesmaki. Learning gene regulatory networks from gene expression measurements using non-parametric molecular kinetics, *Bioinformatics*, Vol 25, No. 22, 2009, pages 2937-2944
2. Marco Grzegorczyk and Drik Husmeier. Improvements in the reconstruction of time-varying gene regulatory networks: dynamic programming and regularization by information sharing among genes, *Bioinformatics*, Vol 27 No 5, 2011 pages 693-699.
3. Simon Rogers and Mark Girolami, A Bayesian regression approach to the inference of regulatory networks from gene expression data, *Bioinformatics*, Volume 21, No 14, pages 3131-3137.

SC1 Network Inference: Team32

Summary

A consensus network learning method that integrates results from several representative network-inference algorithms, applied separately to data from each inhibitor condition.

Introduction
Recently, a large number of techniques, especially consensus learning approaches that integrate different algorithms, have become a potentially promising strategy to infer accurate gene regulatory networks from gene expression datasets [1]. To see whether consensus driven approaches can infer accurately protein signaling networks from protein abundance data, we examined methods that integrate multiple algorithms as well as results from different protein-abundance datasets. A novel innovation in our consensus approach is to integrate results from different protein-abundance datasets, *i.e*., dataset with GSK690693, that with GSK690693 + GSK1120212, that with PD173074, and that with DMSO. The underlying methodology used in this study is our consensus learning approach, "Top*k*net" [1], which was published recently. In this study, Top*k*net integrate multiple algorithms based on random-forests, regression, directed

partial correlation (DPC), dynamic Bayesian networks (DBN), mutual information, and ordinary differential equations.

Methods

Method for experimental data

We used all cell lines, stimuli, time points and inhibitors. We used the “Main” dataset but did not use extra antibodies and time points in the “Full” dataset. We did not normalize, log-transform or discretize the data and ignored missing data. We regard replicates as different samples. In Part A, we used Top*k*net [1] to integrate five representative network-inference algorithms that can infer causal links between two proteins, *i.e.*, DBN, DPC, GENIE3, NARROMI, and TIGRESS. The procedure of our consensus learning method is composed of three steps. (i) From a dataset consisting of a single inhibitor or DMSO, a network-inference algorithm assigns an edge-score to each link and links are ranked according to their confidence levels, *i.e.*, a link with higher edge-score has higher rank value. (ii) The rank values for each algorithm were normalized by scaling from 0 to 1 and the normalized values were regarded as edge-scores by the algorithm. (iii) For each link, the five network-inference algorithms assigned five edge-scores to each link. For each edge, Top*k*net [1] selects the highest edge-score among the five edge-scores and regards the selected edge-score as the score of the edge. For example, if the five algorithms assign five edge-scores, 0.9, 0.8, 0.2, 0.6, and 0.7 for a link, our method regards 0.9 as the edges score of the link._From four datasets (dataset with GSK690693, that with GSK690693 + GSK1120212, that with PD173074, and that with DMSO) under each cell line and stimulus, our consensus learning method generated four edge-scores for each link. We regarded the mean value among the four edge-scores as the edge score of the link under the cell line and stimulus. For example, if our method assigns edge-scores, 0.4, 0.6, 0.1, and 0.3 for a link from four datasets, the score of the link is regarded as 0.35.

Methods for in silico data

We used all stimuli, time points and inhibitors. We did not normalize, log-transform or discretize the data. Note that, because NARROMI, TIGRESS, and GENIE3 regard time points as samples, we combined all 20 time courses (each one corresponding to a different stimulus/inhibitor combination) of the *in silico* challenge to make one combined “time course”(with 220 time points) and used the combined dataset as input to the three algorithms. On the other hand, DBN used all 20 time-courses as inputs to infer one network. We used Top*k*net [1] algorithm to integrate four representative network-inference algorithms, *i.e.*, GENIE3, TIGRESS, NARROMI, and DBN. The procedure used is composed of three steps. (i) A network-inference algorithm assigns an edge-score to each link and links are ranked according to their confidence levels. (ii) As in Part A, the rank values for each algorithm were normalized by scaling from 0 to 1 and the normalized values were regarded as edge-scores by the algorithm. (iii) For each edge, Topknet [1] algorithm selects the highest edge-score among the four edge-scores from the four network inference algorithms and regards the selected edge-score as the score of the edge.

Silencing of indirect links

The procedures for experimental data generated one edge-score matrix, **G**, for each cell line and stimulus, while that for *in silico* data generated one **G** from 20 datasets under different conditions. However, edge-scores in **G** may be affected by indirect and direct links. To address this issue, Barzel et al. developed a method to silence indirect effects [2]. Their method can transform **G** into a highly discriminative silenced matrix, **S**, that may be able to identify direct causal links. For each **G**, we applied the method by Barzel et al. to obtain **S** from **G**. We regard the transformed edge-scores in **S** as final edge-scores of links.

Existing tools and packages used in this study
We implemented Top*k*net algorithm in R. The existing tools, packages, and algorithms that we used are: GENIE3 (http://www.montefiore.ulg.ac.be/~huynhthu/software.html), TIGRESS (obtained from GP-DREAM network inference website), and DPC (https://code.google.com/p/dpcnet/), and Matlab source codes for NARROMI (http://csb.shu.edu.cn/narromi.htm), and DBN (http://mukherjeelab.nki.nl/DBN.html).

Tuning parameters
Our method integrated five existing algorithms and four of them have tuning parameter. For DBN, we set max_in_degree = 4 and reg_mode = 'full'. For Genie3, we set nb.trees = 1,000 and 10,000 for Parts A and B, respectively. For TIGRESS, we set nstelPARS = 5 and 2 and nbootstrap = 100 and 10,000 for Parts A and B, respectively. The other parameters in the four algorithms are default.

Discussion
We applied a consensus learning method, "Top*k*net" [1], that can integrate various types of network-inference algorithms, although, in this challenge, we focused on the five representative network-inference algorithms. To find the best combination of algorithms, in future, we will try a large number of algorithms and various combinations of these algorithms.

References

1. Hase T, Ghosh S, Yamanaka R, Kitano H (2013) Harnessing Diversity towards the Reconstructing of Large Scale Gene Regulatory Networks. *PLoS Computational Biology*, 9:e1003361.
2. Barzel B and Barabási AL (2013) Network link prediction by global silencing of indirect correlations. *Nature Biotechnology*, 31:720-725

SC1 Network Inference: Team34

Summary

A novel integrated approach was used to construct consensus signaling networks with high confidence interactions from three different network inference algorithms, weighted according to their agreement with a prior knowledge network.

Introduction
Protein kinases play a very important role in controlling cell growth, proliferation and survival. Abnormal behaviours of protein kinase signalling networks often result in specific diseases, including cancers. The knowledge of signalling pathways has been captured in manually curated biological databases such as the Protein Interaction Database (pid.nci.nih.gov) and KEGG (www.genome.jp/kegg). Rich data from large-scale functional genomics studies are still largely untapped resources for understanding the signalling cascades**.** Although the research community has proposed many network inference algorithms, none of them can accurately infer the complete protein kinase signalling network and the results vary substantially. Here we describe a novel ensemble method for inferring signalling networks from dynamic proteomic data. Specifically, we devised an integrated approach to construct consensus networks inferred with three network inference algorithms: dynamic Bayesian networks (DBN), Max-min hill climbing (MMHC), and graphical Gaussian models (GGM). Results from these three approaches were integrated to create a single network representation for each condition of the HPN-DREAM challenge.

Methods

Our approach has three main steps: network inference prediction using standard algorithms, evaluation against known biological pathways, and calculation of edge confidence scores. Each of these steps is detailed below.

Step 1. Perform cross-validations of networks in each method using Monte Carlo algorithm to generate the edge score. Initially we investigated five different network inference algorithms: generalized linear model (GLM), lasso and forward stage-wise regression model (LARS), dynamic Bayesian network using 1st order conditional dependencies (DBN), graphical Gaussian models (GGM) and max-min hill-climbing (MMHC). We used the Monte Carlo algorithm to handle replicates in the input data and generate the edge scores for networks constructed with each method. This approach allows repeated measurements, irregular sampling, and unequal temporal spacing of the time points. In each resampling, one zero time data point was used. The repeated sampling also helped the calculation of the edge score, which was equal to the frequency of an edge in the total number of networks.

Step 2. We evaluated the method accuracy by comparing the inferred networks against existing biological pathways from KEGG. To determine the algorithm accuracy and assign method weights. The overlapping rate between each network and KEGG pathways was calculated. Each method was assigned a method weight based on the accuracy rate. Weights of 50%, 30% and 20% were assigned to DBN, MMHC, and GGM correspondingly, which reflect the method accuracy. Two methods, GLM and LARS, were excluded from the later analysis due to the low overlapping rate.

Step 3. Edge confidence scores were calculated from the sum of products of edge scores and method weights. A cut-off threshold for the confidence scores was selected on the basis of the score distribution, and only edges with high confidence scores were used to construct consensus networks. These consensus networks are more robust than the networks constructed by an individual algorithm. The resulting consensus networks represent the nodes and edges predictions from three different algorithms most likely supported by the prior knowledge.

Implementation details

The five network inference algorithms were applied with standard R functions or packages: R glm() function for GLM, lars package for LARS, G1DBN package for DBN, GeneNet package for GGM, and bnlearn package for MMHC. The Monte Carlo algorithm was used to handle replicates with as.longitudinal function in the R longitudinal package. The edge confidence scores, method weights and consensus networks were built using in-house Perl scripts.

We used the data from all cell lines, stimuli, time points and inhibitors. For the experimental sub-challenge, we used "Main" dataset only. The data were log-transformed prior to modelling. Missing data were not specifically handled. The same resampling method and algorithm weights learned from SC1 were applied to the network construction in the *in silico* challenge.

Discussion

It is an extreme challenge to infer the protein kinase signalling networks from the large scale functional proteomics experiment data. The difficulties include 1) The signaling network is a dynamic system which changes in different conditions and time points; 2) The protein interactions are very complex and non-linear; 3) The heuristic approaches modelling the network are subject to generate high false positive and negative results. To deal with these issues, we integrated networks from multiple network inference algorithms into a consensus network structure. Different algorithms could detect different interaction signals in the network. The inferred networks were evaluated with KEGG pathways to determine the accuracy for each algorithm. The interactions consistent within and between methods were assigned high confidence score and selected for building consensus networks. The result should be more accurately reflecting the signalling process than networks from each individual algorithm.

References

1. R Development Core Team (2008). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. http://www.R-project.org.
2. Christian Robert and George Casella. Introducing Monte Carlo Methods with R. Use R. Springer, 2010. ISBN 978-1-4419-1575-7.

SC1 Network Inference: Team35

Summary

A dynamic Bayesian network approach that uses exact model averaging to determine the weight of each possible network edge. A related static Bayesian network approach was implemented as the submission for Team38.

Method

The method is a modified version of the Team38 submission, and full details are provided in that write-up. In brief, the key difference is as follows. As in the static approach, we make each protein a target, and make the remaining proteins predictors. However, we create records in which the predictors have their values at time t and the target has its value at time $t+1$. Using these records, we then proceed exactly as described in the Team38 write-up for the static Bayesian network approach.

SC1 Network Inference: Team36

Summary

A network inference method based on algorithmic information theory using the Normalized Information Distance and its computable version, Normalized Compression Distance.

Introduction

A series of information similarity distances from a proteomics time-course dataset on four breast cancer cell lines provided by the HPN-DREAM challenge were tested in the network reconstruction task. These similarity measures are based on Algorithmic Information Theory (AIT). Among its powerful features, AIT provides the means for optimal inductive inference, making it relevant to the task of biological network reconstruction, as biological data are regularly incomplete and noisy and therefore powerful formal inference methods are needed. The first measure used in the task of network reconstruction was the so-called Normalized Information Distance (*NID*). *NID* measures the *similarity* between 2 objects and retrieves a value between 0 and 1 defined as:

$$NID(x,y) := \max\{K(x|y), K(y|x)\}/\max\{K(x),K(y)\}$$

where $K(x|y) := \min\{l(p) : U(y,p) = x\}$, that is the length of the shortest program p with input y running on a universal Turing machine U that produces x [Ref 1].

Methods
We applied the same methodology for inferring the underlying networks in both the experimental sub-challenge and the *in silico* sub-challenge. All portions of the HPN-DREAM data were used, but only from the Main file. We lumped all data but also began to perform some experiments with combinations, especially taking single replicates over time and making comparisons between them.

In the context of network reconstruction, two nodes x and y were considered to be (causally) related and therefore connected in the resulting network N if $NID(x,y) < n$. The choice of n is determined by the estimation of the Kolmogorov complexity $K(N_m)$ of a set of sampled inferred networks with resulting different topology N_m for varying m. A complete graph, for example, has a low K value while a random graph has maximal K value. Thus $n := m$ when $K(N_m)$ reaches its greatest value for all N_m, under the assumption that the underlying data is not random and therefore the first network with maximal K is the most informative.

One initial difficulty of measures based on Kolmogorov complexity is that their numerical calculation is difficult, if not impossible, due to their power; this is because measures based on AIT are guaranteed to asymptotically recognize any computable regularity, that is a regularity that can be written in a computer program as a statistical test running on a universal Turing machine. A common way to numerically approximate K and NID is using compression algorithms. The Normalized Compression Distance based on NID uses a compression algorithm as a means to approximate NID and is defined by:

$$NCD_C(x,y) = (C(xy) - \min\{C(x), C(y)\})/\max\{C(x), C(y)\}$$

where xy is the concatenation of x and y and $C(x)$ is the compressed length of x using compression algorithm C. The more optimal the value of C, the closer it is in value to $K(x)$, which yields a better approximation of NCD to NID.

The compression deficiency hypothesis
Correct directionality determination of nodes in the inferred network was key to being well ranked in the challenge leader board, but NID is commutative and we were therefore strictly speaking not able to determine the directions. However, given that $NID(x,y) = NID(y,x)$ but $NCD_C(x,y)$ is not necessarily equal to $NCD_C(y,x)$ for some lossless compression algorithm C (ideally robust over different C), we formulated a hypothesis. The compression deficiency hypothesis is the bold idea that a compression algorithm performs better on node variable values if they are provided in the order of their relative causal relation rather than the reverse. We tested this hypothesis by feeding the compression algorithm with both $C(x,y)$ and $C(y,x)$ and determined the direction of a link forward, $x\text{->}y$, if $C(x,y) > C(y,x)$, and $y\text{->}x$ otherwise (bidirectionality for equality). If both networks had similar scores then the deficiency hypothesis would be rejected. Under the same hypothesis to test, we took the absolute value of the difference $NCD_C(x,y) - NCD_C(x,y)$, as a confidence estimation of the directionality of the node connections.

Discussion
One result that we could discern in the HPN-DREAM challenge was that network reconstruction was robust under different lossless compression algorithms---at least for the two compression algorithms tested, Deflate and BZIP2, which are among the most commonly used and best performing in data compression. Indeed, in [Ref. 2] it was shown that algorithmic complexity approximations of graph and networks deliver stable results in agreement with the theoretical expectation using a technique based on algorithmic probability and also by uncompressibility.

The results from this challenge suggest that improvements on various fronts are possible in order to arrive at a reasonable alternative method for network reconstruction. Some preliminary results suggest that the method may be good at finding networks. This is because the main characteristic of Kolmogorov complexity is that it provides an information content value of an object independent of the ensemble and is therefore less sensitive to correlation loops.

Alternatives to compression algorithms may constitute a possible improvement, given that correlation signals may be weaker than the noise from deficiencies introduced by compression algorithms from small sample effects. However, new ideas based on other algorithmic information measures suggest that the determination of link direction is possible and this remains something to be further explored. Additionally, the threshold technique described in the Methods section is independent of the inference network model and hence can be applied to other algorithms independently.

Other similarity distances based on AIT have yet to be tested, and investigations are currently underway. For example, one idea to overcome this drawback is the use of a variation of K to which an important biological currency is added: time, like evolutionary time. Indeed, concepts such as Bennett’s logical depth do tell apart randomness from structure by taking into account the decompression time of compressed data. Methods based on Kolmogorov complexity are very novel and are in quite an early stage of development. We think it will prove to be a very interesting idea.

References

1. M. Li and P. M. B. Vitányi (1997). *An Introduction to Kolmogorov Complexity and its Applications*. Springer, New York, 2nd edition.
2. H. Zenil, F. Soler-Toscano, K. Dingle and A. Louis, Correlation of Automorphism Group Size and Topological Properties with Program-size Complexity Evaluations of Graphs and Complex Networks, Physica A: Statistical Mechanics and its Applications (in press).

SC1 Network Inference: Team38

Summary

A (static) Bayesian network approach that uses exact model averaging to determine the weight of each possible network edge. A related dynamic Bayesian network approach was implemented as the submission for Team35.

Introduction

Epistasis is an interaction in which several genes combined affect disease, and the net effect on phenotype cannot be predicted by simply combining the effects of the individual loci. We had previous success learning epistatic interactions from high dimensional datasets using the *Bayesian Dirichlet Equivalent Uniform* (*BDeu*) score [1] for Bayesian networks [2] to score candidate interactions [3-7]. Because the proteins on a signal pathway could also interact to affect the network, we apply the same approach to learning edges that exist between proteins on a signaling pathway.

We model using a Bayesian network approach, however, we do not try to learn an entire Bayesian network. Rather we make each protein a target, and make the remaining proteins predictors. A *causal model* is the *directed acyclic graph* (*DAG*) in which some subsets of the predictors are the parents, and the target is the child. We score all 0, 1, 2, 3, and *k* predictor causal models M using the BDeu score, which is the $P(Data \mid M)$. We assume all 0, 1, 2, 3,…, and *k* predictor causal models are equally probable a priori, and finally compute $P(M \mid Data)$ for each model M. We then compute the probability of each edge using model averaging. We finally remove edges based on conditional independency considerations, as described in the Methods Section.

Methods

We provide a complete description of the method in this section. Since the method uses Bayesian networks and the BDeu score we first briefly review these concepts. Bayesian networks [2] are commonly used for uncertain reasoning and machine learning in many domains, including biomedical informatics [8-9], and in particular signaling pathways [10]. A *Bayesian network* (*BN*) consists of a *directed acyclic graph* (*DAG*) $G = (V, E)$, whose node set V contains random variables and whose edges E represent relationships among the random variables. A BN also includes a conditional probability distribution of each node $X \in V$ given each combination of values of its parents. Each node *V* in a BN is conditionally independent of all its non-descendants given its parents in the BN. Often the DAG in a BN is a causal DAG [2]. Using a BN, we can determine conditional probabilities of interest with a BN inference algorithm [2].

The task of learning a BN from data concerns learning both the parameters in a BN and the structure (called a DAG model). In a score-based structure learning approach, we assign a score to a DAG based on how well the DAG fits the data. Cooper and Herskovits [11] introduced the Bayesian score, which is the probability of the *Data* given the model *G*. A popular variation of the Bayesian score is the *Bayesian Dirichlet equivalent uniform* (*BDeu*) score [12], which allows the user to specify priors for the conditional probability distributions using a single hyperparameter α, called the prior equivalent sample size. That score is as follows:

$$score_{\alpha}(G \ : \ Data) = P(Data \mid G) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\Gamma(\alpha / q_i)}{\Gamma(\alpha / q_i + \sum_{k=1}^{r_i} s_{ijk})} \prod_{k=1}^{r_i} \frac{\Gamma(\alpha / r_i q_i + s_{ijk})}{\Gamma(\alpha / r_i q_i)},$$

where r_i is the number of states of node X_i, q_i is the number of different instantiations of the parents of X_i, and s_{ijk} is the number of times in the data that X_i took its k th value when the parents of X_i had their j th instantiation.

The steps in our static Bayesian network approach for learning the edges in a signaling pathway from time series phosphorylation data on the proteins in the pathway are as follows:

1) Discretize the values of each of the protein variables separately into 3 values, *low*, *medium*, *high*, using the equal distribution discretization method.
2) Using these data, do the following for each protein:
   a) Make the protein the target and make the remaining variables predictors. A *causal model* is the DAG in which the variables in some subset of the predictors are the parents, and the target is the child. Score all 0, 1, 2, 3,..., and *k* predictor causal models M using the BDeu score with prior equivalent sample size α. The BDeu score provides the $P(Data \mid M)$. Assume all 0, 1, 2, 3,…, and *k* predictor causal models are equally probable a priori, and finally compute $P(M \mid Data)$ for each model M using Bayes' Theorem.
   b) Compute the posterior probability of each edge by averaging over all the models. That

is,

$$P(X \to Y \mid Data) = \sum_{M} P(X \to Y \mid M) P(M \mid Data).$$

where $P(X \to Y \mid M)$ equals 1 if edge $X \to Y$ is in model M and equals 0 otherwise.

3) After the above procedure is performed for all proteins, the result is a saturated directed graph, where the weight on each edge is the probability of the edge.
4) Remove edges from this graph based on conditional independency considerations as follows: If the edge $X \to Y$ exists and there is a directed path ρ from X to Y such that the edge from *X* to *Y* is weaker than a parameter β times the weakest edges on ρ.

The method has three tuning parameters: α, the parameter in the BDeu score*; k*, the number possible parents to consider when performing model averaging; and β, the parameter used when deleting edges. For this submission, we used α=9, k=4, and β=1. If one does a grid search using many values of these parameters, it seems likely that the performance of the method could improve significantly, making it perhaps one of the more viable methods for accomplishing learning edges in a signaling pathway from phosphorylation data. Additionally, different discretization strategies could be tested. This method is strictly a tool for learning from data in that it does not incorporate any biological prior knowledge.

References

1. Heckerman D, Geiger D, Chickering D. Learning Bayesian networks: the combination of knowledge and statistical data. Technical Report MSR-TR-94-09. *Microsoft Research*; 1995.
2. Neapolitan RE. *Learning Bayesian Networks*. Upper Saddle River, NJ: Prentice Hall; 2004.
3. Jiang X, Neapolitan RE, Barmada MM, Visweswaran S, Cooper GF. A fast algorithm for learning epistatic genomics relationships. *AMIA Ann Symp Proc* 2010, 341-345.
4. Jiang X, Neapolitan RE, Barmada MM, Visweswaran S. Learning genetic epistasis using Bayesian network scoring criteria. *BMC Bioinformatics* 2011, 12(89):1471-2105.
5. Jiang X, Barmada MM, Cooper GF, Becich MJ. A Bayesian method for evaluating and discovering disease loci associations. *PLoS ONE* 2011; 6(8): e22075.
6. Jiang X, Neapolitan RE. Mining strict epistatic interactions from high-dimensional datasets: ameliorating the curse of dimensionality. *PLoS ONE* 2012; 7(10): e46771.
7. Jiang X, Visweswaran S, Neapolitan RE. Scoring and searching Bayesian network models of gene-phenotype association, in Sinoquet C, Mourad R (Eds.): *Probabilistic Graphical Models for Genetics*, Oxford University Press, 2014.
8. Neapolitan RE. *Probabilistic reasoning in bioinformatics*. Burlington, MA: Morgan Kaufmann; 2009.
9. Segal E, Pe'er D, Regev A, Koller D, Friedman N, Learning module networks. *Journal of Machine Learning Research*; 2005; 6.
10. Sachs K, et al. Bayesian network approach to cell signal pathway modeling. *Sci STKE*; 2002; 148: pe38.

SC1 Network Inference: Team40

Summary

Direct cause-effect relationships, inferred from a large-scale literature-based phosphorylation network, are used to construct shortest-path trees that explain the experimental data.

Introduction

The purpose of this submission was to make an initial assessment of the usefulness of graph-based methods applied to large-scale literature-derived networks for the type of inference problem represented by the HPN-DREAM challenge. The algorithm operates on a master network of observed cause-effect relationships relating to protein phosphorylation with 4433 protein nodes and 21594 directed edges. Cause-effect relationships in this network reflect diverse experimental contexts, and represent direct as well as indirect causal effects. One of the main challenges to construct meaningful subnetworks for possible causal mechanisms is to distinguish direct relationships from those that are indirect in nature, and which can potentially be explained by a causal sequence (or path) of direct effects.

Here, we employ a heuristic stochastic algorithm that aims to distinguish direct from indirect cause-effect relationships by using the underlying network topology. The central idea is to identify direct edges as those that are likely involved in many causal paths, and therefore able to explain a large number of indirect edges. In particular we look at “transitive causal triangles” with edges $A \rightarrow B$, $B \rightarrow C$, and $A \rightarrow C$, where both (direct) edges $A \rightarrow B$ and $B \rightarrow C$ potentially “explain” the (indirect) edge $A \rightarrow C$. The algorithm samples a maximum entropy ensemble of subgraphs of the original large-scale network such that the expectation value of the total number of explained edges $<N_e>$ assumes a particular value. The probability of a subnetwork with N_e explained edges in this ensemble is given by $p \sim exp(□N_e)$, where $<N_e>$ is a function of the control parameter □.

Marginal edge probabilities *p* are used to define edge weights $w = - log(p)$ where small values of *w* correspond to inferred direct causal effects, and larger values of *w* represent indirect relationships. Potential causal mechanisms are then constructed as trees assembled from shortest paths connecting inhibitor-affected proteins with proteins whose phosphorylation state was observed to have changed in the experiment.

Method

The method was only applied to the experimental data part of the HPN-DREAM challenge since it is based on prior knowledge and cannot be applied to *in silico* data. The method was implemented from scratch in Python and C.

Pre-processing of phosphorylation data

The actions of the different stimuli were not modelled. For the actions of the inhibitors we determined those phosphoproteins whose phosphorylation state changed significantly (>10% relative to DMSO vehicle with greater weight on early time points). No other time course information was used, and no extra time points were used from the “full” data set. All cell lines and stimuli were used separately to construct hypothesis networks. When mapping phosphoproteins, and proteins targeted by inhibitors to the master network, information about different phosphorylation sites on the same protein is lost.

Directness inference

As described in the introduction, edge weights reflecting the “directness” of the underlying causal effects are inferred in a large-scale phosphorylation network. This is done by sampling an ensemble of subnetworks using standard Markov-chain Monte Carlo with given parameter □, by randomly switching edges “on” and “off”, and measuring marginal edge probabilities *p*.
A total number of 1000 “sweeps” through the network is enough to determine edge probabilities with sufficient accuracy. For the subsequent step of causal network construction we found □ =1 to be a reasonable choice, and performed no further optimization for this submission.

Causal network construction
Dijkstra's algorithm with edge weights as determined above was used to construct shortest paths from inhibitor-affected proteins to those phosphoproteins whose phosphorylation changed significantly (see above). The latter step essentially infers causal ordering of the observed effects based on prior-knowledge present in the large-scale network.

Discussion
This submission was an initial attempt to use a large-scale prior-knowledge-derived network for the inference of "mechanistic" causal network hypotheses from time-dependent phosphorylation data. A heuristic algorithm was devised to distinguish between direct and indirect effects present in the network. Work remains to be done to assess strengths and weaknesses of this approach. Obviously the presented method can only be as good as the master network itself, i.e. can only include causal effects that have been previously observed (albeit in a possibly different context). Other constraints are the lack of phosphorylation site specificity, and the fact that directionality of causal effects (i.e. their activating or inhibiting nature) were not taken into account, both of which are not limitations in principle.

SC1 Network Inference: Team41

Summary
Lasso-based regression with iterative random feature selection to identify regulators that best predict the expression level of target proteins.

Introduction
Many methods have been developed for network inference, each with its own set of limitations. We surmise that combining different methodologies will overcome the limitations of individual methods through complementary integration. To that end, we developed a novel lasso-based algorithm to identify the regulators that best predict the expression level of target genes and proteins. Our method integrates ideas from lasso regression and bootstrapping. A limitation of lasso regression is that selected features tend to be over-fitted to the tuning parameter λ, which leads to instability of results. The instability in feature selection can be mitigated with bootstrapping-based methods where the data are randomly re-sampled. In the case of small sample size, however, re-sampling may not be effective. Another limitation of bootstrapping is that the true variable (regulator gene or protein) is likely to be missed (false negative) when strong indirect or direct regulators exist. To overcome these weaknesses, we developed a lasso-based random feature selection algorithm (LARF). In LARF, we regard a sparse linear regression as a feature selection because our goal is to identify the regulators that best predict the expression level of target genes and proteins. LARF is similar to bootstrapping, except that it selects variables among randomly pre-selected candidate features in each iteration over different tuning parameters of lasso optimization. The result is that true features weakly correlated to the target gene or protein can be identified, excluding indirect or direct regulators from the feature set. The proposed method has been preliminarily tested with DREAM3 data, and also compared to state-of-the-art methods such as top-ranked methods in the DREAM3/4 challenge.

Methods
To improve upon the problems of over-fitting and strong indirect regulation, we iteratively performed lasso regression over different λ with randomly predefined candidate features, rather than random samples as with bootstrapping. More precisely, the basic idea of LARF is that

lasso is iteratively performed with only randomly selected candidate features while increasing the tuning parameter, then giving weight to each feature by counting how many times each feature is selected in the iterations. We pre-defined the fraction of the number of all possible features as a parameter ($0 < \alpha < 1$) for the candidate features. For example, when the number of all possible regulators is n=100, α=0.2 means that 20 random candidate features are used in a single iteration of lasso. After random featuring, random sampling is performed with parameter *r,* which decides how many samples are used from the original data. With randomly (uniform distribution) selected features and samples by parameter, we iteratively run lasso over increasing tuning parameter λ until lasso does not select any features due to a high λ. In each iteration, random candidate features and samples are re-defined again. The tuning parameter starts from zero and increases incrementally with a small step-size (e.g 0.001) to ensure that re-featuring and re-sampling will be unbiased. For each iteration, the frequency matrix F is updated. The i-th row of F is the frequency of feature selection for target node i ($F_{i\,i}$ is zero). We applied the method to the *in silico* data only and used all training data without any external data. The data were mean-centered prior to modeling. The method is implemented in Matlab with the LASSO toolbox. The parameter α and r is set to 0.5 and 1, which means random sampling is not applied.

Discussion

There are several reasons why our method may not have performed well. Importantly, LARF was originally designed for steady state data, and so we may need to reconsider how it is applied to dynamic time series data. In this challenge, we set a target node to $X_i^{t+1}$and regulators to $X_{\setminus i}^{t}$. A blind assessment of our method tested on *in silico* data was informative, and we aim to test this method with experimental data in another DREAM challenge.

Supplementary Note 2: HPN-DREAM Time-course Prediction Sub-challenge (SC2) Methods

Team submission files and code (where available) can be found on Synapse at https://www.synapse.org/HPN_DREAM_Network_Challenge, under the section "HPN-DREAM Community Resource".

SC2 Time-course Prediction: Team3

Summary

We used COPASI to model reactions using mass-action kinetics with phosphorylated species as essential modifiers for subsequent reactions.

Methods

We implemented our network into COPASI, a software application for simulation and analysis of biochemical networks in which reactions are modeled using mass-action kinetics with phosphorylated species as essential modifiers for subsequent reactions. Stimuli and test inhibitors were included as global quantities. Test-inhibitors and stimuli may be applied to the model by setting the global quantity from "0" to "1" (found at: Model -> Biochemical -> Global Quantities). In general, when switching between these two values, all subsequent forward-reactions of the inhibited target are reduced by (at least) 90%. The stimuli lead to a 10-fold increase of the reaction speed of forward-reaction of the stimulated target (receptor). Reproducing our results is possible by applying the specific test-inhibitor and stimulus set and running the time-course (found at: Tasks -> Time-Course).

We used the network inferred from sub-challenge 1 as the basic structure for constructing our dynamic model. As mentioned in the network inference write-up document, our evolving model was used as cross-validation for the network and vice-versa. In this case we used COPASI's ODE based modeling to describe the species' time-courses. To avoid complications when opening the models we suggest using COPASI build 55. Using an earlier or later version may cause problems. The model was constructed with high simplicity to exclude as many unfounded assumptions as possible.
R was used to compute small scripts to simplify organization and handling of the data files. R was also used for our script to analyze dynamics for the network inference (see SC1A-write-up file). Since we used the graphical interface of COPASI, no further programming language was used.

Parameter fitting in COPASI may use automatic weighting methods to change the importance of data lanes. Since this led to rather bad fits, which completely ignored important dynamics, we manually adjusted this to be "1". This greatly improved the quality of our fits. We tried several algorithms already included in COPASI, such as particle swarm, simulated annealing and evolutionary programming to get the best fit result. Switching between methods helped us to avoid a bias introduced by using only one method. Furthermore, the possibility of being stuck in a local minimum was greatly reduced.

Stimuli were included as species that served as essential activators of their respective receptors. Below is the Standard Rate Law for Receptor Activation:

$$v = k1 * \frac{[stimulus]}{[stimulus] + Ka} * [substrate] - k2 * [product]$$

Where *k1*: rate of activation; *stimulus:* respective ligand; *Ka*: affinity of the ligand-receptor pair; *substrate:* unphosphorylated receptor; *k2:* rate of inactivation; *product*: phosphorylated receptor.

Because the presence of an inhibitor was applied via a global quantity, the influence of the inhibitors on subsequent reactions were modeled via a mathematical term, reducing respective reactions by at least 90%, when the respective global value. The equation is modified to:

$$v = k1 * \frac{[modifier]}{[modifier] + Km} * (1 - 0.9 * Inhibitor) * [substrate] - k2 * [product]$$

Where *k1*: rate of activation; *modifier:* active species; *Km*: affinity of modifier-substrate pair; *Inhibitor*: applied inhibitor; *substrate*: unphosphorylated species; *k2*: rate of inactivation; *product*: phosphorylated species.

We modified the relevant rate laws for the affected proteins to include the test-inhibitor and decrease the reaction rate when applied. Since the inhibitors applied in the experimental data appeared to have an effect on the initial concentration of the phosphorylated species, we corrected some of the species' initial concentration according to the test inhibitors. We used the same approach for the *in silico* sub-challenge, since COPASI is a versatile user interface for modeling.

We can only guess how much computational time was used for sub-challenges 2A and 2B, since we relied heavily on the local computer cluster for our parameter estimations. On average, each model may have spent up to 200 h on the cluster, often with several parallel fits.

It seemed necessary to us to build individual models for each cell line, which ideally was valid for all stimuli. For the parameter fitting we focused on the "Main" dataset but used the information about total protein concentration provided in the "Full" dataset to support our estimations. In some cases, such as MCF7 we noticed that the stimuli had little effect on the overall behavior of the cell line. In our parameter estimations we fitted against all experimental conditions at the same time. The inhibitors had been included as global quantities that significantly down-regulated the downstream reactions of the inhibitors' respective target.

If we could not represent dynamics that were clearly visible in the experimental data with our model, we checked for additional regulatory feedbacks not included in our networks for sub-challenge 1 or our models. One case was the internalization of the activated EGFR, which significantly improved the fits for phosphorylated EGFR species in the BT20 model. To validate our model we used the provided inhibitor data, which showed us that certain feedback loops and edges had to be improved. One such feedback loop of AKT on itself, which was not included in our model initially, significantly improved the reaction to the inhibitors.

Discussion

During the course of the competition, we noticed many times how different cell lines may be and how changed the signaling pathways are compared to healthy cells. Our approach of building one model for each cell line was a step in the right direction, but the models still need a lot of work and more experimental data to really describe the dynamics correctly.

Working with the *in silico* model taught us the difficulty of evaluating data without the possibility to access prior information from the scientific literature about the system at hand. Clearly, our model is able to reproduce the great majority of the dynamic trends in the original data, but might still lack a few causal edges to explain some of the more complex interactions. Therefore, our resulting model heavily depends on the quality of the underlying signaling network.

References

1. COPASI Version 4.10 (Build 55) (copasi.org)

SC2 Time-course Prediction: Team6

Summary

Predictions on real data are based on the network models learned in the network inference sub-challenge, using Ordinary Differential Equations (ODE) models and least squares method for parameter identification.

Introduction

A signaling network can be described by a system of ODEs, whose parameters can be used to describe its dynamic behavior after the interacting proteins and the type of these interactions are determined. One advantage of such systems is that, once identified, it provides a powerful tool for data simulation. Since we based our predictions of network structures in the network inference sub-challenge on ODE systems, we were then able to easily predict protein time-courses.

Methods

We identified stimulus-specific models such that for each protein, a module network was built considering a set of possible regulators and a pruning procedure based on an F-test was applied to remove redundant interactions. Then, the module networks were integrated in a single global network that was globally re-identified to obtain the final parametric estimation. The stimulus-specific ODE system with its set of parameters was used for prediction. Stimuli were considered as inputs in the network system acting on a set of upstream proteins opportunely selected according to prior biological knowledge. The overall approach is explained above in the SC1 network inference method description for Team6.

For each stimulus, the parameters identified independently for the four different inhibitory conditions were averaged using a weighted scheme that takes into account the effect of different inhibitors on kinase action. Weights were calculated as the ratio between the concentrations of the inhibited proteins (AKT under GSK690693 inhibitor, AKT and MEK under GSK690693+GSK1120212 inhibitor, FGFR1 and FGFR3 under PD173074 inhibitor) at time 0, prior to addition of the stimulus, and the concentrations under the DMSO condition, was calculated and used as Because GSK690693+GSK1120212 effect is the result of the combination of two inhibitors, individual GSK1120212 effect was computed subtracting the effect of GSK690693 alone. A similar approach was used to mimic the effect of test inhibitors.

Concerning simulated data, predictions are based on the network models learned in the *in silico* network inference challenge, where we associated a p-value to each edge, corresponding to its marginal effect on the goodness-of-fit of the system. To build the network, we selected all edges with a p-value lower than 0.05, corresponding to a good balance between network sparsity and goodness of the resulting fit, assessed by visual inspection of the temporal profiles. To generate temporal profiles, we integrated each equation of the system with a piecewise analytical solution, assuming constant regulators in each time step. The effect of the inhibitors was modeled by clamping protein activity to a minimal value for the entire experiment, while the stimuli were simulated by adding a constant term (1 for high stimuli, 0.01 for low stimuli) to the differential equation corresponding to the stimulated proteins. The fit to temporal data was carried out by minimizing the Residual Square Sums through an evolutionary algorithm, Covariance Matrix Adaptation - Evolution Strategies (CMA-ES, Hansen et al., Evolutionary Computation 2003).

The algorithm implementing the method was written in both R, C and MATLAB language, requiring additional packages R.matlab, Statistical Toolbox, Control System Toolbox, Optimization Toolbox and the MATLAB implementation of the CMA-ES algorithm (https://www.lri.fr/~hansen/cmaes_inmatlab.html#matlab). R code was used to organize input/output data and to generate some preliminary stats useful for the model fit. MATLAB and C code were used to implement definitive ODEs models and the approach to time-series prediction.

Discussion
Our ODE-based method enables to describe the topology and the dynamic properties of the considered protein networks. However, the choice of linear models, despite being appropriate for network topology reconstruction (as shown by our results in the network inference sub-challenge), might fail in capturing non-linear protein interactions, lowering prediction accuracy. Thereby, improvement in the prediction of protein abundances might be gained by using non-linear models, provided that the adopted models are a priori-identifiable and that the data are sufficiently informative to guarantee a posteriori identifiability.

SC2 Time-course Prediction: Team8

Summary
Gradient tree boosting is used to predict forward steps in time series data, represented as successive points under a Markov assumption.

Introduction
Our approach uses the traditional dynamic Bayesian network model, augmented to include gradient boosting regression to fit the response of each phosphoprotein. The salient features of this algorithm are (a) it learns a strong regression function from an ensemble of weaker regression trees, and (b) it re-weights the training data at each iteration to emphasize hitherto poorly modeled examples (Hastie et al., 2004). Once a model is trained, predictions at a given time step can be used as input to predict future time steps. The underlying methodology is based on an ensemble regression approach that can capture nonlinear interactions among covariates. Our approach was validated on known literature models (Xu et al., 2010) of signaling networks before applying to the challenge data.

Methods
Dynamic Bayesian networks ‘unroll’ Bayesian networks by making the Markovian assumption that phosphoprotein activity at each time point depends only on the values at the previous time point. Given this representation, we use a gradient boosting approach to fit a model that predicts activity for each phosphoprotein individually, then pools those predictions and uses them as

input for the next time step. Our method was identical for both the *in silico* and experimental components of the sub-challenge.

Several preprocessing steps were performed prior to learning the model. Each experimental condition was mean centered. Missing time points were mean imputed to facilitate comparison between experimental conditions. Multiple replicates were averaged together to produce one time series for each experimental condition. We explored training our model on both the response level and response rate, and found response level to have better performance. Parameters, including maximum tree depth, number of boosting rounds, and learning rate were set using a grid search and evaluated under cross-validation. No information about the network connectivity learned in Sub-challenge 1 was used. For the experimental dataset, no external data was incorporated into the model, and only the main dataset was used (not the full dataset).

Inhibitors were modeled using a perfect fixed-effects model (Spencer et al., 2012). The stimulus was not explicitly modeled. We dealt with stimuli in two ways: by grouping datasets across stimuli, and by training independent models for each stimulus. We found training separate models for each stimulus performed better than grouping across stimuli.

Our algorithm was implemented in Python, using the scikit-learn library for model development and testing. The pandas library was used for data manipulation and preprocessing.

Discussion

The gradient tree boosting approach performs well on both *in silico* and breast cancer RPPA data. Our performance on the real dataset was hindered by not integrating a biological prior into our model, however it was not immediately obvious how to do this. Predicted time series for target phosphoproteins consistently mimic the shape of the responses observed in the data as visualized in Figure 1. We expect ensemble models such as gradient boosting regression to gain increasing traction in this field due to their advantages over traditional regression approaches, including robustness to overfitting and ability to identify nonlinear effects.

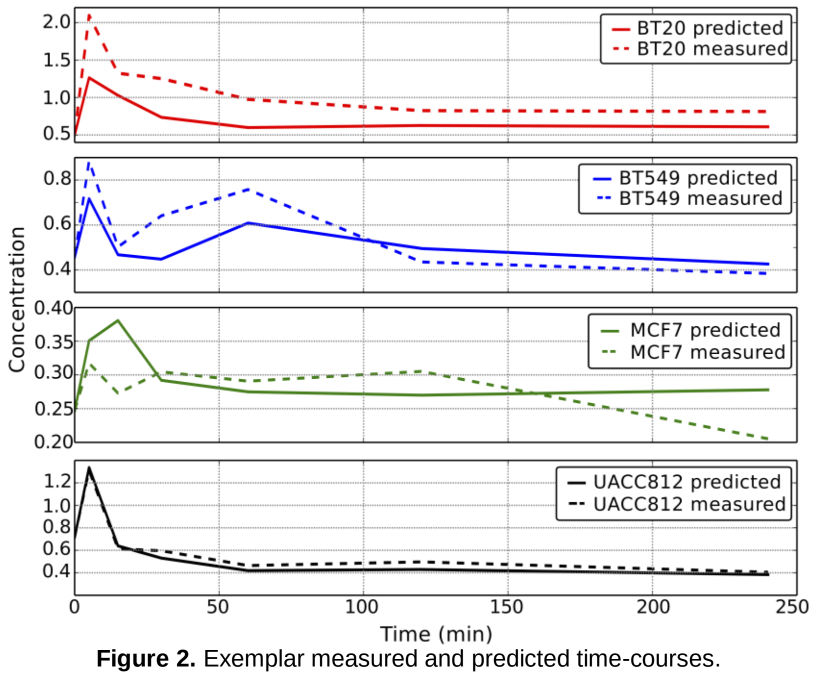


Figure 2. Exemplar measured and predicted time-courses.

References

4. T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, volume 1. 2004.
5. Simon E F Spencer, Steven M Hill, and Sach Mukherjee. Inferring network structure from interventional time-course experiments. *Annals of Applied Statistics* 9: 507-524.
6. Tian-Rui Xu, Vladislav Vyshemirsky, Am´elie Gormand, Alex von Kriegsheim, Mark Girolami, George S Baillie, Dominic Ketley, Allan J Dunlop, Graeme Milligan, Miles D Houslay, and Walter Kolch. Inferring signaling pathway topologies from multiple perturbation measurements of specific biochemical species. *Science Signaling*, 3(113):ra20, January 2010.

SC2 Time-course Prediction: Team10

Summary

Our method predicts the projection of phosphorylation levels under unseen inhibitors using truncated singular value decomposition based on Markov and stationarity assumption.

Introduction

Mass-spectrometry-based methods can be used to quantify dynamic changes in phosphorylated proteins over time and therefore help us to predict the phosphorylation networks. However, such experiments are expensive and time-consuming, making them unlikely to be extended to very large scale including perturbation on most proteins. Additionally, computational modeling becomes vitally important for generalizing the observed time-course network dynamics to unseen situations, which remains a challenging task. To solve this problem, we developed a protein phosphorylation dynamics prediction method using truncated singular value decomposition (SVD). Our method is based on stationary Markov assumption and uses a regression method comparable to Lasso regression. Any time-course data could be used as inputs (with or without inhibitors) to predict the perturbation effects under other inhibitors within the same cell line. Our method is highly accurate (as measured by the mean root-mean-square error (RMSE) in DREAM8) and efficient, making it an ideal algorithm scalable to genome-wide studies. For the DREAM8 datasets, our algorithm takes less than one second per cell line and inhibitor combination.

Methods

Our method predicts the projection of phosphorylation levels under unseen inhibitors using truncated singular value decomposition based on Markov and stationarity assumption. Figure 1 illustrates the workflow of our algorithm.

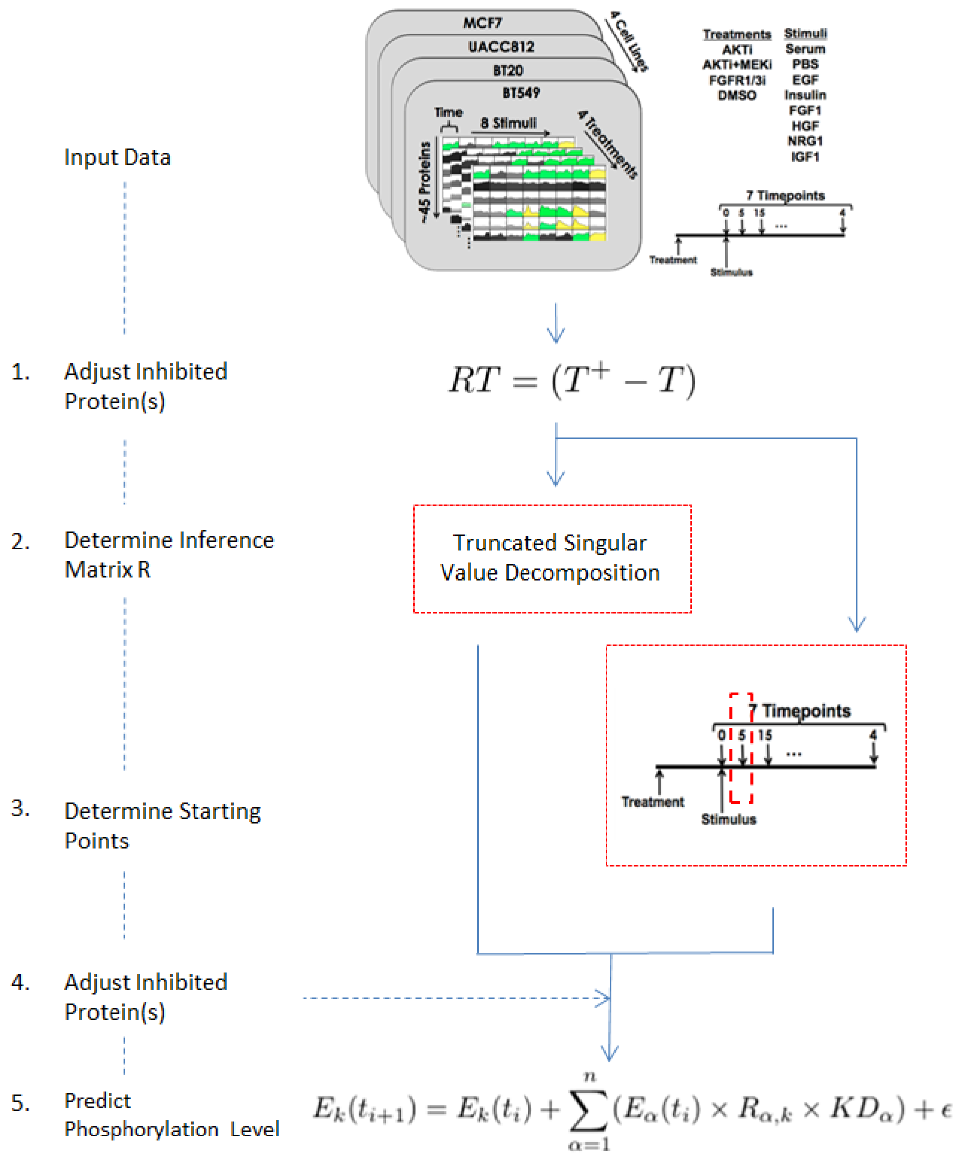

Input Data
MCF7
UACC812
BT20
BT549
4 Cell Lines
Time
8 Stimuli
4 Treatments
~45 Proteins
Treatments
AKTi
AKTi+MEKi
FGFR1/3i
DMSO
Stimuli
Serum
PBS
EGF
Insulin
FGF1
HGF
NRG1
IGF1
7 Timepoints
0 5 15 ... 4
Treatment
Stimulus
1. Adjust Inhibited Protein(s)
$RT = (T^{+} - T)$
2. Determine Inference Matrix R
Truncated Singular Value Decomposition
3. Determine Starting Points
7 Timepoints
0 5 15 ... 4
Treatment
Stimulus
4. Adjust Inhibited Protein(s)
5. Predict Phosphorylation Level
$E_k(t_{i+1}) = E_k(t_i) + \sum_{\alpha=1}^{n} (E_\alpha(t_i) \times R_{\alpha,k} \times KD_\alpha) + \epsilon$

Figure 1. Workflow of time-course prediction using truncated SVD.

Our time-course prediction method contains five steps: 1) Adjust activity level of inhibited protein(s). 2) Determine the inference matrix representing the influence level of protein *i* on protein *j* in a specific cell line without inhibitors. 3) Estimating the starting phosphorylation levels of the proteins. 4) Adjust activity level of protein(s) under unseen inhibitors. 5) Iteratively predict the phosphorylation level across a time course.

Assumption

We make first-order Markov and stationarity assumption [1, 2] that: every variable at a given time point t_i only depends on variables at the previous time point t_{i-1}. Furthermore, we assume that the value of a variable will not change without the influence of any other variables (including self-regulation). As a result, all the variables contributed are the value change of targeted variable. Equation 1 is the foundation of our method:

$$E_k(t_{i+1}) = E_k(t_i) + \sum_{\alpha=1}^{n}(E_\alpha(t_i) \times R_{\alpha,k} \times KD_\alpha) + \epsilon \qquad (\forall t \in [t_1..t_{T-1}],\ k \in [1..N]) \tag{1}$$

Where $E_k(t_i)$ represents the phosphorylation value for protein k at the i_{th} time point, $R_{\alpha,k}$ is the inference relationship factor indicating how much protein α will affect protein k, N is the number of proteins and ε is the error factor. KD_α is the knockdown factor that equals to 0 if α is targeted by the inhibitor or equals to 1 otherwise.

Inference Matrix R
Combining Equation 1 for all proteins, all time points and all inhibitors, we can form the following equation:

$$\begin{pmatrix} R_{1,1} & R_{2,1} & \cdots & R_{N,1} \\ R_{1,2} & R_{2,2} & \cdots & R_{N,2} \\ \vdots & \vdots & \ddots & \vdots \\ R_{1,N} & R_{2,N} & \cdots & R_{N,N} \end{pmatrix} \begin{pmatrix} E_1(t_1) & \cdots & E_1(t_{T-1}) \\ E_2(t_1) & \cdots & E_2(t_{T-1}) \\ \vdots & \ddots & \vdots \\ E_N(t_1) & \cdots & E_N(t_{T-1}) \end{pmatrix} = \begin{pmatrix} (E_1(t_2) - E_1(t_1)) & \cdots & (E_1(t_T) - E_1(t_{T-1})) \\ (E_2(t_2) - E_2(t_1)) & \cdots & (E_2(t_T) - E_2(t_{T-1})) \\ \vdots & \ddots & \vdots \\ (E_N(t_2) - E_N(t_1)) & \cdots & (E_N(t_T) - E_N(t_{T-1})) \end{pmatrix} \tag{2}$$

For the convenience of description, we assign an abbreviation for each matrix in Equation 1:

$$RT = (T^+ - T) \tag{3}$$

Where each element in T is the observed value for antibody protein at a given time point, while corresponding element in T^+ is the observed value for the same protein at the next time point. Both T and T^+ can be obtained from the input data and R is the unknown inference matrix we interested.
The problem can be solved by:

$$R = (T^+ - T)\ \ T^{-1} \tag{4}$$

Since T is not a square matrix, it is not invertible. T^{-1} in Equation 4 is the pseudo-inverse of matrix T.
There are numerous pseudo-inverse methods, such as singular value decomposition (SVD) [3], QR method, L1-regulation and L2-regulation. In order to handle the noise ε mentioned in Equation 1, we used truncated SVD [4, 5], a variant of SVD, in our approach. Truncated SVD allowed us to minimize the effect of noise and then calculate the pseudo-inverse matrix. In truncated SVD, the first step is to decompose the target matrix into:

$$T = U\Sigma V^T \tag{5}$$

Where U is an $N\times N$ unitary matrix, Σ is $N\times T$ diagonal matrix and V^T is the transpose of a $T\times T$ unitary matrix V. A convention is to order the diagonal matrix Σ in a decreasing order, and the diagonal entries of Σ are known as the singular values of original matrix T. Elements on the diagonal matrix Σ are non-negative real numbers.
Since there may be zero-value elements on the diagonal matrix Σ, its inverse matrix Σ^{-1} does not exist. In traditional SVD, the Σ^{-1} is calculated using following equation:

$$\Sigma^{-1}(i,i) = \begin{cases} \frac{1}{\Sigma(i,i)} & (\Sigma(i,i) > 0) \\ 0 & (\Sigma(i,i) = 0) \end{cases} \tag{6}$$

Note that this is a pseudo-inverse operation in that $\Sigma\Sigma^{-1} \neq I$.
Since the small values caused by noise in diagonal matrix may result in extremely large values in Σ^{-1}, the noise is emphasized and dominates the inverses diagonal matrix using traditional SVD. To deal with the noise, a step in our method is the truncated SVD. A threshold is defined and elements of the diagonal matrix Σ with values smaller than this threshold will be set to zero in Σ^{-1}, so Equation 6 in truncated SVD is written as:

$$\Sigma^{-1}(i,i) = \begin{cases} \frac{1}{\Sigma(i,i)} & (\Sigma(i,i) \geq Threshold) \\ 0 & (\Sigma(i,i) < Threshold) \end{cases} \tag{7}$$

Finally, based on Equations 4, 5 and 7, the inference relationship matrix can be calculated as:

$$R = (T^{+} - T) \quad V\Sigma^{-1}U^{T} \tag{8}$$

Determine Starting Points
Since we have no access to the phosphorylation levels at the first time point, a protein's initial phosphorylation level under an unseen inhibitor was set to its average value under all training inhibitors. Proteins targeted by one inhibitor could have very different phosphorylation levels compared to those under other inhibitors in the training data. We removed the data points for inhibited proteins in the training data when estimating the starting points. Due to the same reason, antibodies targeted by the new inhibitors are excluded in the prediction (and follow up evaluations).

Adjust the Influence of Inhibited Antibodies
The activity of the phosphorylation proteins targeted by inhibitors should be much lower than it is calculated in the inference matrix *R* in Equation 4. Therefore, the contribution of inhibited proteins towards other proteins will be reduced to 0 during the prediction.

$$R_{i,j} = \begin{cases} 0 & \text{Antibody i is inhibited} \\ R_{i,j} & \text{otherwise} \end{cases} \tag{9}$$

Note that the activity levels of a protein under different inhibitors may be combined together in Equation 2, 3, 4 and 8.
Equation 9 is applied to both inferring the R matrix and predicting the levels of phosphorylation under unseen situation.

Iteratively Predict the Phosphorylation Level across a Time-course
After estimating the phosphorylation values at the first time point and adjusting the inference matrix based on the inhibitors, Equation 1 can be used iteratively to predict the breast cancer time-course perturbation in the rest time points.

Execution Time
Our method is efficient, which only takes less than one second to predict the time-course perturbation for one cell line, inhibitor combination.

Discussion
In the DREAM 8 challenge, we presented a time-course prediction method based on truncated SVD. Our method ranked number 1 in the experimental timecourse prediction challenge, as well as in the combined results of the experimental and *in silico* data sub-challenges. Our method could potentially be improved on two aspects. First, we could use alternative methods to determine the starting point. The method we used to determine the starting point highly depended on the assumption that inhibitors will not affect the phosphorylation levels of non-targeted proteins, however this is not always correct. Second, we could reduce--but not remove--the influence of inhibited antibodies. Inhibitors usually reduce the activity level of its target differently, by 2 to 10 fold. With more information it is possible that the effects could be modeled more accurately.

References

1. Friedman, N., K. Murphy, and S. Russell. Learning the structure of dynamic probabilistic networks. *Proceedings of the Fourteenth conference on Uncertainty in Artificial Intelligence*. 1998: Morgan Kaufmann Publishers Inc.
2. Husmeier, D., Sensitivity and specificity of inferring genetic regulatory interactions from microarray experiments with dynamic Bayesian networks. *Bioinformatics*, 2003. **19**(17): p. 2271-2282.
3. Golub, G.H. and C. Reinsch, Singular value decomposition and least squares solutions. *Numerische Mathematik*, 1970. **14**(5): p. 403-420.
4. Hansen, P.C., Truncated singular value decomposition solutions to discrete ill-posed problems with ill-determined numerical rank. *SIAM Journal on Scientific and Statistical Computing*, 1990. **11**(3): p. 503-518.
5. Henry, E. and J. Hofrichter, Singular value decomposition: application to analysis of experimental data. *Essential Numerical Computer Methods*, 2010. **210**: p. 81-138.

SC2 Time-course Prediction: Team28

Summary

The mean value of the inhibitor-free training perturbations was used to predict the time-course of phosphorylation kinetics for all test inhibitors.

Method

We chose a straightforward, simple, and computationally inexpensive method for predicting the time course of phosphoprotein levels. We took the arithmetic unweighted mean of the untreated (no inhibition) training data. The submission files were manually curated using Excel. The model for Part I was not taken into account for these predictions. There was no prior data or cross-validation used.

Discussion

We believe that training data should be considered as a ‘prior’ when trying to make predictions of unknown inhibitors, as new time course predictions may remain close to the training data for most phosphosites. In future work, it would be useful to determine the ideal weights of the prior training data in order to most accurately predict new dynamics.

SC2 Time-course Prediction: Team34

Summary

Time series data prediction was carried out using consensus networks and generalized linear models.

Introduction

Predicting time-courses under perturbations in the inferred signaling network is important to understand the function of a network. In this study, we had designed a novel method to integrate networks from three different inference algorithms. We used the Monte Carlo algorithm to handle replicates in the input data and generate the edge score for each inferred network. The networks were further evaluated with existing biological pathways from KEGG to determine the algorithm accuracy and assign method weights. We calculated the edge confidence scores as the sum of products of edge scores and method weights. Only edges with high confidence scores were used to construct consensus networks. Then we applied generalized linear models (glm) to carry out the prediction. In the glm, the dependent variables (Y) are the child nodes in the network and the independent variables (X) are the parent nodes.

A robust network is the foundation of the time-course prediction. We had designed a novel approach to integrate networks from three different inference algorithms. Each edge in the network was assigned a confidence score and a cut-off threshold was determined based on the distribution of the confidence scores so that only reliable interactions will be kept in the network, which could improve the accuracy in our prediction. Generalized linear models extend ordinary regression to non-normal response distributions. This flexibility allows us to model the nonlinear relationship between nodes.

Methods
We used the data from all cell lines, stimuli, time points and inhibitors. For the experimental sub-challenge, we used “Main” dataset only. The data were log-transformed, and we did not impute missing data. The Monte Carlo algorithm was used to handle replicates with as.longitudinal function in the R longitudinal package. The function allows repeated measurements, irregular sampling, and unequal temporal spacing of the time points. In each resampling, one zero time data point was used. The initial networks were calculated with three different algorithms: Dynamic Bayesian Networks (DBN), Max-Min Hill Climbing (MMHC), and Graphical Gaussian Model (GGM). Edges with high confidence scores were selected to build the consensus network.

Generalized linear models were used to predict the time-series change under each inhibitor condition. For each consensus network, the node for the inhibitor target was excluded in the model. The initial values for building glm were selected from re-sampling of replicates with the Monte Carlo algorithm. In the glm, the dependent variables (Y) are the child nodes in the network and the independent variables (X) are the parent nodes. The predictions were carried out for each re-sampling of replicates. Mean predicted values from the re-sampling of replicates were used in the submission. We did not use any external information in either portion of this time-course prediction challenge. We implemented our algorithms in R.

Discussion
It is an extreme challenge to infer the protein kinase signaling networks from the large scale functional proteomics experiment data. The difficulties include 1) The signaling network is a dynamic system which changes in different conditions and time points; 2) The protein interactions are very complex and non-linear; 3) The heuristic approaches modeling the network are subject to generate high false positive and negative results. To deal with these issues, we integrated networks from multiple network inference algorithms into a consensus network structure. Different algorithms could detect different interaction signals in the network. The inferred networks were evaluated with KEGG pathways to determine the accuracy for each algorithm. The interactions consistent within and between methods were assigned high confidence score and selected for building consensus networks. The result should be more accurately reflecting the signaling process than networks from each individual algorithm. The reliable consensus networks allow us model the perturbation accurately with generalized linear models. The mean values from re-sampling of replicates with the Monte Carlo algorithm reduce the variability and contribute to the accuracy.

References

1. R Development Core Team (2008). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, http://www.R-project.org.
2. Christian Robert and George Casella. Introducing Monte Carlo Methods with R. Use R. Springer, 2010. ISBN 978-1-4419-1575-7.

SC2 Time-course Prediction: Team42

Summary

For the experimental data challenge, protein expression data for a given cell line, stimulus and time point are averaged across inhibitors. For the *in silico* data challenge, correlation was used to find a sparse causality network and fit a "pseudo-linear" system identified by minimizing mean squared error.

Introduction

We applied two different approaches to experimental and *in silico* data. Prior to developing a model for either dataset, we ran unsupervised exploratory analyses to ascertain global trends in the phospho-proteomic expression. Notably, we applied a Markov chain Monte Carlo matrix factorization algorithm, CoGAPS [1], to infer dominant temporal patterns in the global phospho-proteomic profiles that were specific to each stimulus or inhibitor. Our analysis showed that most proteins in the experimental data are impacted more by stimuli and presence of inhibitor, than by the particular inhibitor selected. We therefore used an averaging model that incorporated information from each inhibitor in the training set. This model was the co-best performer for the experimental data. Further refining this model to filter the effect of direct therapeutic targets improved the accuracy in the Collaborative Round.

On the other hand, the phosphoprotein time-courses in the *in silico* data depend equally on both the stimuli and inhibitors. Therefore, we treated the *in silico* data as an outcome of a nonlinear system. We approximated the nonlinear system by a linear model combined with a nonlinear term that guarantees the bounds of the expressions. This model was applied only to proteins directly altered by the therapeutic or immediate downstream nodes in a sparse network.

Methods: Experimental

Samples have strong correlation, regardless of stimulus or inhibition

Preliminary correlation analyses performed in R showed that all samples have large Pearson correlations, regardless of stimulation or inhibition (Fig. 1). Samples treated with AKT inhibitor (GSK690693) alone or in combination with the MET inhibitor (GSK1120212) cluster distinctly from samples that were either untreated or treated with the FGFR inhibitor (PD173074). However, the clustering does not clearly delineate time points.

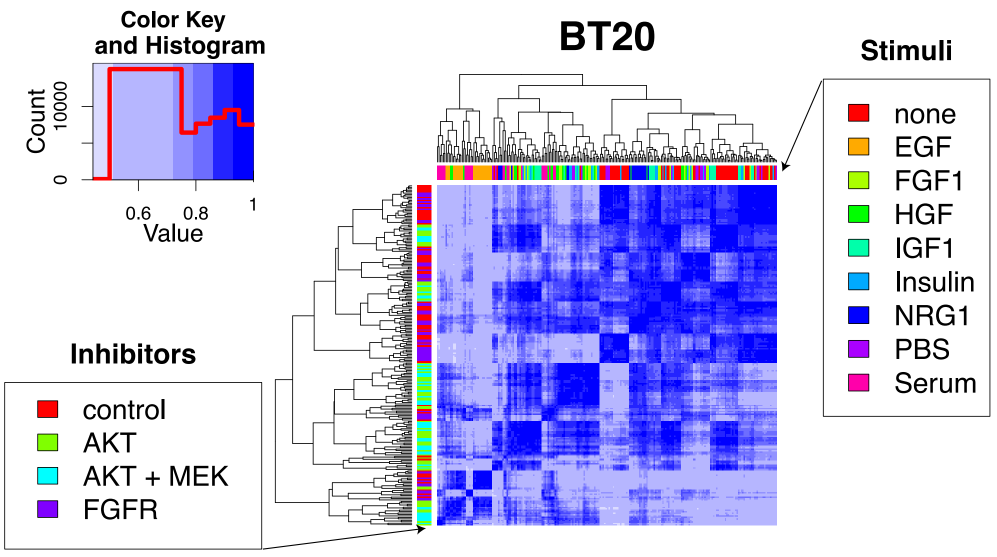


Figure 1. Heatmap of Pearson correlation coefficients computed for samples from each stimulus, inhibitor, and time point measured for the BT20 cell line. Darker blue indicates stronger correlation, according to the color key. Rows are colored according to the inhibitor and columns according to the stimulus, as indicated in the corresponding legends.

CoGAPS analysis suggests dominant temporal signal of stimuli in most proteins

To distinguish the temporal dynamics, we selected the subset of training data with FGFR inhibition with PD173074 and control, stimulated with PBS, serum, or FGF. We then applied the CoGAPS matrix factorization [1], to associate the expression of each phosphoprotein to a linear combination of temporal patterns across stimuli or inhibitors. This algorithm takes as input the phospho-proteomic data matrix, an estimate of uncertainty in the data, and number of dimensions. We used the replicates at time zero to estimate the relative uncertainty in each phosphoprotein, and assumed the uncertainty at each data point was given by that uncertainty estimate times the observed phosphoprotein expression. We selected 3 patterns for analysis, based upon the robustness estimates described in [2].

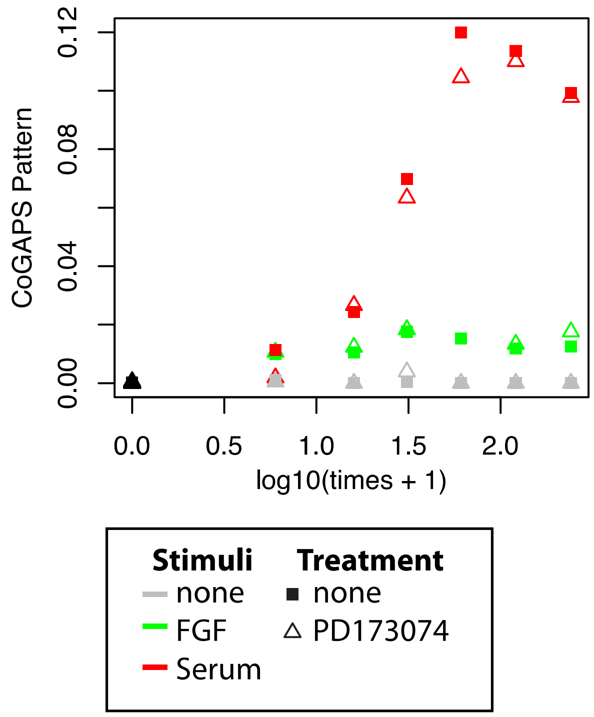


Figure 2. One of the temporal patterns in global phosphoprotein time-courses inferred with CoGAPS for BT20 cell lines distinguishes distinct protein upregulation for each stimulus.

We plotted the patterns inferred for the cell line BT20, separated by stimulus or inhibitor, as a function of time (Fig. 2). These patterns distinguished differences in global phosphoprotein response between stimuli. However, inhibitors had a smaller effect on phosphoprotein expression.

Averaging model

As a result of the clustering and CoGAPS analyses, we hypothesized that the time-courses of only a few direct target proteins were affected over this time-course. Therefore, we used an averaging model for a given time and stimulus. Specifically, we applied linear interpolation to estimate the state of each phosphoprotein for each stimulus and inhibitor in the training set at the output times. We then computed the average phosphoprotein state across all inhibitors for the given stimulus, cell line, and time as the modeled time-course for all inhibitors in the test set. We replaced replicate measurements by a single average, prior to averaging across inhibitors.

This preliminary model did not require any estimated network or parameters. Analyses were performed with R, using the package CellNOptR to load MIDAS files. Data was only pre-processed with temporal interpolation and averaging across replicates. These analyses were performed only using the Main data, and did not consider the Full dataset. We did not use external information for this submission.

Methods: *in silico* data
Most dynamical systems can be approximated locally by a linear system. Therefore, we assumed that the time evolution of a state of each phosphoprotein (*x_i* for the ith phosphoprotein) was a linear combination of the states of connected proteins and all inputs (e.g. stimuli and inhibitors, *u_{ik}*) as follows:

$$\dot{\tilde{x}}_i = \left(\sum_j A_{ij}\tilde{x}_j + \sum_k b_{ik}u_{ik}\right)(1-\tilde{x}_i)\tilde{x}_i \qquad (1)$$

Here, as the pre-processing, we loess smoothened each phosphoprotein time-course x_i in the training data. To standardize notation across proteins, we then normalized the smoothed data to rescale each phoshoprotein measurement such that $x_i \to \tilde{x}_i \in [0,1]$. The multiplicative terms in eq. (1) represented an alteration to the standard linear system, which introduce fixed points at zero and one to reflect technical bounds in measurement of phosphoprotein expression. *$A=[A_{ij}]$* represented the internal dynamics of the system and *$b=[b_{ij}]$* the effects of the stimuli and inhibitors. We also assumed that the inputs u_k represent both stimulus and inhibition. If inhibitor *k* inhibits protein *i*, u_{ik}=-1 and was zero, otherwise. To model the stimuli, we assumed u_{ik} is nonzero if stimulator *k* activates protein *i*. If the amount of a stimulus was indicated as low, we set u_{ik} to 0.5 and 1 when high.
We applied a variational approach to estimate parameters A_{ij} and b_{ij} for the sparse, "pseudo-linear" system of equations given by eq. (1). Specifically, we sought values of A_{ij} and b_{ik} to minimize the following cost function:

$$J(A,b,y) = \sum_t \sum_i \left(\tilde{x}_i(t,A,b) - \tilde{y}_i(t)\right)^2 \qquad (2)$$

where *t* indexed the measured time points and $\tilde{y}_i(t)$ represented the measurement of protein *i* at time *t* rescaled to be between zero and one. $\tilde{x}_i(t,A,b)$ was the result of solving eq. (1) for a given value of *A* and *b* using Matlab's ode45 from the initial condition for no inhibition for each stimulus To further ensure the bounds on $\tilde{x}$, we enforced a threshold such that if when it exceeds the maximum for that stimulus learned from the data, we report the maximum value of $\tilde{y}$ for that stimulus. We implemented our algorithm in Matlab, using the Data Rail, Optimization, and Curve Fitting toolboxes.

We enforced sparsity in our network (A) to avoid over-fitting. We did not participate in the first challenge. However, we still needed a network to find the non-zero values of *A* estimated to optimize eq. (2). Although direct application of correlation may find the zero values in A, it cannot reflect the causality essential to inference in this application. For example, e.g. $A_{ij} \neq 0, A_{ji} = 0$ means the protein *j* affects protein *i*'s dynamics, but does not imply that protein *i* likewise contributes to the dynamics of protein *j* at future time points. Therefore, we started from the directly stimulated proteins and assumed that a protein *j* has a causal link ($A_{ij} \neq 0$) if it is significantly correlated to protein *i*. Thereafter, we discarded the stimulated and inhibited proteins and calculated the Spearman correlation between targets of stimulated (or inhibited) proteins and the remaining proteins.

In order to make a sparse network with this algorithm, the significance level of the correlation becomes a parameter in our model. We chose the significant level of the correlation to have a p-value below 0.01 (i.e. 0.001) so that each protein is influenced by at most three other proteins. Specifically, p-value of 0.01 led too many edges for our optimization to converge and 0.0001 led to no edges. As a result, we had 33 causal edges (number of $A_{ij} \neq 0$ for $i \neq j$) and for each protein we need to estimate the rate of decay (i.e. 20 A_{ii} parameters). For two stimuli and three inhibitions, we need five parameters (i.e. $b_{5,5}$, $b_{8,8}$, $b_{12,12}$, $b_{12,21}$, $b_{10,22}$, $b_{20,22}$). So, we had a total of 58 parameters. We only had training data for inhibition of AB10, AB12, and AB20. Therefore, we assumed that all the b_{ii}'s are equal for all other proteins. We estimated this constant b_{ij} with the median the obtained for the b_{ii}'s for AB10, AB12, and AB20 when optimizing eq. 2.

We estimated the parameters to minimize average mean squared error (eq. 2) by applying Matlab's fminsearch. In general, we iteratively updated A_{ij}, b_{ij} until convergence. Since the optimization was time-consuming when performed for all proteins simultaneously, we wanted to divide the optimization into a partial optimization for each protein. In so doing, the biggest hurdle was the interactions among the genes, i.e. $\tilde{x}_i$ is a function of $\tilde{x}_j$, where $i \neq j$. We used an approximation to make the optimization for each gene independent. Specifically, we used an iterative approach to solve for $\tilde{x}_i(t)$ in equation (1) in a given iteration based upon the values of $\tilde{x}_j(t)$ obtained in the previous iteration. In this step, we estimated parameter values in parallel for each protein *i* by applying Matlab's fminsearch function to optimize equation (2). We iterated this procedure until the parameters converge. The iterative procedure was initialized using the smoothed, interpolated phosphoprotein time-course from the measurements $\tilde{y}_j(t)$. Because the computation time for parameter estimation was long, we only used resubstitution error as the measure of the improvement of the algorithm prediction.

Discussion

Time-course prediction of protein expression in response to inhibition is essential to infer the molecular mechanisms responsible for evolving therapeutic response in cancer. Over a short time course, most phosphoprotein time-courses are dominated by stimuli. As a result, we obtained accurate prediction of inhibitor response by computing an average of inhibitor response in the training set for a given time and stimulus. We suspect multiple factors contribute to this observation. First, signal in the experimental data may be confounded by noise in the RPPA array. Secondly, factors more than phosphoproteins measured with the array may play roles in the regulation of these proteins. Finally, the drugs may kill the cells, leaving cell populations with distinct phosphoprotein profiles unrelated to inhibitor silencing.

We hypothesized that encoding network structure and the appropriate response in sub-populations of cells remaining after treatment may further improve estimates of the time-course in inhibitor targets and their direct targets on regulatory networks. Therefore, we applied the *in silico* approach to the model the time-courses of a few, phospho-proteomic measurements. We used the canonical signalling pathways from cellsignaling.org to distinguish inferred molecular targets from upstream regulators, to determine non-zero terms A_{ij} in eq (1). However, these attempts increased the RMS error.

To understand the increase in the RMS error with the ODE model, we analysed the modelled phosphoprotein time-course of the direct therapeutic targets. AKT inhibition with GSK690693 in the training data showed an unanticipated increase in phospho-AKT expression in response to treatment (Fig. 3). The observed increase in phospho-AKT expression with AKT inhibition is consistent with the hypothesized mechanism of action for this inhibitor, and does not necessarily imply a commensurate increase in downstream targets of AKT. Based on this, we hypothesize that using this signal to infer coefficients for inhibitor effects (b_{ik} in eq. 1) may have lead to significant errors in modeling the effect of other inhibitors on their targets, which propagated through the network to all phosphoproteins. Solving a differential equations model with parameters derived from the appropriately filtered training data may still improve the accuracy of predicted phosphoprotein time-courses.

With this insight, we modified our approach in the collaborative round of the challenge. Specifically, we applied our averaging model to all proteins, except phospho-AKT. For this protein, we used the interpolated values from the FGFR inhibitor (PD173074). Just filtering this unanticipated increase in AKT expression with AKT inhibition substantially decreased the z-score (-3.37 in the averaging model, and -3.79 in the new model). The mean RMS error also dropped from 0.50 to 0.45. This is lowest RMS error reported on any leaderboard.

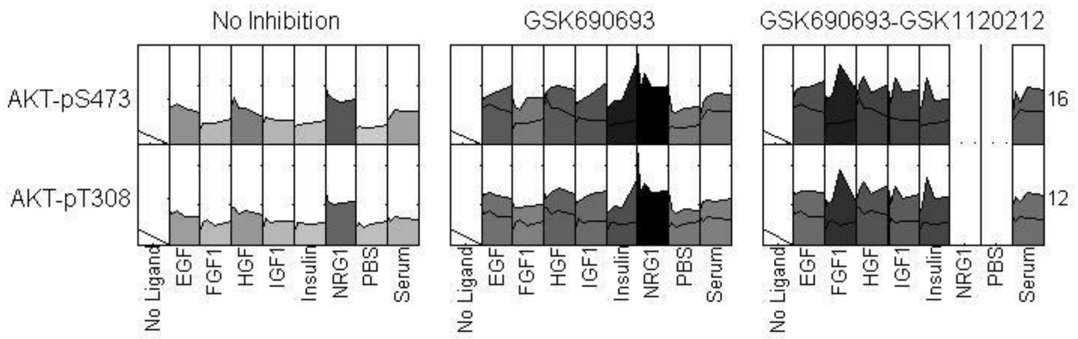


Figure 3. AKT expression in BT20 without inhibition and with AKT inhibition with GSK690693. The black lines in the AKT inhibition plots are the "no-inhibition" control.

References

1. Fertig, E.J., et al., CoGAPS: an R/C++ package to identify patterns and biological process activity in transcriptomic data. Bioinformatics, 2010. **26**(21): p. 2792-3.
2. Bidaut, G., et al., Determination of strongly overlapping signaling activity from microarray data. BMC Bioinformatics, 2006. **7**: p. 99.

SC2 Time-course Prediction: Team43

Summary

Partial least squares and a prior knowledge network are used to predict time-courses.

Introduction

Partial least squares (PLS) is a projection method where the independent variables, represented as the matrix **X**, are projected onto a low dimensional space. PLS uses both independent variables **X** and dependent variables **Y**. Partial least squares is a regression method that does not rely on normal data and can handle more variables than observations (Geladi and Kowalski 1986; Opiyo and Moriyama 2007). In this challenge I was able to predict experimental and *in silico* time series experiments PLS models. PLS allowed me to obtain the results without using test datasets. This was possible because PLS algorithm is very robust that it allows for modeling datasets that has more variables than observations. Secondly, PLS methods are not affected by normality of the data.

Methods

The approach used was PLS methods. The models for this challenge were created using information from experimental breast cancer proteomic dataset description. In addition, we used information from AKT Signaling Pathway found BioCarta website (http://www.biocarta.com/pathfiles/h_aktPathway.asp) to define network connections. The information provided in Table 1 was used to select datasets used to train predict phosphoprotein time-courses under 5 test inhibitors presented in Table 2.

Discussion

I used PLS method and Main dataset plus information from BioCarta and I was able to make good predictions. This shows models created by PLS methods are robust to make predictions without test datasets.

Table 1. The information used for building the models

Inhibitor	Test inhibitor Target (s)
AKT	mTOR1, mTOR2
AKT/MEK	HER2, EGFR
FGFR1	N/A
AKT	EGFR
AKT/MEK	BCR-ABL, SRC family kinase
AKT/MEK	IGFR/INSR

Table 2. Inhibitors and Test Inhibitor Target (s) provided in Sub-challenge 2A

Inhibitor	Test inhibitor Target (s)
Testinhib1	EGFR, HER2
Testinhib2	BCR-ABL, SRC family kinases
Testinhib3	mTOR1, mTOR2
Testinhib4	EGFR
Testinhib5	IGFR, INSR

Table 3. The datasets used for each cell line and the dataset used for prediction

Cell line/inhibitor	Inhibitor
BT20-Testinib1	TR:PD173074:Inhibitors
BT20-Testinib2	TR:GSK690693_GSK1120212:Inhibitors
BT20-Testinib3	TR:GSK690693:Inhibitors
BT20-Testinib4	TR:GSK690693_GSK1120212:Inhibitors
BT20-Testinib5	TR:GSK690693_GSK1120212:Inhibitors
BT549-Testinib1	TR:PD173074:Inhibitors
BT549-Testinib2	No Inhibitors
BT549-Testinib3	TR:GSK690693:Inhibitors
BT549-Testinib4	No Inhibitors
BT549-Testinib5	No Inhibitors
MCF7-Testinib1	TR:PD173074:Inhibitors
MCF7-Testinib2	TR:GSK690693_GSK1120212:Inhibitors
MCF7-Testinib3	TR:GSK690693:Inhibitors
MCF7-Testinib4	No Inhibitors
MCF7-Testinib5	TR:GSK690693_GSK1120212:Inhibitors
UACC812-Testinib1	TR:PD173074:Inhibitors
UACC812-Testinib2	TR:GSK690693_GSK1120212:Inhibitors
UACC812-Testinib3	TR:GSK690693:Inhibitors
UACC812-Testinib4	TR:GSK690693_GSK1120212:Inhibitors
UACC812-Testinib5	TR:GSK690693_GSK1120212:Inhibitors

References

1. Geladi, P. and B. R. Kowalski (1986). Partial least squares regression: A tutorial. *Anal. Chim. Acta*. 185: 1-7.
2. Opiyo, S. O. and E. N. Moriyama (2007). Protein family classification with partial least squares. *J Proteome Res* 6(2): 846-853.
3. Wehrens, R. and B. Mevik (2007). pls: Partial Least Squares Regression(PLSR) and Principal Component Regression (PCR). R package version 1.2-1.

SC2 Time-course Prediction: Team44

Summary

For a given protein, cell line, stimulus and time point, training data were averaged across inhibitors. This simple approach outperformed more complex Bayesian network approaches.

Introduction

In silico reconstruction of the signaling networks from perturbation data is a critical and still challenging task for better understanding of the biological processes. Different methods have been used to approach this problem that include linear[1], Monte Carlo[2] and dynamic Bayesian networks[3]. Here we applied three distinct formulations of the dynamical Bayesian graphical model to reconstruct the time-course abundances of a particular set of proteins, using the HPN-DREAM dataset for training. While more complex models have more frequent parameters that cannot be estimated due to the limited size of the training dataset, simpler models exhibited significantly better performances. We also developed a simple *Inhibitor Independent* model that could overcome the other methods by ignoring small effect of the inhibitors.

Method

We used the whole discrete time series of the main dataset, with no particular preprocessing. Missing data were omitted, and the models were trained with the available data. For the replicates we kept only the average value. The protein-protein interaction and signaling networks of the selected proteins were extracted from two databases (STRING and Ingenuity IPA), and the inhibitor targets were incorporated as an external data.

Let S be the set of all inhibitors. For learning and validating different algorithms, we created the following framework: The same process was executed for each inhibitor *x* separately. First, the time series data of three different cell lines (BT20, MCF7 and UACC812) treated with inhibitors of *S*- {*x*} was used to train each model. Next using the data of the same cell lines treated with *x* was used to measure the accuracy of the results generated by each model, by Mean Squared Error (MSE) and also Pearson Correlation Coefficient (PCC).

First we developed our *in silico* network reconstruction method based on the dynamical Bayesian graphical models, since they are widely used in modelling causal interactions[4]. Consider $Z_t = (z_t^1, z_t^2, ..., z_t^n)$ to be the abundances of proteins at time *t*, while *n* is the number of proteins. The general equation of the dynamical Bayesian graphical models is[4]:

$$P(Z_t \mid Z_{t-1}) = \prod_{i=1}^{n} P(z_t^i \mid Pa(z_t^i)), \tag{1}$$

Where $Pa(z_t^i)$, the parents of z_t^i, is the set of all proteins of the dynamical network that can affect on z_t^i between the time *t - 1* to *t*. Distinct formulations of this equation result in different statistical models that are listed below.

Vector Auto-regressive. This model is based on the following formulation:

$$P(Z_t \mid Z_{t-1}) : N(W Z_{t-1}, \Sigma), \tag{2}$$

where *N*(.) denotes the normal distribution. In this model, the matrix $W_{n\times n}$ represents the causal interaction network. To eliminate the effects of one protein into the others (i.e. where the cell line is treated with that protein inhibitor), we set the corresponding row of *W* to zero.

*Sparse Vector Auto-regressive***.** In the sparse approach, we assumed that the interaction matrix *W* is sparse (many elements are zero). This assumption leads to the simpler model: the absolute values of the elements in *W* are exponentially distributed with constant mean λ.

$$\begin{aligned} &P(Z_t \mid Z_{t-1}; W) : N(W Z_{t-1}, \Sigma) \\ &|w_{ij}| : \exp(\lambda). \end{aligned} \tag{3}$$

*Kalman Filter***.** This is more complicated model but with less parameters to be estimated. *A* vector of hidden variables H_t of length *m* represents the dynamics of the time series at time *t*. To reduce the number of parameters, we assumed that the hidden variable are less frequent than the proteins ($m<n$). The following equations represent the parent-child relationship:

$$\begin{aligned} &P(H_t \mid H_{t-1}) : N(A H_{t-1}, \Sigma_1) \\ &P(Z_t \mid H_t) : N(B H_t, \Sigma_2). \end{aligned} \tag{4}$$

Causal interaction among the proteins can be obtained through the matrices $A_{m\times m}$ and $B_{n\times m}$. More precisely, the matrix $W_{n\times n} = BAB^{-1}$ denotes the causal relationships.
When applied to the real data, we found the simpler methods are exhibiting better results. Hence we developed a surprisingly simple method (*Inhibitor Independent*) that for each protein *p*, time *t*, cell line *l*, inhibitor *i* and stimuli *s,* predicts the protein abundance matrix as $A_{p,t,l,i,s} = \frac{1}{|S|}\sum_{j\in S} A_{p,t,l,j,s}$. The accuracies of different methods are provided below.

Method	Vector auto-regressive	Kalman Filter	Sparse Regression	Inhibitor Independent
MSE	29.40	28.90	5.07	1.47
PCC	0.53	0.41	0.82	0.90

Discussion

By ignoring presence of inhibitors, our Inhibitor Independent method could surprisingly exhibit the best results for the experimental timecourse prediction sub-challenge among all the competitors, suggesting the minor effects of the inhibitors in the global network structure and time-course protein abundances. Our results also suggest the mean values of time series can be used as a suitable prior for other Bayesian methods.

References

1. Knapp, B. & Kaderali, L. Reconstruction of Cellular Signal Transduction Networks Using Perturbation Assays and Linear Programming. *PLOS ONE* **8**, e69220 (2013).
2. Bender, C. *et al.* Inferring signalling networks from longitudinal data using sampling based approaches in the R-package 'ddepn'. *BMC Bioinformatics* **12**, 291 (2011).
3. Hill, S.M. *et al.* Bayesian inference of signaling network topology in a cancer cell line. *Bioinformatics* **28**, 2804-2810 (2012).
4. Murphy, K.P. *Machine learning: a probabilistic perspective*. (The MIT Press, 2012).
5. Bishop, C.M. & Nasrabadi, N.M. *Pattern recognition and machine learning*, Vol. 1. (Springer New York, 2006).

Supplementary Note 3: Visualization Sub-challenge (SC3) - BioWheel

Video and Tutorial: http://youtu.be/i9f6AYaIoOE
HPN-DREAM BioWheel Visualization Tool: http://dream8.dibsbiotech.com

With the advances in high-throughput technologies, the ability to categorize, analyze, and visualize “big data,” has become an increasing need within the biomedical community. Techniques like Reverse Phase Protein Analysis (RPPA) can provide rapid, high-throughput characterization of a cell’s proteomic state, yielding insight into the altered cellular signaling profiles that accompany diseases such as cancer. Unfortunately, high-dimensional data of this type is often both noisy and complex, making its interpretation challenging for both researchers and clinicians. To address this challenge, we describe the functionality and advantages of BioWheel, a new visualization platform designed to improve discovery of data trends and facilitate the interpretation of time resolved molecular expression data.

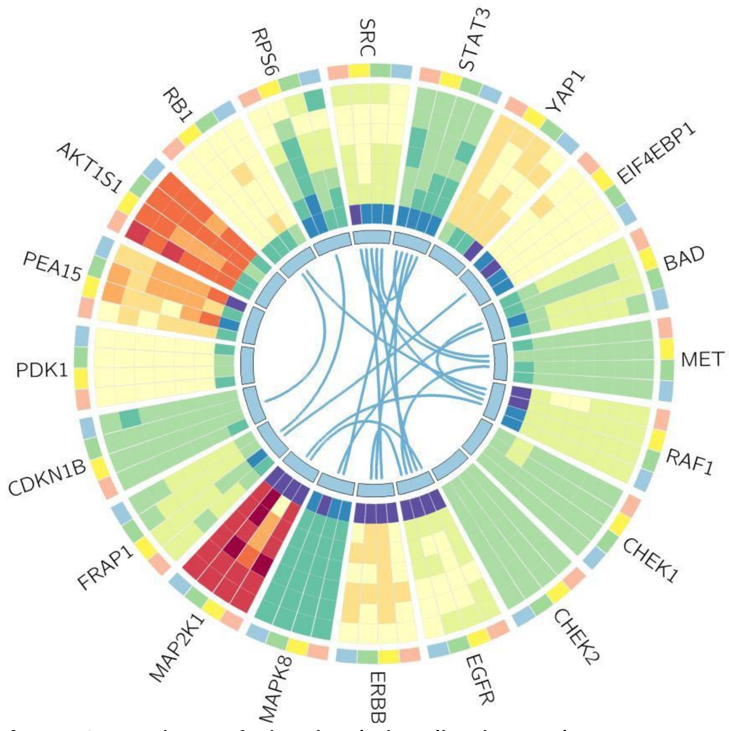


Figure 3. Mock-up of BioWheel visualization tool.

BioWheel displays high throughput timecourse data in an intuitive manner, and it has general applicability to numerous fields that generate high-dimensional data. Here we describe BioWheel’s application and implementation to display the DREAM 8 breast cancer RPPA proteomics data. Proteomics data associated with a single stimulus is plotted to depict protein expression levels by color, essentially as a heat map, but oriented in the shape of a ring, or wheel. Time is plotted along the radial axis: time increases from the center towards the rim (Figure 1). For the DREAM 8 data, time ranges 0 min at the center to 4 hrs / 240 min, represented by the outermost row of the heatmap. Protein names are displayed along the circumference of the wheel. Different experimental or drug perturbations, *e.g.*, inhibitors, are designated by blocks beneath each protein name and specified by unique colors. Each protein is divided into the number of selected inhibitors, so in the DREAM 8 dataset, if all four inhibitors are displayed each protein will have 4 divisions. These blocks and the associated expression data form a set for each protein, which are separated by thin radial oriented lines, forming the spokes of BioWheel. At the inner circumference of the wheel, uniformly colored blocks (light blue in Figure 1) guide the viewer’s eye so that the different expression data sets appear as a single group for each protein. These blocks outline the “hub cap” of the wheel, where an intricate pattern at the center is created by plotting the relationships (*e.g.*, network connectivity) between each protein.

BioWheel is interactive. By clicking the Stimulus button, the user will be able to switch between different stimuli, effectively re-plotting the data. When the Inhibitor option is selected, the user can select any combination of inhibitors to view, displayed by the outermost layer of colors. Color schemes and scale bars are also customizable. The current selection of heatmap colors in Figure 1 display a low protein expression level as blue, increasing through green, yellow, orange, until reaching red as the highest expression level. To gain even more specific information, the user can click a certain time point, and a histogram displaying the expression level data at that moment will come in to view (Figure 2). The innermost circle of BioWheel visualizes the proteins as small blocks, connected by arcs that represent know interactions and pathways, queried from the MiMi public database. By clicking on a specific protein, the user will be shown the time course data for that particular protein under the specified inhibitors (Figure 3). Inhibitors can be further toggled in the graphical window. When an arc connecting two proteins is clicked, a textbox will appear, providing related information from the MiMi database about the know protein interactions (Figure 4).

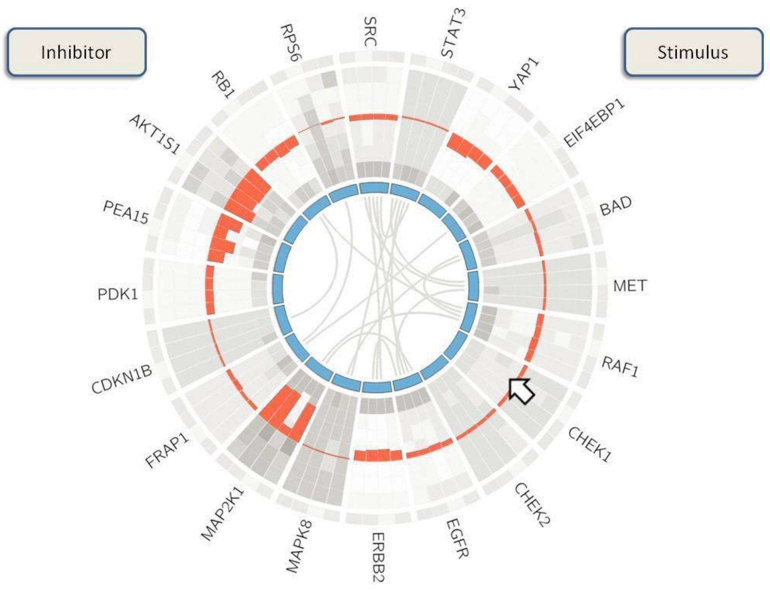


Figure 2. When a user clicks on one time point, BioWheel displays histograms that show expression levels of the 20 proteins for the selected time, for 4 inhibitors.

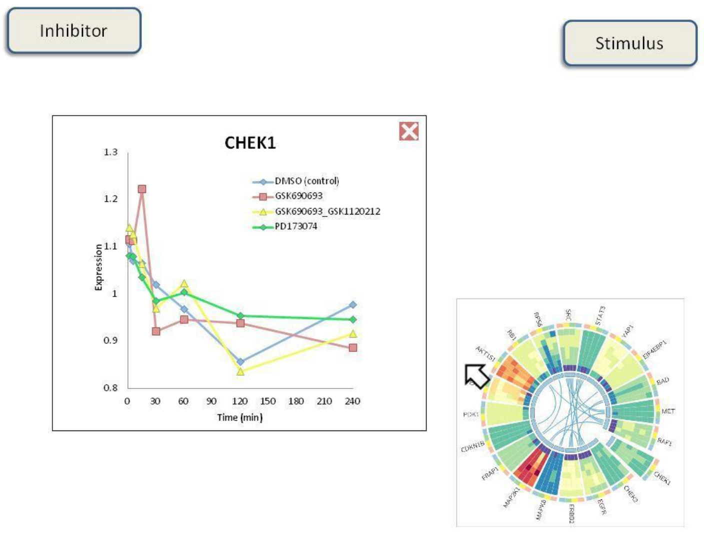
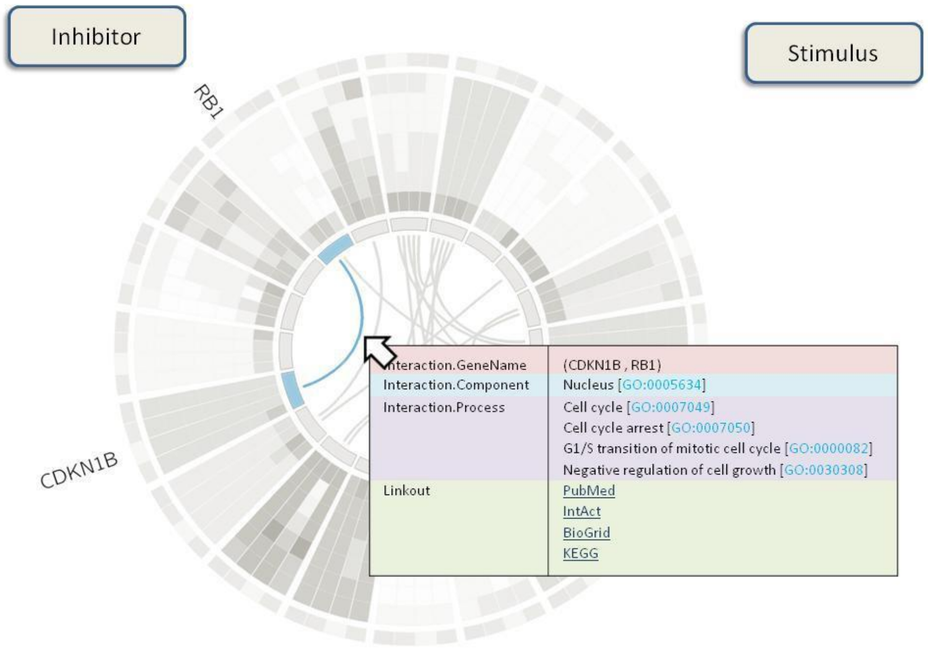


Figure 3. When a user clicks on a particular protein (in this example, CHEK1), time course data is displayed for that protein as a function of the selected inhibitors.

In sum, BioWheel combines static information from multiple traditional heatmaps, temporal dynamics, network relationships and response to perturbations into one graph. For the RPPA breast cancer data, this quickly elucidates some important trends in the data. As an example, MAP2K1 expression clearly decreases in time under the influence of the inhibitor PD172974. Furthermore, the dynamics of MAP2K1 under various different inhibitors has notable similarities to the dynamics of AKT1S1 but no observable correlation with CHEK2 (Figure 1; and see Video: http://youtu.be/i9f6AYaIoOE). Scientists and clinicians will be able to use this form of interactive data visualization as a discovery tool - to rapidly glean new relationships from the data. During the DREAM 8 Challenge, our team used this information to inform hypotheses regarding signaling network architecture, and help choose the types of modeling methods we employed.

With the advent of rapid-throughput biomedical technologies, biomedical data like the sort presented as part of the DREAM 8 Challenge has become increasingly common. Recent studies have shown the benefit of incorporating prior biological knowledge into the interpretation of data and the design of algorithms applied to biomedical data (Küffner et al., Nature Biotech, 2014; Noren et al., PLoS Comp Bio, 2015, in submission). Furthermore, humans are unparalleled in their ability to rapidly recognize complex objects and patterns visually (DiCarlo et al., Cell, 2012; Singwa, Nature Neuro, 2002). BioWheel capitalizes on this natural ability and allows the experts – biomedical experimentalists, clinicians and/or systems biologists – to make decisions on raw data prior to modeling or processing. Interactive visualization tools like BioWheel are poised to become integral methods for discovery, data sharing and model development. Adaptable versions of BioWheel, where any data can be uploaded online through a drag-and-drop option, can be licensed through the EASEL software platform designed by DiBS (www.dibsbiotech.com).



nteraction.GeneName	(CDKN1B , RB1)
Interaction.Component	Nucleus [GO:0005634]
Interaction.Process	Cell cycle [GO:0007049] Cell cycle arrest [GO:0007050] G1/S transition of mitotic cell cycle [GO:0000082] Negative regulation of cell growth [GO:0030308]
Linkout	PubMed IntAct BioGrid KEGG

Figure 4. Clicking on an arc connecting two proteins reveals known information about the relationship between the molecules from public databases (e.g., MIMi).

Supplementary Note 4: Correlation between scores for the network inference sub-challenge experimental data and *in silico* data tasks

For the network inference sub-challenge, team scores for the experimental data task (mean AUROC) were compared with scores for the *in silico* data task (AUROC; **Fig. 3b** in main text). Pearson correlations were calculated and reported in main text for all teams that participated in both tasks (*r* = 0.35, 52 teams; some teams with correlated submissions were filtered out to avoid bias, see **Supplementary Note 10**) and for those teams that did not use prior information in the experimental data task (*r* = 0.68, 18 teams). These calculations included teams that used a different method for each task (**Supplementary Table 2**), which could lead to inflated correlations (here, “different method” means a difference that is not solely due to use of prior information). To investigate this we recalculated the correlations, excluding teams that used different methods in each task, and found good agreement with those reported above (*r* = 0.40 for all 46 teams that used the same method in each task and *r* = 0.67 for the 15 teams that used the same method in each task, but did not use prior information in the experimental data task).

Supplementary Note 5: Classification of methods

41 of the 80 teams that participated in the network inference sub-challenge (SC1) provided information about their methods (**Supplementary Note 7**). This included a self-reported method classification. These classifications were then reviewed by challenge organizers by referring to full method descriptions provided by teams (**Supplementary Note 1**) and, in some cases, by discussing the classification directly with the team concerned.

This process resulted in a classification consisting of eight groups: Bayesian networks, ensemble, linear regression, nonlinear regression (tree-based approach), ordinary differential equations (ODEs), pairwise score, prior network only and other (**Fig. 3e,f** in main text**, Table 1** in main text and **Supplementary Table 2**). The “ensemble” group consists of methods that combine several different approaches. The “nonlinear regression” group consists of tree-based approaches such as random forests and gradient tree boosting. The methods in the “pairwise score” group consider bivariate relationships only (for example, based on correlation). The “other” group contains methods that do not fit into any of the other seven groups. Some teams used different approaches for the experimental data task and *in silico* data task and therefore have a different method classification for each task.

As discussed in main text, caution should be exercised when comparing performances between method classes. Boundaries between method classes are not well defined (for example, a Bayesian network approach with linear Gaussian conditional distributions could also fit into the linear regression category) and factors which are not captured by the method class, such as pre-processing steps and use of prior information, can influence performance.

Supplementary Note 6: Scoring metric for time-course prediction sub-challenge

For the experimental data task, participants were asked to predict context-specific time-courses under 5 test inhibitors not contained in the training data (this included an mTOR inhibitor). The test dataset, obtained under mTOR inhibition, was used to assess participants' mTOR inhibitor predictions. Predictions for the remaining 4 test inhibitors were not used for scoring, but were included as part of this task to ensure that the identity and number of test inhibitors for the network inference sub-challenge remained blinded.

Prediction accuracy was quantified using root mean squared error (RMSE).To avoid bias due to data from different cell-lines and proteins being on different scales, a separate RMSE score was calculated for each *(cell line, phosphoprotein)* pair. Specifically, let $x_{p,c,s,t}$ denote the gold-standard abundance (as given by the test data, on a $\log_2$ scale) of phosphoprotein p for cell line c and stimulus s at time point t. Any replicates in the test data were averaged (on the $\log_2$ scale). Let the corresponding abundance value predicted by team m be denoted by $\hat{x}^m_{p,c,s,t}$. Then, the RMSE score for team m for *(cell line, phosphoprotein)* pair (c,p) is given by

$$\mathrm{RMSE}^m_{p,c} = \sqrt{\frac{1}{TS}\sum_{t=1}^{T}\sum_{s=1}^{S}\left(\hat{x}^m_{p,c,s,t} - x_{p,c,s,t}\right)^2}$$

where $T = 7$ is the number of time points and $S = 8$ is the number of stimuli. Therefore each RMSE score resulted from a comparison of 56 data. In the main text, data for each *(cell line, phosphoprotein)* pair is referred to as a "data block".

Statistical significance of $\mathrm{RMSE}^m_{p,c}$ scores were assessed using simulated null distributions. A null distribution of 100,000 RMSE scores was obtained for each *(cell line, phosphoprotein)* pair by scoring time-courses generated by sampling data points at random from the training data. In particular, for a given *(cell line, phosphoprotein)* pair, 8 time-courses (one for each stimulus) each with 7 time points were formed by sampling from a subset of the training data. This subset consisted of all time points, stimuli and inhibitor regimes for the corresponding *(cell line, phosphoprotein)* pair (replicates were averaged prior to sampling). These 8 time-courses were then scored and the procedure repeated 100,000 times to obtain the null RMSE distribution. Gaussian fits to the null distributions were used to calculate *p*-values. For each *(cell line, phosphoprotein)* pair, the set of *p*-values (across all teams) underwent multiple testing correction using the Benjamini-Hochberg FDR procedure. There were 16 *(cell line, phosphoprotein)* pairs for which no team achieved a statistically significant score (**Supplementary Table 5**; FDR<0.05) and these pairs were disregarded in the scoring procedure. Teams were ranked according to RMSE within each *(cell line, phosphoprotein)* pair and a final ranking was obtained by taking the mean rank across all pairs.

Scoring proceeded in a similar fashion for the *in silico* data task. Participants were asked to make predictions for each node under 20 test inhibitors (each node inhibited in turn; predictions were not required for the inhibited node) and test data was available to score predictions for 16 of these test inhibitors. RMSE scores were calculated for each *(test inhibitor, predicted node)* pair. Let $x_{p,i,s,t}$ denote the gold-standard abundance (as given by the test data, on a $\log_2$ scale) of node p under inhibition of node i for stimulus s at time point t. Let the corresponding abundance value predicted by team m be denoted by $\hat{x}^m_{p,i,s,t}$. Then, the RMSE score for team m for *(test inhibitor, predicted node)* pair (i, p) is given by

$$\mathrm{RMSE}^m_{p,i} = \sqrt{\frac{1}{TS}\sum_{t=1}^{T}\sum_{s=1}^{S}\left(\hat{x}^m_{p,i,s,t} - x_{p,i,s,t}\right)^2}$$

where $T = 10$ is the number of time points (the 45 minute time-point in the *in silico* test data was not used in scoring) and $S = 8$ is the number of stimuli. Therefore each RMSE score resulted from a comparison of 80 data points. In the main text, data for each *(test inhibitor, predicted node)* pair is referred to as a "data block". Statistical significance was assessed analogously to the experimental data task with null RMSE distributions generated for each *(test inhibitor, predicted node) pair* and time-courses sampled from data points in the training data for the corresponding predicted node (after averaging of replicates). This revealed two *(test inhibitor, predicted node)* pairs for which no team achieved a statistically significant score (**Supplementary Table 6**) and these pairs were excluded from scoring (dummy nodes were also excluded). Teams were ranked according to RMSE within each *(test inhibitor, predicted node)* pair and a final ranking was obtained by taking the mean rank across all pairs.

Supplementary Note 7: Criteria to be a member of the HPN-DREAM Consortium and to be eligible for a prize

To be included as a member of the HPN-DREAM Consortium and listed as contributors of this manuscript, participants were required to complete an online survey to provide high-level details of their method (network inference and time-course prediction sub-challenges only) and submit a detailed write-up describing their method. These write-ups were edited and compiled by challenge organizers to form **Supplementary Notes 1-3.** Where applicable, all teams, including those not members of the HPN-DREAM Consortium, were included in analyses (**Supplementary Table 2**). For the network inference sub-challenge, 33 out of 80 teams are included as Consortium members. An additional 8 teams provided a limited amount of information about their methods in an initial brief write-up, but did not provide a full write-up. For these teams, summary sentences were included in **Supplementary Table 2**, but they are not members of the Consortium. For the time-course prediction sub-challenge, 9 out of 14 teams are included as Consortium members and for the visualization sub-challenge all 14 teams are included.

For the network inference and time-course prediction sub-challenges, to be eligible for a challenge prize, top-performing teams had to meet two criteria in addition to the Consortium member inclusion criteria described above. First, teams had to submit code for their method which, upon execution by challenge organizers, reproduced their final submission. Code from a number of teams, including top-performing teams, has been made available through Synapse at https://www.synapse.org/HPN_DREAM_Network_Challenge, under the section "HPN-DREAM Community Resource". Second, teams had to make submissions to both the experimental data and *in silico* data tasks of the corresponding sub-challenge. The top-ranked team for the time-course prediction sub-challenge experimental data task (Team44) did not participate in the *in silico* data task and so the teams ranked second (Team42) and third (Team10) were declared as best-performers (**Supplementary Table 4**; the team ranked second was not robustly ranked above the team ranked third resulting in two best-performers).

Supplementary Note 8: Model for *in silico* data generation

ErbB Signaling Model

The model for this sub-challenge is an extended version of the ErbB signalling model by Chen et al. [1]. It is a mass action model of ErbB1-4 receptors (EGFR, HER2/Neu2, ErbB3 and ErbB4), and the MAPK and PI3K/Akt signalling cascades. The model can be stimulated by the two ligands Epidermal Growth Factor (EGF) and Heregulin (HRG). We extended the model by the three inhibitors Cetuximab (ErbB1), Pertuzumab (ErbB2) and U0126 (MEK).

Implementation of inhibitors

The inhibition of the phosphorylated form of a protein reduces its downstream kinase activity. The mode of action for the three inhibitors is the same: they sequester away the unphosphorylated form of the target kinase and therefore limit the amount of phosphorylated forms.

Cetuximab (ErbB1)

Cetuximab sequesters ErbB1 and prevents its activation. Implementation (see Table A.3) and kinetic rates (see Table A.2) of this inhibitor are based on [2].

Pertuzumab (ErbB2)

Pertuzumab blocks ErbB2 dimerization. Implementation (see Table A.4) and kinetic rates (see Table A.2) of this inhibitor are based on [2].

U0126 (MEK)

U0126 inhibits the phosphorylation of inactive MEK. For Implemented reactions see Table A.5, for the kinetic rates see Table A.2.

SC2B Test Data Inhibitors

Inhibitions in test data are implemented in a similar way to those in the training data, but in contrast to the training data we assume a complete or perfect effect such that the inhibited phosphoprotein is completely removed.

Implementation of antibodies

All antibodies are implemented to bind specific phosphorylation states of a protein. For example, ABx may contain all single phosphorylated forms of a protein, while ABy contains all doubly phosphorylated forms.

Readouts and Reference Causal Graph

We parallel the definition of a causal graph introduced to the statistics in the context of structural equation models [3]. Every node of the graph corresponds to one molecular species and furthermore to one state in our ordinary differential equation (ODE). In the case where the states track the concentration of all involved species, we will generally have (linear) algebraic relations (mass-conservations) between states, implying that the ODE solution is constrained to some lower-dimensional linear subspace. Hence, the considered ODE system can be made equivalent to a differential-algebraic set of equations, where only a subset of species are retained in the differential part of this equation system. In the following we will use the full ODE description without those explicit algebraic constraints. In general, there is a directed edge from node i to node j if the right-hand-side (rhs) of the differential equation for state j is a function of state i, i.e.

$$\frac{d}{dx} x_j(t) = f_j(x_i(t),\cdot),$$

where "$\cdot$" should indicate a possible dependency of f_j on other state variables. In the case of ODEs resulting from elementary mass-action reactions, all ODE right-hand-sides – and hence the causal graph – can be determined directly from the stoichiometric matrix S of the reaction system. We call a reaction elementary if all species on left-hand-side (lhs) are consumed by it. For example the homo-dimerization is $2A \rightleftharpoons A{:}A$ elementary, while $2A \rightleftharpoons A + C$ is not elementary[1]. The element S_{ij} denotes the net change of species i through reaction j. We illustate this by considering the following reaction motif

$$\begin{aligned} R &\rightleftharpoons R_p \\ R_p + M &\rightleftharpoons R_p{:}M \rightharpoonup R_p + M_p \\ R_p + M_p &\rightleftharpoons R_p{:}M_p \rightharpoonup R_p + M_{pp} \\ P + M_p &\rightleftharpoons P{:}M_p \rightharpoonup P + M \\ P + M_{pp} &\rightleftharpoons P{:}M_{pp} \rightharpoonup P + M_p, \end{aligned} \qquad (1.1)$$

which constitutes a single stage of a mitogen-activated-protein-kinase (MAPK) cascade. More specifically, it represents the double-phosphorylation of protein MEK (M_{pp}) by a phosphorylated Raf protein (R_p). Assume the forward reaction in the first line is the j-th reaction and R is the i-th species then $S_{ij} = -1$, while the only other non-zero elements of the j-th column is $S_{kj} = 1$, with R_p the k-th species. In general S is the incidence matrix (node-edge matrix) of a directed hypergraph. Hyperedges are introduced by reactions of higher arity, such as the bi-molecular reaction in line two of (1.1). The corresponding hypergraph for line two is shown in **Fig. 1a**. In order to determine the rhs dependency of the ODE for species i we also need to determine the velocity of reactions in which species i is participating. For a mass-action reaction the velocity is a monomial containing those states of species that are part of the lhs of the reaction. Moreover, for elementary reactions the species on the lhs of a reaction can be read off from the stoichiometric matrix. For instance, the forward reaction of the reversible reaction in line two of (1.1) consumes R_p and M and hence its velocity is $R_p \cdot M$, assuming unit reaction constants throughout[2]. In order to construct a causal graph the particular functional form of the rhs dependency is irrelevant and we can simplify the hypergraph to a normal graph that just retains

[1] For non-elementary mass-action reactions the substrate and the product stoichiometric coefficients are required to determine the effective dependencies.
[2] We adopt the slight notational abuse of denoting the concentration or state variable corresponding to a molecular species directly by the name of that species.

the dependency information. For the discussed example the corresponding normal graph is given in **Fig. 1 b**. Note that this graph is equivalent to the graph that can be extracted from the zero-patterns of the Jacobian matrix corresponding to the system of ODEs.

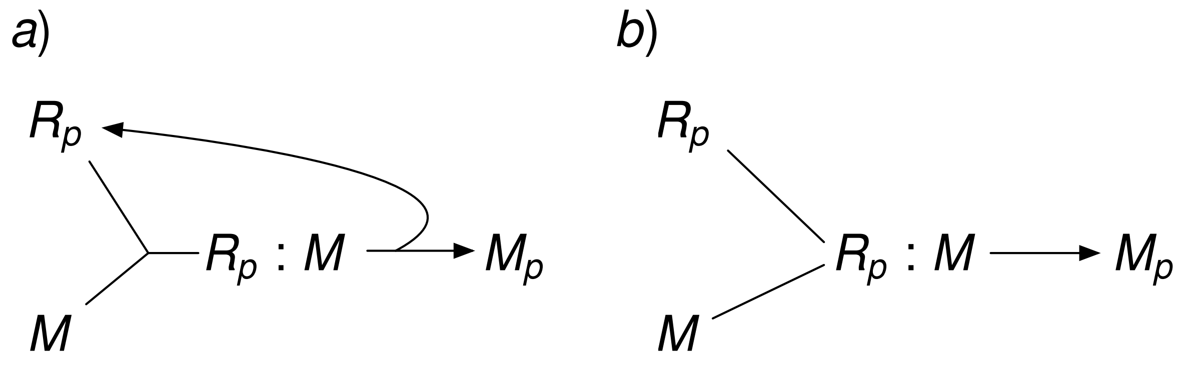


Figure 1. (a) Hypergraph representation of the reactions in line two of (1.1). (b) Corresponding normal graph extracted from the hypergraph; edges without arrow indicate bidirectional edges.

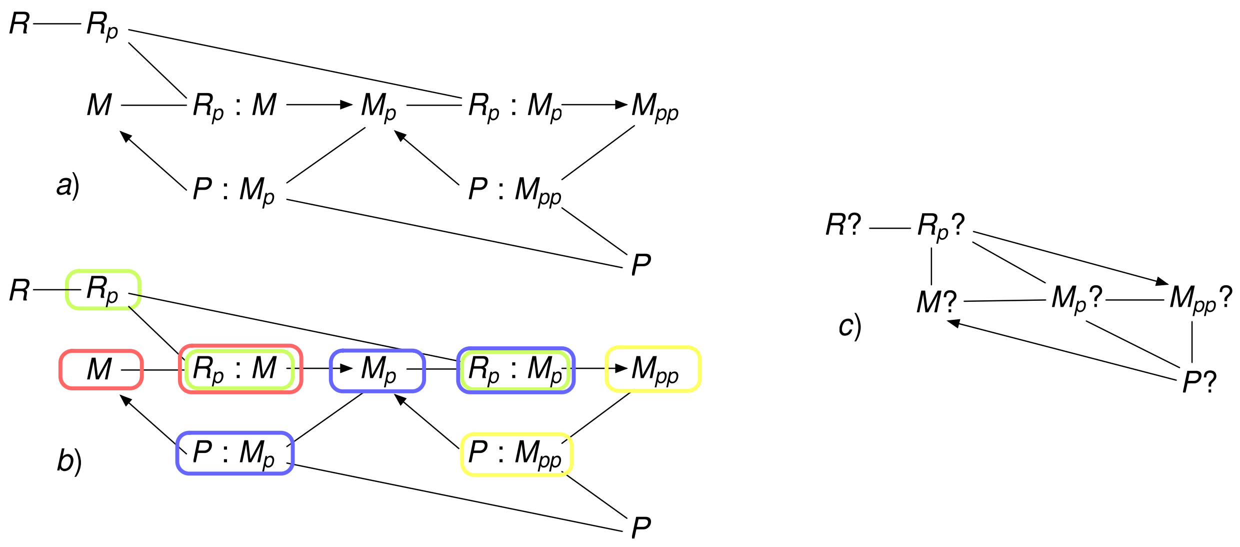


Figure 2. (a) Causal graph of reaction system (1.1) on the single-species level. (b) Annotation to denote species aggregates (aggregates for P and R are not shown). (c) Resulting dependency graph between aggregates, where aggregates are denoted by "?"; edges without arrow indicate bidirectional edges.

Most molecular techniques do not permit the measurement of a single molecular species but are rather able to measure a species in all complexations. For instance, the free form M alone cannot be measured but $M + R_p{:}M$ can be measured. We call such measurable linear combinations of species aggregates and we finally need to define a causal graph between aggregates in order to be consistent with experimental measurements. Aggregates do not correspond to partitions of the graph nodes but rather to coverings, i.e. the aggregates overlap. A direct approach to define the dependency structure among aggregates is to inherit the dependency structure from its constituents. That is, there is a directed edge from aggregate i to j if there is a directed edge from any species in i to any species in j. We will see later that this approach introduces spurious edges because it does not account for mass conservation relations. The steps in the proposed graph aggregation are illustrated in **Fig. 2**, where aggregates are defined in terms of all phosphorylated forms of a protein and all its non-phosphorylated forms. We note that in the species-level causal graph of **Fig. 2a** there exists a directed path from M_p to R_p apparently justifying the bidirectional edge between aggregate $M_p?$ and $R_p?$ in **Fig. 2c**. However, by inspecting the system of ODEs we see that the aggregate $R_p?$ shows no functional dependency on any members of $M_p?$. To see this we write down the ODE of the involved species with unit rates,

$$\begin{gathered}\frac{d}{dt}R_p = R - R_p - R_p \cdot M + 2R_p{:}M - R_p \cdot M_p + 2R_p{:}M_p \\ \frac{d}{dt}R_p{:}M = R_p \cdot M - 2R_p{:}M \\ \frac{d}{dt}R_p{:}M_p = R_p \cdot M_p - 2R_p{:}M_p\end{gathered} \qquad (1.2)$$

yielding

$$\frac{d}{dt}\left(R_p + R_p{:}M + R_p{:}M_p\right) = R - R_p$$

and hence show that the edge from $M_p?$ to $R_p?$ is spurious. We remark that such significant cancelation is due to the particular choice of aggregates. In order to remove such spurious edges we need to first determine the net effect of each reaction on the aggregate instead of the species. Hence we can define another stoichiometric matrix where the number of rows is the number of aggregates. If a reaction causes the increase of one and the decrease of another species that are both within the same aggregate, then the net effect of that reaction with respect to the concentration of the aggregate can be zero. More formally a reaction j is inactive with respect to the aggregate A_k consisting of species $i \in A_k$ if

$$\sum_{i \epsilon A_k} N_{ij} = 0. \qquad (1.3)$$

For those reaction for which (1.3) does not hold, one needs to determine the functional dependency of aggregate's A_k concentration $y_k(t)$ with respect to reaction j

$$\frac{d}{dt}y_k(t) = \frac{d}{dt}\sum_{i \epsilon A_k} x_i(t) = \sum_{i \epsilon A_k} N_{ij} \prod_{p=1}^{n} x_p(t)^{S_{pj}} + R(t) \qquad (1.4)$$

where $R(t)$ denotes the contributions of all reactions other than the j-th one and where the matrix S_{ij} is obtained from N_{ij} by retaining only its negative elements and setting positive entries to zero[3]. The remaining rhs functional dependency of aggregate A_k in terms of the original state variables $x_i(t)$ can be read off from (1.4). Defining the aggregate stoichiometric matrix $\widetilde{N}_{kj} = \sum_{i \epsilon A_k} N_{ij}$, the expression

$$\frac{d}{dt}y_k(t) = \sum_{j=1}^{m} \widetilde{N}_{kj} \prod_{p=1}^{n} x_p(t)^{S_{pj}}$$

makes apparent that the rhs of this ODE can in general not be written only in terms of aggregate variables y_i, i.e. cannot be of the form

$$\frac{d}{dt}y_k(t) = \sum_{j=1}^{m} \widetilde{N}_{kj}\, \tilde{v}_j\left(y_1(t), \dots, y_q(t)\right),$$

[3] Hence, we managed to reconstruct the stoichiometric substrate coefficients, by virtue of assuming elementary mass-action reactions.

with $q \leq n$ the number of aggregates. Hence, the evolution of the aggregate concentration cannot be described in terms of a self-consistent (i.e. autonomous) q-dimensional ODE. In turn, however, it means that a causal graph over aggregates that explains all dynamics is impossible. Even though the aggregate causal graph is an incomplete account of the system, it needs to preserve causality relations. More specifically, it should not indicate causality, where there cannot be any. For instance, consider the ODE for the aggregate $M_p?$ of reaction system (1.1)

$$\frac{d}{dt}M_p? = \frac{d}{dt}\left(M_p + M_p{:}R_p + M_p{:}P\right) = R_p{:}M + M_{pp}{:}P - M_p{:}R_p - M_p{:}P.$$

The rhs cannot be written as a function only in terms of aggregates. Every summand of the rhs stems from one particular reaction but is sometimes associated with two aggregates. For instance, $R_p{:}M$ is element of $R_p?$ and $M?$ In the construction of the aggregate causal graph we follow the negative results and require that there must not be an edge from aggregate A_k to A_j if and only if none of the elements of the rhs of state $y_k(t)$ is contained in aggregate A_j. In turn, it means for our example that we need to draw an edge from $R_p?$ to $M_p?$ and from $M?$ to $M_p?$. The resulting causal graph is depicted in **Fig. 2c**. For the considered ErbB signaling model the resulting causal graph between the phosphoforms, including all ErbB receptors is shown in **Fig. 3**; extracted by automatic means through an algorithm based on the above considerations.

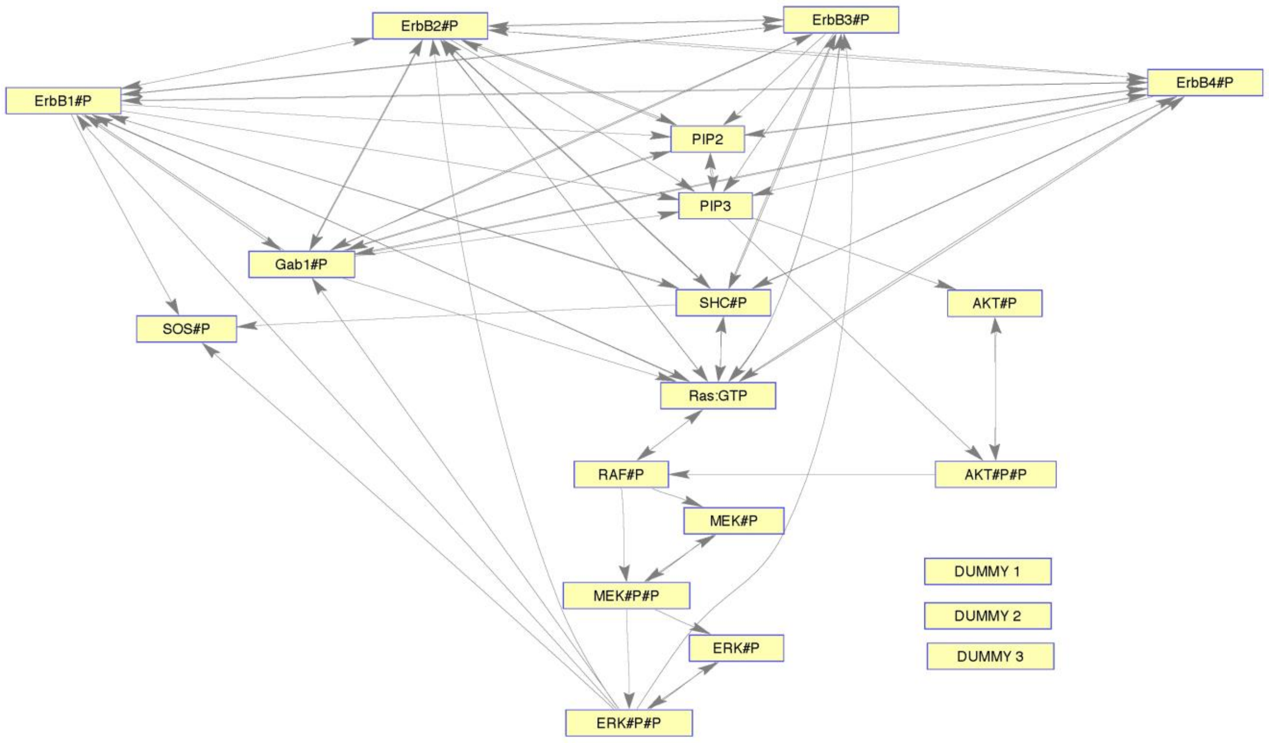


Figure 3. Aggregate causal graph of the ErbB signaling model.
Reproduced from **Supplementary Figure 4.**

Measurement Error Model

The readout of a RPPA assay is the intensity signal per spot caused by the fluorescently labeled antibody. The finite dynamic range of the scanner is modeled as a logistic function resulting on the lower end into a detection threshold, and on the upper end into saturation. Moreover, we include a small additive Gaussian noise and multiplicative noise, giving rise to a lognormal component. The latter can be justified by the fact that error incurred in intensity quantification of the spots is proportional to the spot intensity. Denoting by $y_i^{(k)}(t_j)$ the concentration of aggregate i at time t_j in the k-th perturbation/stimulation condition we choose

$$z_{ijk} = \alpha + \beta \frac{\exp\left[\gamma\left(y_i^{(k)}(t_j) - \delta\right)\right]}{1+\exp\left[\gamma\left(y_i^{(k)}(t_j) - \delta\right)\right]} \eta_{ijk} + \epsilon_{ijk}, \qquad (1.5)$$

where $\epsilon_{ijk} \sim \mathcal{N}(0, \sigma^2)$ and $\eta_{ijk} \sim e^{\mathcal{N}(0,\omega^2)}$. With $\omega = 0$, (1.5) reduces to the model used in [4]. Moreover, note that we assume the absence of a batch effect, i.e. all sigmoidal parameters as well as the noise parameters are not a function of k. The dynamic range for $y_i^{(k)}(t_j)$ can be very large for ODE models. First, the protein concentration are in very different orders of magnitude, e.g. ErbB4 receptor in several hundreds and SOS proteins of order 10^7. Second, dynamic states may transverse many orders of magnitudes. Practically, using (1.5) this results in states that are well in saturation or are within the noise floor. In order to circumvent this problem, we assume that every antibody is optimized for the expected concentration of its protein. That is, low copy number proteins come with antibodies with large affinities. One way to achieve that numerically is to normalize the signal $y_i^{(k)}(t_j)$ by the mean concentration over all conditions and times for aggregate i. Such a normalized variable can then be defined as

$$\xi_{ijk} = \sigma_i + \frac{y_i^{(k)}(t_j)}{\kappa_i} \qquad (1.6)$$

with the normalization

$$\kappa_i = \frac{1}{NM} \sum_{k=1}^{N} \sum_{j=1}^{M} y_i^{(k)}(t_j), \qquad (1.7)$$

and addition of an aggregate specific offset σ_i. The purpose of that offset is to bound the fold change. With standard fold change of $\sigma_i = 0$ we still encounter the problem of unrealistic dynamic ranges mostly due to low initial values. For instance, consider a species aggregate with average concentration of $\kappa_i = 10^3$ and with initial values of 10^{-16} (i.e. numerical precision) at a certain condition, then clearly the fold change has a dynamic range of at least 19 orders of magnitude (given that there must be at least one condition for which the concentration is higher than 10^3). Such a dynamic range is utterly unrealistic for a technical device (e.g. in the RPPA data, we see dynamic ranges of roughly three orders of magnitude). We decide to choose the offset in that normalized domain randomly as $\sigma_i \sim 2^{-\Delta}$ with $\Delta \sim \mathcal{U}[8,0]$. Hence, the offset added to the normalized time-course is at most the mean (2^0) of that time-course. Evidently, an aggregate-unspecific offset of just $\sigma_i = \sigma$ could also be chosen, however, this gives rise to a significant overrepresentation of values around σ in the data. In order to generate measurements we thus replace $y^{(k)}(t_j)$ in (1.5) with ξ_{ijk}. In **Fig. 4** we show the normalized concentration measure (1.6) and how it is transformed under the measurement characteristics (1.5). The histograms show the distribution of (1.6) for the whole dataset. The above randomization to distribute very small fold changes is clearly visible. Moreover, the three included dummy measurements that are constant throughout all conditions and time points (set to $2^{-2}, 2^{0}$, and 2^{2}) unfortunately leave their fingerprint. A further randomization to get more realistic distribution shapes would require the inclusion of a batch effect, e.g. all dummy states could have different levels for each batch. To assess the technical variability we compute the correlation coefficient ρ between replicates. A scatter plot over the whole dataset for two replicates is shown in **Fig. 5**.

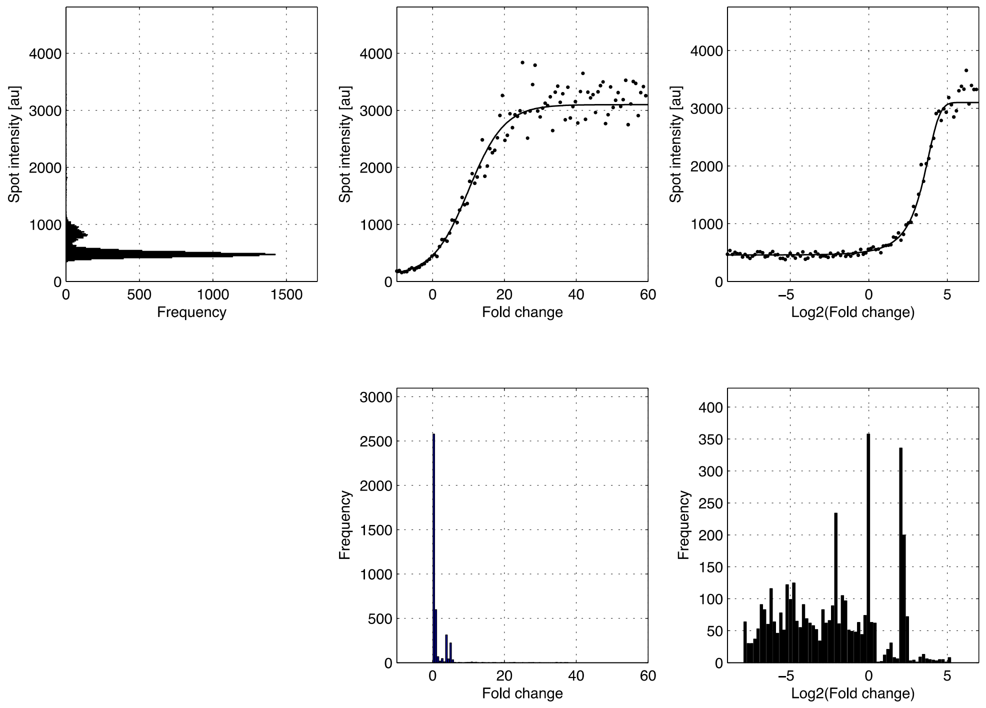


Figure 4. Characterization of the measurement (1.5) in terms of input-output distribution shown for logarithmic and linear fold changes (1.6) for the complete dataset; random samples from (1.5) are overlaid to the noise-free sigmoidal characteristics.

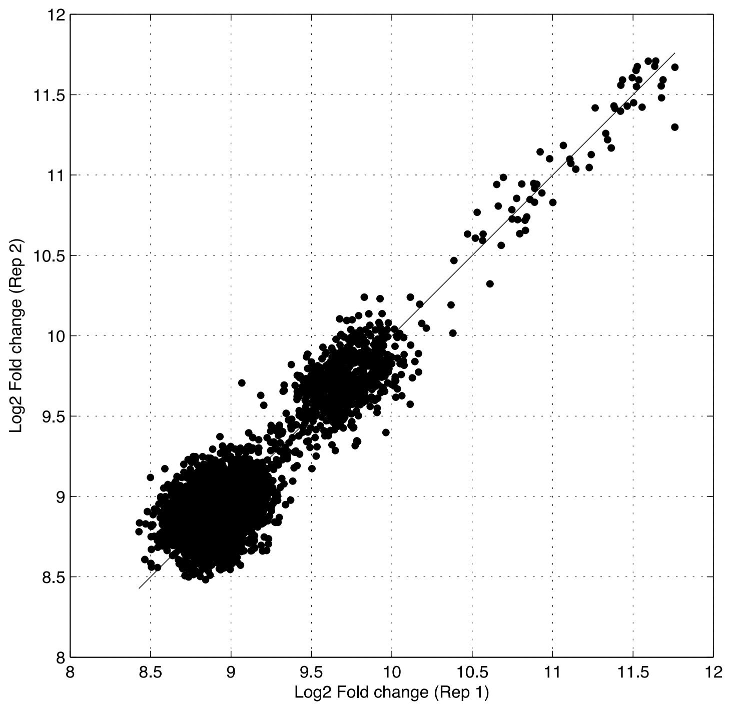


Figure 5. Scatter plots of all values in the *in silico* dataset for two replicates. Correlation coefficient $\rho = 0.96$.

Time-course prediction sub-challenge - test data normalization issues
After publication of the final leaderboards for the time-course prediction sub-challenge (SC2B), two minor issues concerning the intensity levels in the gold-standard test data were discovered.

The first issue was a missing offset term in the test data. Aggregate-specific offset terms σ_i were included in the additive noise term ξ_{ijk} for the training data (see Eq. 1.6), but were not included when generating the test data.

The second issue was the use of an incorrect normalization factor κ_i (see Eq. 1.7) when generating the test data. This normalization factor should remain constant for a complete set of training and test data. However, while generating the original test data it was recomputed using only a subset of the 20 training data conditions (no inhibitor conditions were used). As a result, different normalization factors were used to generate the training data and test data.

To resolve these issues we regenerated the test data including the offset term and using the correct normalization factor. Submissions to SC2B were rescored using the corrected test data and the final leaderboards were updated. While final rankings of some teams were changed by this update, the top-performing team (Team34) remained the same and so the overall result of SC2B was unaffected. The updated scores can be found in **Supplementary Table 4** and on the final leaderboards on Synapse (https://www.synapse.org/HPN_DREAM_Network_Challenge).The initial version of the final leaderboard is also available to download on Synapse (https://www.synapse.org/#!Synapse:syn4922077).

References

1. William W Chen, Birgit Schoeberl, Paul J Jasper, Mario Niepel, Ulrik B Nielsen, Douglas Lauffenburger, and Peter K Sorger. Input-output behavior of ErbB signaling pathways as revealed by a mass action model trained against dynamic data. *Molecular systems biology*, 5(239):239, January 2009.

2. Birgit Schoeberl, Emily Pace, Jonathan B Fitzgerald, Brian D Harms, Lihui Xu, Lin Nie, Bryan Linggi, Ashish Kalra, Violette Paragas, Raghida Bukhalid, Viara Grantcharova, Neeraj Kohli, Kip a West, Magdalena Leszczyniecka, Michael J Feld- haus, Arthur J Kudla, and Ulrik B Nielsen. Therapeutically targeting ErbB3: a key node in ligand-induced activation of the ErbB receptor-PI3K axis. *Science signaling*, 2(77):ra31, January 2009.

3. Judea Pearl. Causal inference in statistics: An overview. *Statistics Surveys*, (3):96– 146, January 2009.

4. S. Troncale, A. Barbet, L. Coulibaly, E. Henry, B. He, E. Barillot, T. Dubois, P. Hupé, and L. de Koning, NormaCurve: a SuperCurve-based method that simultaneously quantifies and normalizes reverse phase protein array data. *PLoS One*, vol. 7, no. 6, Jan. 2012.

Appendix

Species

Table A.1.: List of species (excerpt)

Name	Note
Cetuximab	Cetuximab
Pertuzumab	Pertuzumab
U0126	U0126
c2	ErbB1:ATP
c2:Cetuximab	*ErbB1:ATP:Cetuximab*
c2:Cetuximab:c2	*ErbB1:ATP:Cetuximab:ErbB1:ATP*
c3	EGF:ErbB1:ATP
c3:Cetuximab	*EGF:ErbB1:ATP:Cetuximab*
c3:Cetuximab: c3	*EGF:ErbB1:ATP:Cetuximab: EGF:ErbB1:ATP*
c6	ErbB1:ATP
c6:Cetuximab	*ErbB1:ATP:Cetuximab*
c6:Cetuximab: c6	*ErbB1:ATP:Cetuximab:ErbB1:ATP*
c10	EGF:ErbB1:ATP
c10:Cetuximab	*EGF:ErbB1:ATP:Cetuximab*

c10:Cetuximab: c10	*EGF:ErbB1:ATP:Cetuximab: EGF:ErbB1:ATP*
c45	Raf#P
c47	MEK
c47:U0126	*MEK:U0126*
c72	(Raf#P)_i
c74	(MEK:Raf#p)_i
c72:U0126	*(MEK:Raf#P)_i:U0126*
c141	ErbB2
c141:Pertuzumab	*ErbB2:Pertuzumab*
c141:Pertuzumab:c141	*ErbB2:Pertuzumab:ErbB2*
c155	ErbB2
c155:Pertuzumab	*ErbB2:Pertuzumab*
c155:Pertuzumab:c155	*ErbB2:Pertuzumab:ErbB2*
c524	ErbB1_h:ATP
c524:Cetuximab	*ErbB1_h:ATP:*Cetuximab
c524:Cetuximab:c524	*ErbB1_h:ATP:*Cetuximab:*ErbB1_h:ATP*
c529	EGF:ErbB1_h:ATP
c529:Cetuximab	*EGF:ErbB1_h:ATP:*Cetuximab
c529:Cetuximab:c529	*EGF:ErbB1_h:ATP:*Cetuximab: *EGF:ErbB1_h:ATP*
c530	ErbB1_h:ATP
c530:Cetuximab	*ErbB1_h:ATP:*Cetuximab
c530:Cetuximab:c530	*ErbB1_h:ATP:*Cetuximab:*ErbB1_h:ATP*
c531	ErbB1
c531:Cetuximab	*ErbB1:*Cetuximab
c531:Cetuximab:c531	*ErbB1:*Cetuximab:*ErbB1*
c532	ErbB1_h
c532:Cetuximab	*ErbB1_h:*Cetuximab
c532:Cetuximab:c532	*ErbB1_h:*Cetuximab:*ErbB1_h*

Parameters

Table A.2.: List of parameters (excerpt)

Name	Value
kfCetuximab_1	stepfunc(time,TpInhibitor-0.01,0,TpInhibitor,1.1E-10)
kfCetuximab_1	1.1E-3
kfCetuximab_2	7.64E-5
krCetuximab_2	2.2E-3
kfPertuzumab_1	stepfunc(time,TpInhibitor-0.01,0,TpInhibitor,5.60-11)
kfPertuzumab _1	9.5E-4
kfPertuzumab _2	3.9E-5
kfPertuzumab _2	1.9E-3
kfU0126_1	stepfunc(time,TpInhibitor-0.01,0,TpInhibitor,1.0-10)
kfU0126_1	1.1E-3
k44U0126	stepfunc(time,TpInhibitor-0.01,0,TpInhibitor,1.07-5)
kd52U0126	stepfunc(time,TpInhibitor-0.01,0,TpInhibitor,0.033)

Table A.3.: Implemented Cetuximab reactions.

Name	Reaction Rate
Cetuximab_v1	
Cetuximab + c531 → C531:Cetuximab	kfCetuximab_1*Cetuximab*c531
Cetuximab + c531 ← c531:Cetuximab	kfCetuximab_1*c531:Cetuximab
Cetuximab_v2	
c531:Cetuximab + c531 → C531:Cetuximab:c531	kfCetuximab_2*c531:Cetuximab*c531
c531:Cetuximab + c531 ← C531:Cetuximab:c531	kfCetuximab_2*c531:Cetuximab:c531
Cetuximab_v3	
Cetuximab + c2 → c2:Cetuximab	kfCetuximab_1*Cetuximab* c2
Cetuximab + c2 ← c2:Cetuximab	kfCetuximab_1* c2:Cetuximab
Cetuximab_v4	
c2:Cetuximab + c2 → c2:Cetuximab:c2	kfCetuximab_2* c2:Cetuximab*c2
c2:Cetuximab + c2 ← c2:Cetuximab:c2	kfCetuximab_2* c2:Cetuximab:c2
Cetuximab_v5	
Cetuximab + c3 → c3:Cetuximab	kfCetuximab_1*Cetuximab*c3
Cetuximab + c3 ← c3:Cetuximab	kfCetuximab_1* c3:Cetuximab
Cetuximab_v6	
c3:Cetuximab + c3 → c3:Cetuximab:c3	kfCetuximab_2*c3:Cetuximab*c3

c3:Cetuximab + c3 ← c3:Cetuximab:c3	kfCetuximab_2*c3:Cetuximab:c3
Cetuximab_v7	
Cetuximab + c10 → c10:Cetuximab	kfCetuximab_1*Cetuximab*c10
Cetuximab + c10 ← c10:Cetuximab	kfCetuximab_1*c10:Cetuximab
Cetuximab_v8	
c10:Cetuximab + c10 → c10:Cetuximab:c10	kfCetuximab_2*c10:Cetuximab*c10
c10:Cetuximab + c10 ← c10:Cetuximab:c10	kfCetuximab_2*c10:Cetuximab: c10
Cetuximab_v9	
Cetuximab + c6 → c6:Cetuximab	kfCetuximab_1*Cetuximab*c6
Cetuximab + c6 ← c6:Cetuximab	kfCetuximab_1*c6:Cetuximab
Cetuximab_v10	
c6:Cetuximab + c6→ c6:Cetuximab: c6	kfCetuximab_2* c6:Cetuximab* c6
c6:Cetuximab + c6← c6:Cetuximab: c6	kfCetuximab_2* c6:Cetuximab: c6
Cetuximab_v11	
Cetuximab + c532 → c532:Cetuximab	kfCetuximab_1*Cetuximab*c532
Cetuximab + c532 ← c532:Cetuximab	kfCetuximab_1*c532:Cetuximab
Cetuximab_v12	
c532:Cetuximab + c532 → c532:Cetuximab:c532	kfCetuximab_2*c532:Cetuximab*c532
c532:Cetuximab + c532 ← c532:Cetuximab:c532	kfCetuximab_2*c532:Cetuximab:c532
Cetuximab_v13	
Cetuximab + c524 → c524:Cetuximab	kfCetuximab_1*Cetuximab*c524
Cetuximab + c524 ← c524:Cetuximab	kfCetuximab_1*c524:Cetuximab
Cetuximab_v14	
c524:Cetuximab + c524 → c524:Cetuximab:c524	kfCetuximab_2*c524:Cetuximab*c524
c524:Cetuximab + c524 ← c524:Cetuximab:c524	kfCetuximab_2*c524:Cetuximab:c524
Cetuximab_v15	
Cetuximab + c529 → c529:Cetuximab	kfCetuximab_1*Cetuximab*c529
Cetuximab + c529 ← c529:Cetuximab	kfCetuximab_1*c529:Cetuximab
Cetuximab_v16	
c529:Cetuximab + c529 → c529:Cetuximab:c529	kfCetuximab_2*c529:Cetuximab*c529
c529:Cetuximab + c529 ← c529:Cetuximab:c529	kfCetuximab_2*c529:Cetuximab:c529
Cetuximab_v17	
Cetuximab + c530 → c530:Cetuximab	kfCetuximab_1*Cetuximab*c530
Cetuximab + c530 ← c530:Cetuximab	kfCetuximab_1*c530:Cetuximab
Cetuximab_v18	
c530:Cetuximab + c530 →	kfCetuximab_2*c530:Cetuximab*c530

c530:Cetuximab:c530	
c530:Cetuximab + c530 ← c530:Cetuximab:c530	kfCetuximab_2*c530:Cetuximab:c530

Table A.4.: Implemented Pertuzumab reactions.

Name	Reaction Rate
Pertuzumab_v1	
Pertuzumab + c141 → c141:Pertuzumab	kfPertuzumab_1* Pertuzumab *c141
Pertuzumab + c141 ← c141:Pertuzumab	kfPertuzumab_1* c141:Pertuzumab
Pertuzumab _v2	
c141:Pertuzumab + c141 → c141:Pertuzumab:c141	kfPertuzumab_2*c141:Pertuzumab*c141
c141:Pertuzumab + c141 ← c141:Pertuzumab:c141	kfPertuzumab2*c141:Pertuzumab:c141
Pertuzumab_v3	
Pertuzumab + c155 → c155:Pertuzumab	kfPertuzumab_1*Pertuzumab*c155
Pertuzumab + c155 ← c155:Pertuzumab	kfPertuzumab_1*c155:Pertuzumab
Pertuzumab _v4	
c155:Pertuzumab + c155 → c155:Pertuzumab:c155	kfPertuzumab_2*c155:Pertuzumab*c155
c155:Pertuzumab + c155 ← c155:Pertuzumab:c155	kfPertuzumab2*c155:Pertuzumab:c155

Table A.5.: Implemented U0126 reactions.

Name	Reaction Rate
U0126_v1	
U0126 + c47 → c47:U0126	kfU0126_1*U0126*c48
U0126 + c47 ← c47:U0126	kfU0126_1*c47:U0126
U0126_v2	
U0126 + c48 → c48:U0126	kfU0126_1*U0126*c48
U0126 + c48 ← c48:U0126	kfU0126_1*c48:U0126
U0126_v3	
U0126 + c74 → c74:U0126	kfU0126_1*U0126*c74
U0126 + c74 ← c74:U0126	kfU0126_1*c74:U0126
U0126_v4	
c47:U0126 + c45 → c48:U0126	k44U0126_1*c47:U0126*c45
c47:U0126 + c45 ← c48:U0126	Kd52U0126_1*c48:U0126
U0126_v5	
c47:U0126 + c72 → c74:U0126	k44U0126_1*c47:U0126*c72
c47:U0126 + c72 ← c74:U0126	Kd52U0126_1*c74:U0126

Supplementary Note 9: Leaderboard feedback and formation of the final leaderboards

For the network inference and time-course prediction sub-challenges participants were provided with feedback via weekly leaderboards. There were separate leaderboards for the experimental data tasks and *in silico* data tasks. Frequency and content of feedback was chosen to give a balance between actively engaging participants and avoiding overfitting of models to the test data. The number of weekly leaderboards varied between four and seven depending on the sub-challenge and task. We think it highly unlikely that this small amount of feedback would have allowed teams to systematically optimize model parameters based on performance on the held-out test data. Indeed, a grid search over parameter space would typically require many calls to the scoring function. In addition, metrics provided gave information about overall performance only. For example, mean AUROC, mean rank and mean z-score were provided for the network inference sub-challenge experimental data task leaderboard, but no scores were provided for individual *(cell line, stimulus)* contexts. This prevented teams from adapting their methods based on known failure or success in individual contexts (an extreme example of such adaptation would be to use a completely different method for each context in order to optimize performance on the test data).

For the time-course prediction sub-challenge, one of the metrics provided in the leaderboards was mean RMSE, where mean values were calculated over *(cell line, phosphoprotein)* or *(test inhibitor, predicted node)* pairs (**Supplementary Note 6**). Note that due to scaling issues, mean RMSE does not assign the same weighting to each pair and so is not correlated with mean rank. For this reason, mean rank was the metric used to assign final team rankings and we do not advise use of mean RMSE in analyses.

After the weekly leaderboard phase, participants were able to make one further final submission. A final leaderboard was then formed by taking the most recent submission for each team. At final submission, teams were asked to provide an initial brief method write-up and these were used to identify and remove duplicates from the leaderboard (teams with different names, but same methodologies and participants). Additional metrics were provided in the final leaderboards, which can be found on the Synapse pages describing the challenge, together with all weekly leaderboards (https://www.synapse.org/HPN_DREAM_Network_Challenge).

Supplementary Note 10: Removal of correlated submissions from analyses for the network inference sub-challenge

For the network inference sub-challenge, a number of teams in the final leaderboard did not provide any additional information regarding their approaches (34 teams for the experimental data task and 26 for the *in silico* data task). It is possible that some of these teams used very similar approaches, for example, if a team changed its team name during the challenge. Including such submissions in our analyses could introduce bias and we therefore sought to remove them (in particular, for formation of the aggregate submission networks and for comparison of scores between the experimental and *in silico* data tasks). Pearson correlations were calculated between predicted edge scores for each pair of teams; for the experimental challenge, a correlation was calculated for each *(cell line, stimulus)* context and these were then averaged. If two teams had (average) correlation of more than 0.75, in either of the tasks, then the lower ranked team was removed from analyses for both tasks (**Supplementary Table 2**).

Supplementary Note 11: *DREAMTools* – a software package containing DREAM challenge scoring functions

The scoring functions for the HPN-DREAM challenge are provided within "*DREAMTools*", a software package that provides scoring functions used during the different DREAM challenges (Cokelaer et al., 2015). This Python package is available on GitHub (https://github.com/dreamtools/dreamtools) and on the Python Package Repository (PyPI) (https://pypi.python.org/pypi/dreamtools).

The HPN-DREAM scoring functions return AUROC scores for the network inference sub-challenge (SC1) and RMSE scores for the time-course prediction sub-challenge (SC2); see Online Methods for full details of scoring procedures. For the network inference sub-challenge experimental data task (SC1A) the function provides an AUROC score for each *(cell line, stimulus)* context together with a mean AUROC score, calculated across contexts. For the *in silico* data task (SC1B) a single AUROC score is returned. For the time-course prediction sub-challenge experimental and *in silico* data tasks (SC2A & SC2B) a RMSE score is provided for each *(cell line, phosphoprotein)* pair (SC2A) or each *(test inhibitor, predicted node)* pair (SC2B). Mean RMSE scores, calculated across pairs, are also provided, but this metric does not provide a good indication of performance (**Supplementary Note 9**).

Using the *DREAMTools* library, a Python script can be written to score a submission. For instance, the network inference sub-challenge (SC1A) can be scored as follows:

```
from dreamtools.dream8.D8C1 import scoring
sc1a = scoring.HPNScoringNetwork(sc1a_submission.zip)
sc1a.compute_all_aucs()
sc1a.get_auc_final_scoring()  # mean AUROC
sc1a.auc                      # gives the individual AUROC for the 32 contexts
```

where "sc1a_submission.zip" is the filename of the submission and this file should be formatted as described on the Synapse challenge pages (https://www.synapse.org/HPN_DREAM_Network_Challenge). See https://github.com/dreamtools/dreamtools/tree/master/dreamtools/dream8/D8C1 for scripts for the other sub-challenges.

The *DREAMTools* library can also be used to rank a new submission relative to the submissions on the final leaderboard (i.e. provides the final ranking that would have been attained had the new submission been officially submitted to the challenge). See https://github.com/dreamtools/dreamtools/tree/master/dreamtools/dream8/D8C1 for further details.

References

Cokelaer, T. *et al*. DREAMTools: a Python package for scoring collaborative challenges [version 1; referees: 3 approved with reservations]. *F1000Research* **4**, 1030 (2015).