Data Generation

 Jesse: please use the active voice in all your writing whenever possible, e.g. GeneNetWeaver can easily generate … *In-silico* data for testing gene network inference algorithms can be easily generated using the software GeneNetWeaver (Marbach et al., 2010) (Prill, Marbach, Saez-Rodriguez, & Sorger, 2010) (Marbach, Schaffter, & Mattiussi, 2009) (available at ?). Using this software, we can extract sub-networks and generate kinetic models (i.e. expression data) from gene networks. Jesse: what includes? Some dream database? Use the active voiceIncluded are networks from E. Coli, Yeast, and the DREAM 3 and 4 challenges, but new networks can be added as well. When generating kinetic models and datasets for those models, there are several parameters to be aware of. In our running example, we set the parameters as follow

The first parameter is whether or not autoregulatory interactions should be removed from the kinetic model. That is, whether a gene can regulate itself. In the DREAM competitions, this setting was set to “Yes”. The reason for removing the autoregulatory interactions was two-fold: autoregulatory intereactions are difficult to infer and they are (according to ???) uncommon in the organisms DREAM was using as the basis for its models. using them makes the problem much more difficult and, in the networks of the organisms DREAM was basing its models off of, they were uncommon.

[Jesse: you’ll notice that I’ve shortened this. We want every word to count. It’s a matter of consideration towards the reader.] The “Coefficient of noise term” parameter adjusts how much noise is in the dynamics of the model. For DREAM, this is set to $0.05$. This parameter is adding noise during the generation of the expression data between each time point, rather than adding Gaussian noise at the end, and is intended to simulate variations in the actual dynamics of the system being modeled. [Jesse: please give an example of what 0.05 means.]

DREAM adds a mixture of normal and log-normal noise to the expression values after their generation. In our running examples, we follow the DREAM setting of the parameter “Model of noise in microarrays” of ??. We leave the default values of … for the standard deviation of the normal and/or log-normal noise added.

Most of the other parameters in GeneNetWeaver are fairly self explanatory, but there is also a detailed guide available on the GeneNetWeaver website (?). You should say how you set them with a couple of word explanation or refer to some site to state their settings.

The Running Examples

 Over the course of this text, we will have two running examples of gene networks. The first is a small network of only 5 genes, and the second is a network of 200 genes (large enough to show some of the combinatorial issues, but not so large as to require excessive computational resources). The data for each of these networks were generated using GeneNetWeaver with the DREAM4 settings. That is, they have noise in the dynamics during the generation of the expression values, and have measurement noise added to the expression values after generation, but have no autoregulatory interactions. Both networks are sub-networks in a larger simulated E. Coli gene network. We have generated time series, knockout, knock-down, and steady state data for each network.

 The small network is pictured in Figure 1. We can see that there is a central gene that regulates the four genes to which it points. Two of these edges are inhibitory, and the other two are excitatory. [Jesse: in genomics, we use repressive and inductive. Please stick with that] One of the genes in each of these pairs also has an excitatory edge to the other gene in its pair. This network is fairly simple, but exhibits a common feature of gene networks, in that the behavior of some genes can exhibit an affect on many other genes.

 This feature of gene networks is even more apparent in Figure 2. We see many “hubs” of genes, where one or a few genes control many genes around them. This network also has a few genes that do not have edges to any other genes. We’ll use this network as a full-scale test of each algorithm we present.





* Note: Remake these network graphs

cMonkey – Reiss et al, 2006

* Biclustering algorithm
* Useful for pairing down extremely large datasets
* Incorporates a wide variety of data, much of it publically available and not experiment dependent
* Can treat each cluster like a gene in inference algorithms
* (Reiss, Baliga, & Richard Bonneau, 2006)

DFG4GRN – Krouk et al, 2010

* ODE based
* Few parameters (3)
* Only requires time series data
* Models the noise in the data, removes it, then uses an ODE to model the “idealized” values
* Works well on small datasets

(Krouk, Mirowski, LeCun, Shasha, & Coruzzi, 2010) NIR – Gardner 2003/Gregoretti 2010

* ODE based
* NIR was extremely computationally expensive, and could not handle networks over 100 because of time constraints. However, recently, Gregoretti 2010 developed a parallel implementation of it. This allows NIR to be used on large networks with a computing cluster.
* Parameter:
	+ Max(reskK): The maximum number of ingoing edges into any node.
* Performs extremely well. Consistently better than ARACNe and BANJO by the numbers in the Gregoretti paper.
* Estimates parameters of differential equations using multiple linear regression.
* Uses a linearized version of the differential equation AX = -P, where P is a vector containing the initial effects of the perturbations to each gene. This may not be known, which is a disadvantage of NIR.
* (Gardner, 2003)(Gregoretti, Belcastro, Di Bernardo, & Oliva, 2010) Check with Alex and Aviv to see what they think of this.

Inferelator 2.0 – Greenfield, Et al. 2010

* Mix of ODE and MI
* Tied for first on DREAM4 100 gene network challenge with Pinna et al. (2010)
* Three steps in pipeline:
	+ Step 1: Calculate mean corrected z-scores on steady state data. Ok, this will require some explanation. This step, while it is optional, is powerful but requires knockout data for each gene in the dataset.
	+ Step 2: Calculate the time-lagged Context Likelihood of Relatedness between each gene. This step consists of three substeps:
		- Substep 1: Model temporal changes with an ODE
		- Substep 2: Calculate static and dynamic MI between each pair of genes.
		- Substep 3: Apply a filter, trimming weaker pairs of genes and enforcing sparsity. This is based on Z-scores.
	+ Step 3: Inferelator 1.0: learn a sparse dynamical model of regulation for each gene by regulators in some set of genes. These regulators can be selected by the previous two steps, or with a known list of transcription factors. LARS is used to implement an l\_1 constraint and enforce sparsity.
* (Greenfield, Madar, Ostrer, & R Bonneau, 2010) (Madar, Greenfield, & Ostrer, 2009)(Madar, Greenfield, & Vanden-Eijnden, 2010)

Jesse: we may want to break things down a bit more and then reconstruct algorithms. For example, we may want to have a section on biclustering, a section on handling knockout data (MI etc), a section on general steady state data, and a section on dynamical models. (ODE/Bayesian/DFG etc) Then we can have a framework that mixes and matches these. Your nice result on Piotr’s data would be such a mix and match.

Time-Delay ARACNe - Zoppoli,et al 2010

* Uses Mutual Information
* Original formulation, ARACNe, was unable to infer directionality of edges, but with time delay extension, it can
* Parameters:
	+ K: The how many steps to look ahead for influence?
	+ Pruning tolerance:
* Works in 3 steps:
	+ Step 1: Detect time point at which each gene becomes excited/inhibited
	+ Step 2: Estimate the joint probability between gene\_a at time t and gene\_b at time t+k, trying to find the best k, then calculate the mutual information between those genes using the joint probability. Because the MI is calculated using the time shifting above, directionality can be inferred. If information between a pair is above a certain threshold (calculated automatically by bootstrapping on the data type), then a directed edge is drawn between the nodes.
	+ Step 3: Prune the gene networks using the data processing inequality and a corresponding threshold (default: 15%). This essentially penalizes indirect edges and only accepts them if they are strong

BANJO

* Dynamic Bayesian Networks
* Works well when there are far more experiments than genes
* Does not make assumptions about linearity or non-linearity of relationships
* First order Markov model
* Uses greedy search with many random restarts, simulated annealing, or genetic algorithms to create the networks. The networks are then evaluated based on scoring criteria. These scoring criteria are the innovate parts of this algorithm.
* (Yu, Smith, Wang, & Hartemink, 2004)

Jesse: and Bayesian to replace some of the steps above (I think just the dynamic network inference)

NTW (Lai, et al 2011)

* Similar to NIR in that it uses the linearized form of AX = -P.
* ODE based model
* Does not have the restriction that NIR does where P must be known *a priori*. P is instead estimated using the top X% of gene expression values. Sum of squared errors is used to calculate the error between A and P at each iteration.
* Can imbed known information (non-null edges) into network easily
* Performs well, on par with NIR according to data presented in paper.
* (Lai, Yang, Wu, & Liu, 2011) [Jesse: Ok, goes straight in the ODE section]

Pinna et al. (2010)

* Uses only steady state data
* Z-Score based
* Simple, easy, works if you have perfect data (knockouts for each gene in turn)
* Tied for first with Inferelator 2.0 on DREAM4 100 gene network (very similar to their mean-calculated z-score step)
* (Pinna, Soranzo, & La Fuente, 2010) [Jesse: ok an approach to steady state data]

Gardner, T. S. (2003). Inferring Genetic Networks and Identifying Compound Mode of Action via Expression Profiling. Science, 301(5629), 102-105. doi:10.1126/science.1081900

Greenfield, A., Madar, A., Ostrer, H., & Bonneau, R. (2010). DREAM4: Combining Genetic and Dynamic Information to Identify Biological Networks and Dynamical Models. PloS one.

Gregoretti, F., Belcastro, von, Di Bernardo, D., & Oliva, G. (2010). PLoS ONE: A Parallel Implementation of the Network Identification by Multiple Regression (NIR) Algorithm to Reverse-Engineer Regulatory Gene Networks. PloS one.

Krouk, G., Mirowski, P., LeCun, Y., Shasha, D., & Coruzzi, G. (2010). High-resolution dynamics of transcriptome responses to NO3- in Arabidopsis roots: Molecular physiology and predictive modeling. Genome Biology, Preprint, 1-33.

Lai, D., Yang, X., Wu, G., & Liu, Y. (2011). Inference of gene networks—application to Bifidobacterium. Bioinformatics.

Madar, A., Greenfield, A., & Ostrer, H. (2009). The Inferelator 2.0: a scalable framework for reconstruction of dynamic regulatory network models … Conference of the ….

Madar, A., Greenfield, A., & Vanden-Eijnden, E. (2010). DREAM3: Network Inference Using Dynamic Context Likelihood of Relatedness and the Inferelator. PloS one.

Marbach, D., Prill, R. J., Schaffter, T., Mattiussi, C., Floreano, D., & Stolovitzky, G. (2010). Revealing strengths and weaknesses of methods for gene network inference. Proceedings of the National Academy of Sciences, 107(14), 6286-6291. doi:10.1073/pnas.0913357107

Marbach, D., Schaffter, T., & Mattiussi, C. (2009). Generating realistic in silico gene networks for performance assessment of reverse engineering methods. Journal of ….

Pinna, A., Soranzo, N., & La Fuente, de, A. (2010). From Knockouts to Networks: Establishing Direct Cause-Effect Relationships through Graph Analysis. PloS one, 5(10), e12912.

Prill, R., Marbach, D., Saez-Rodriguez, J., & Sorger, P. (2010). Towards a rigorous assessment of systems biology models: the DREAM3 challenges. PloS one.

Reiss, D., Baliga, N., & Bonneau, Richard. (2006). Integrated biclustering of heterogeneous genome-wide datasets for the inference of global regulatory networks. BMC Bioinformatics, 7(1), 280. doi:10.1186/1471-2105-7-280

Yu, J., Smith, V., Wang, P., & Hartemink, A. (2004). Advances to Bayesian network inference for generating causal networks from observational biological data. Bioinformatics.