Inferring Causal Networks in Plants

Inferring casual networks in planets can be done using many techniques oijoijoijwords  
  
General idea of causality from the book

The goal of causal inference is to discover whether the behavior of some node X causes the behavior of some node Y to change. If an increase in X’s value causes Y’s value to increase, then removing X from the network will prevent Y’s value from increasing. The value of X can also be manipulated in order to control the value of Y. Further, there may be several causal factors X1, 2, … Xk influencing Y some positively and some negatively. In our context nodes the Xs and Y are genes and their behaviors are expression values, but these ideas are also applicable in many other contexts such as ecological networks and so on..

Causal links may be detected experimentally by

1. Mutation: manipulating the behavior of a node (a particular gene in our case) and observing its effects on the network by performing:

* Experiments that excite the behavior of X.
* Experiments that depress the behavior of X.

2. Time Series: Causal links may also be inferred by

* Experiments that change the condition of an organism and then measure the behaviors of X and Y with closely spaced time points.

3. Steady State: General correlations can be found from experiments that change the condition of an organism and wait for the steady state. Effectively this is a degenerate case of the time series.

Overview of data types available (steady state, knockout, overexpression,  
and time series).  
-- Do we want to have a blurb about what the data looks like here or will this audience be familiar enough with that where it is fine? [Just a short description would help]

There are four basic types of experimental data available for use in gene network inference: Steady state, knockout, overexpression, and time series. Steady state data is a measurement taken after a perturbation is introduced to the network, and the network is allowed to settle into a “steady state” where the expression values of the genes have stopped changing. [Do we want an example here?] Knockout data is where a gene is removed from the organism. [Sometimes the knockout is the only perturbation.] Then a perturbation is introduced to this new gene network, and the results are compared to data that did not have the gene knocked out. Overexpression is similar to knockout data, except that instead of removing a gene, it is locked into a state of constantly being excited. A time-series experiment is when a perturbation is introduced to a network, such as in a steady-state experiment, but instead of waiting for the network to settle data is recorded at multiple time points until the network reaches a steady state. [Could add a sentence about how we can infer causality from each of these if we want] [Not needed but thanks]

Different tools (list with one sentence summary including the data  
types it works well with)  
There are many different tools available for gene network inference, encompassing a wide variety of theoretical approaches. Each of these tools uses a different approach to extract information from different data types. The basic idea behind all of these algorithms is to examine each possible edge between each pair of genes and give it a score, then rank those edges and select some number of the highest scored edges.

[Maybe these should go in a table?] No I like the prose, but they should be associated with ready they are for mutation, time series, or steady state.

Median-Corrected Z-Scores (MCZ) - Requires a dataset where each gene in the dataset is knocked out in turn. The idea is that if gene X influences gene Y, then knocking out gene X should change the value of gene Y. The amount of change in the knockout condition is compared to the median value of that gene across all experiments.

Network Identification by Multiple Regression (NIR) – Uses steady state data. Multiple-regression is used to causal edges in a network.

Gene Network Inference with Ensemble of Trees (GENIE3) – Uses steady-state data. GENIE3 works by creating a large number of regression trees, ranking the potential regulators for each gene from each tree, and then combining those ranked lists so that the most likely regulators are selected for each gene.

Context Likelihood of Relatedness (CLR) – Uses over-expression or knockout data. CLR computes the mutual information between each pair of genes and from that calculates the probability that each gene X is a regulator of gene Y.

Convex Optimization – Uses steady-state data. Convex optimization is a technique used to find weights that minimize some cost function.

Time-Delay ARACNE – Uses time-series data. Calculates the mutual information between gene X at time t and gene Y at time t+1 to build a network, and then prunes the weak edges.

Time-Lagged Context Likelihood of Relatedness (tlCLR) – Uses steady-state and time-series data. An extension of the CLR algorithm that takes into account differences in time in order to establish directionality between the edges.

Inferelator – Uses steady-state and time-series data. Inferelator uses differential equations to learn a sparse dynamical model for each gene.

Dynamic Factor Graphs (DFG) – Uses time-series data. DFG models the experimental noise in the data, subtracts that noise model out of the data, and then creates a network by learning sparse dynamical models for each gene.

Bayesian Network Inference with Java Objects (BANJO) – Uses time-series data. BANJO models each gene’s expression value at a time t by some combination of the expression of genes at time t-1.

Prose summary of results on simulated data  
(e.g. why combining different approaches  
can work etc.)

Each of the above approaches use different techniques and different data types to extract information about causal edges from the data. An algorithm using a regression method to infer edges may be extracting different information from the data than an algorithm using mutual information. The output from a wide array of algorithms covering different data types and different theoretical/statistical approaches can be combined to form a “consensus network.” There are many ways to combine these outputs; the two approaches we will discuss are “pipelines” and “ensemble networks.”

A pipeline is a series of algorithms strung together in a series, so that the output of one algorithm may be used in the following algorithm. Most of the algorithms listed above use a random starting point to begin their network inference. One use of a pipeline is to replace that random starting point with the inferred network from a previous algorithm. This can help network inference by removing some weak edges early, so then a later algorithm is dealing with a smaller dataset. It can also give good “candidate” edges with high weights, so the next algorithm can confirm those with different data or a different technique.

[FIGURE OF PIPELINE]

An ensemble network is a network stitched together from the outputs of many other algorithms. For example, an ensemble network can be created using a weighted sum of the networks created by inference algorithms. These weights can be found using a gold standard training network, and then applied to data where a gold standard is not known.

[FIGURE OF ENSEMBLE WORKFLOW]

Simulation experiments showing which data types help the most.

sims

Case study of Dynamic Factor Graphs on our time series data  
- Present DFG

Dynamic Factor Graphs (DFG) [Krouk, et al] is an algorithm that uses ordinary differential equations to create a model of experimental noise in the data, remove that noise, and then create a network model from the “noiseless” data. It uses time-series data with replicates. DFG strongly pushes weak edges to 0 using Least Angle Regression (LARS). This helps DFG create a parsimonious model that uses only the strongest edges in the network and removes everything else.

DFG has three different parameters that must be set, eta\_z, lambda\_w, and tau [I’ll fill these in with the actual symbols]. Eta\_z is the weight on the noise term. Lambda\_w is the importance of the noise model compared to the actual observed data. Tau is a weight on the importance of the amount of time between time points. DFG is relatively robust to the values of these parameters, but it is still important to set them in the correct ballpark.

To test DFG, 10 and 100 gene gold standard networks were generated using GeneNetWeaver [ref]. Time-series data was generated using these generated networks. DFG was then run using different parameter values of eta\_z, lambda\_w, and tau.

[ 10 gene network figure ROC ]

[ 100 gene network figure ROC ]

On these generated datasets, DFG does not perform particularly well. It is possible that this is because we are only using time-series data to generate this network, and this data does not have the rich gene-by-gene information that something like knockout data can give. However, we can use an algorithm that uses this data to generate a starting point for DFG. In this case we have selected Median-Corrected Z-Scores (MCZ), and generated the set of knockout data from the networks that this algorithm requires.

First a network is created using MCZ. This network is then pruned so that a certain percentage of the ranked edges are used, and this pruned list is used as a starting point for the DFG inference algorithm.

[ 10 gene network figure ROC MCZ-DFG]

[ 100 gene network figure ROC MCZ-DFG]

The MCZ-DFG pipeline performs much better than DFG alone. This is because of the good starting point created by MCZ. In this particular test, MCZ alone is actually the top performer. However, because the MCZ-DFG pipeline builds a dynamical model of the time-series data of the network, predictions can be made about new time points or new experiments (such as what happens if a gene were knocked out, or overexpressed). This cannot be done with MCZ alone. [Jesse, is it really true that this can’t be done alone. After all, it would give a causal network wouldn’t it or does it give data only about knockouts? Also, should we use inferelator and MCZ here?]

Enhanced case study with some overexpression data

Dfg overexpression results

[Perfect. The last four points are the key things that I want from you before you get busy on Jan 22.]

Jesse, Also, I think we want to repeat the results on real Arabidopsis data from the Krouk paper and also the improvements that even a few overexpression results give us. I think this is just cut and paste from one of our grant proposals.

What isn’t cut and paste is a systematic study to show that (i) time series may be more useful than steady state experiments (mutation experiments are more difficult so shoudn’t be compared). Can you do such a thing with dreamweaver.

(ii) more time points and two replicates may be more useful than fewer time points and three replicates.]