
Senior Thesis in Mathematics

# Active Learning Experimental Design of Bayesian Networks

*Author:*

Frances Hung

*Advisor:*

Dr. Jo Hardin

Submitted to Pomona College in Partial Fulfillment

of the Degree of Bachelor of Arts

May 4, 2019

**Abstract**

The importance and use of experimental design in biological applications was one motivation for my thesis [Sverchkov and Craven, 2017]. We go over a theory overview of Bayesian statistics, Bayesian networks, and experiments in networks. We then test different edge-wise priors and explain their effect on expected information gain of experiments.

# Contents

# Chapter 1

# Introduction

Scientists are often interested in constructing cause-and-effect networks describing biological processes and models. From observed data, however, multiple potential causal networks can be proposed, so additional biological experiments are usually needed to narrow the potential candidates down. To cut down on cost and time expenses, researchers can determine what experiments will narrow down the candidate field most by evaluating potential experiments on the proposed networks. Each experiment in a network consists of altering chosen nodes (variables) and seeing in real life how that change propagates throughout the different networks [Pearl, 1995]. A score is assigned to each experiment based on how much more precise the network becomes due to that experiment, and the experiments with the highest scores are the best candidates for real-life experiments.

I'll be exploring the experiment-ranking process using Bayesian networks.

A Bayesian network consists of nodes representing uncertain variables and directed edges representing cause-effect relationships, where each node has multiple possible discrete values and each value's probability is conditional on parent node input. We can use the joint distribution of the variables to update each node based on priors and new information [Heckerman, 2008]. Another way to look at Bayesian networks is to consider the local distribution function of each node independently; the function is the probability of a node being a certain discrete value given its parents and own probability parameters. The most familiar form such a local distribution function takes is that of a certain probabilistic distribution (binomial, multinomial, etc.). However, the distribution functions can also be generalized linear models or neural networks. A Bayesian network is an amalgation of the distribution functions for each node.

This thesis will focus on refining networks which already are good representations of the observed data they represent, so we first cover how to arrive at good network representations in Chapters 2 and 3. In summary, we use a hill-climbing algorithm to iteratively update potential networks until a maximum expected posterior is reached. In the hill-climbing algorithm, we use the properties of Bayesian networks to calculate the posterior Bayesian scores of each modified graph from the prior graph and data [Gamez et al., 2011]. In Chapter 4, we then study how to quantify the structural information we gain from performing experiments [Ness et al., 2017] and in Chapter 5, we test different priors and their effects on the expected information gain across

an example network's nodes.

# Chapter 2

# Bayesian Statistics

## 2.1 Introduction

This chapter is a brief introduction to Bayesian statistics, a basis for Bayesian networks. A Bayesian probability of an event is someone's degree of belief in the true value of the event. This is different from classical probability, where probability is measured using a relative frequency approach. The difference between the two schools of thought can be seen when we consider $\Theta$, the Bayesian equivalent of a frequentist random variable (which we call an uncertain variable instead). We aim to find the probability distribution representing our belief in the variable's true value, while in classical statistics we would try to find a probability distribution representing the observable data on that variable, but not one of our belief in the variable's true value [Sarwan, 2016].

As an example of a Bayesian statistic scenario, we can use a six-sided die where the possible true probability values of rolling each side are represented by the six-dimensional vector $\Theta = (\theta^{(1)}, \ldots, \theta^{(6)})$ where $\theta^{(1)} + \cdots + \theta^{(6)} = 1$. Each component of the vector, $\theta^{(i)}$, itself has a probability distribution and together, they form a multidimensional joint distribution [Heckerman, 2008]. Here, the probability of vector $\Theta$ can be represented by a Dirichlet distribution with six shape parameters, one for each possible side rolled. In other words, each possible value of the die roll has a corresponding probability distribution representing our belief in the probability of rolling that number. The probability density function for a Dirichlet distribution is

$$f(\theta^{(1)}, \ldots, \theta^{(N)}; \alpha_1, \ldots, \alpha_N) = \frac{\Gamma(\alpha)}{\Pi_{i=1}^{N}\Gamma(\alpha_i)}\Pi_{i=1}^{N}(\theta^{(i)})^{\alpha_i - 1} \tag{2.1}$$

where $\alpha_i$ is the shape parameter for the ith possible category, N is the number of categories, $\theta^{(i)}$ is the probability of observing the ith category where $\square_{i=1}^{N}\theta^{(i)} = 1$, and $\alpha$ is the sum of all $\alpha_i$ [Heckerman, 2008]. In terms of our example, $\theta^{(i)}$ is the probability of rolling the ith side, and $\alpha_i$ is the shape parameter of the probability distribution of rolling the ith side. The aim of Bayesian statistics in this example's context would be to update the Dirichlet probability density function to better reflect the data we see when we roll the die. Using the updated probability density function, we can then find the expected value of the density function and describe how confident we are in the expected value based on the distribution. One benefit of Bayesian statistics

is the ability to incorporate prior knowledge before updating the probability density function (in contrast to classical statistics, which formalizes learning from observed data only).

## 2.2 Bayes' Rule

An integral part of being able to better estimate $\theta$ is Bayes' Rule, the cornerstone of Bayesian statistics. Bayes' Rule says we can use prior knowledge and observed data $D$ to update our $\theta$ distribution:

$$p(\theta|D, \ ) = \frac{p(\theta| \ )p(D|\theta, \ )}{p(D| \ )}$$

.

In the die example, is represented as $\alpha_i$, the shape parameters of the Dirichlet distribution which represents our beliefs in the value of $\theta$ [Heckerman, 2008].

### 2.2.1 Prior, Posterior, and Likelihood

The prior $p(\theta| \ )$ represents our previous belief in $\theta$ before we take into consideration any new data, and the posterior $p(\theta|D, \ )$ represents our updated belief in $\theta$ after we take into consideration the new data. The ability of Bayesian statistics to draw from previous knowledge is one of the main concrete differences between it and classical statistics [Sarwan, 2016].

**Definition 2.1** *A* $prior$ *distribution* $p(\theta| \ )$ *of* $\theta$ *contains our confidence in*

*the true values of $\theta$, represented as a probability distribution, before we take into account new data $D$. The term is optional and is used here just to clarify the dependence of the prior on previous information [Heckerman, 2008].*

**Definition 2.2** *A* ***posterior*** *distribution $p(\theta|D, )$ of $\theta$ contains our confidence in the true values of $\theta$, represented as a probability distribution, after we take into account new data $D$ [Heckerman, 2008].*

In our die example, the prior distribution of $\Theta$ can be in the form of a Dirichlet distribution described above, with neutral parameters if we have no good initial guess of the side probabilities. Two possibilities of a neutral Dirichlet would be a distribution with all shape parameters equalling zero or one. Such a neutral distribution represents our lack of knowledge (or surety of belief) in the $\Theta$ distribution [Heckerman, 2008].

The posterior distribution in the example will take the same form as the prior: a Dirichlet pdf. However, the parameters will be different due to multiplying the prior by the likelihood term $p(D|\theta)$, which represents the likelihood of seeing our data $D$ conditional on the value of $\theta$.

**Definition 2.3** *A* ***likelihood*** *$p(D|\theta)$ of observed data $D$ is the likelihood of seeing $D$ assuming a given value of $\Theta$, $\theta$ [Heckerman, 2008].*

In our die example, the probability of seeing observed data ($n_i$ rolls of each ith side) in terms of $\Theta$ is the multinomial pdf

$$f(n_1, \ldots, n_6; \theta^{(1)}, \ldots, \theta^{(6)}) = \frac{n!}{n_1! \ldots n_6!} \Pi_{i=1}^{6} (\theta^{(i)})^{n_i} \tag{2.2}$$

where $n$ is the total number of observed rolls, $\theta^{(i)}$ is the prior probability of the ith variable state, and $n_i$ is the observed number of rolls of the ith side. We are assuming our rolls are mutually independent and can be modelled through multinomial sampling. Bayes'equation gives the following:

$$p(\theta|D, \ ) = \frac{Dir(\alpha_1, \ldots, \alpha_6) \times Multinom(\{n_1, \ldots, n_6\}, \{\theta^{(1)}, \ldots, \theta^{(6)}\})}{p(D)} \tag{2.3}$$

where the Dirichlet and multinomial pdfs are those in Equations 2.1 and 2.2 and the denominator is the expression derived below in Section 2.2.2.

### 2.2.2 Marginal Probability

The denominator of the Bayes' equation is the marginal probability, the unconditional probability of seeing $D$. $P(D)$ is a scaling constant for the posterior distribution, so it doesn't affect the shape of the distribution. However, it is still important in comparing different models via the Bayes' factor. We can write the marginal probability as follows:

$$p(D) = \int_{\theta} p(D|\theta, \ )p(\theta| \ )d\theta$$

We assume a Dirichlet representation of p($\theta|$ ) and multinomial representation of our observed data. Since the likelihood of seeing our entire dataset with k possible values is the multinomial pdf (Eq. 2.1) and $\theta$ has a Dirichlet

pdf (Eq. 2.2), $p(D)$ is equivalent to

$$p(D) = \frac{n!}{n_1! \dots n_N!} \frac{\Gamma(\sum_{i=1}^{N} \alpha_i)}{\prod_{i=1}^{N} \Gamma(\alpha_i)} \int_{\theta} \prod_{i=1}^{N} (\theta^{(i)})^{n_i + \alpha_i - 1} d\theta$$

To simplify, we multiply in a constant and its inverse:

$$p(D) = \frac{n!}{n_1! \dots n_N!} \frac{\Gamma(\sum_{i=1}^{N} \alpha_i)}{\prod_{i=1}^{N} \Gamma(\alpha_i)} c^{-1} \int_{\theta} c \prod_{i=1}^{N} (\theta^{(i)})^{n_i + \alpha_i - 1} d\theta$$

and set

$$c = \frac{\Gamma(\sum_{i=1}^{N} \alpha_i + n_i)}{\prod_{i=1}^{N} \Gamma(\alpha_i + n_i)}$$

so that the integral is of the complete Dirichlet probability density function. The integral equals 1 because the integral of any probability density function over the entire space is 1. We rearrange to get:

$$p(D) = \frac{n!}{n_1! \dots n_N!} \frac{\Gamma(\sum_{i=1}^{N} \alpha_i)}{\Gamma(\sum_{i=1}^{N} \alpha_i + n_i)} \prod_{i=1}^{N} \frac{\Gamma(\alpha_i + n_i)}{\Gamma(\alpha_i)}$$

This closed-form expression isn't especially neat, but it is a way to get the marginal probability of observing our data from just observed data counts $n_i$ and prior shape parameters $\alpha_i$.

## 2.3 Predicting Probability of New Data

Once we have the posterior distribution of our parameter $\theta$, a goal would be to evaluate the probability of seeing observed data given our posterior. To

predict the probability of seeing the next data point $x_{n+1}$ with discrete value $x^{(k)}$, we take the expected value of the probability of seeing our data and the data point in question. Finding the expected value involves integrating, over the entire $\theta$ parameter space, the probability $\theta^{(i)}$ of observing the next category [Heckerman, 2008]. For the remainder of the paper, we use $d\theta$ to denote integration over the domains of each parameter in the vector $(\theta^{(1)}, \ldots, \theta^{(N)})$.

Put simply, we are weighting the probability of seeing the value $x^{(k)}$ by its corresponding posterior distribution. To show the weighted posterior is equivalent to $p(x_{n+1} = x^{(k)}|D)$, we let A be the event $x_{n+1} = x^{(k)}$. Then

$$P(A) = \int P(A, \theta) d\theta$$

since for the marginal probability $P(A)$ we must consider all possible values of $\theta$. We can rewrite $P(A, \theta)$ in terms of conditional probabilities:

$$\int P(A, \theta) d\theta = \int P(A|\theta) P(\theta) d\theta$$

We condition $P(A)$ on the data we observe (a subset of the population of interest):

$$P(A|D) = \int P(A|\theta, D) P(\theta|D) d\theta$$

We conclude that

$$p(x_{n+1} = x^{(k)}|D) = \int p(x_{n+1} = x^{(k)}|\theta) p(\theta|D) d\theta$$

We can now predict the probability of specific dice rolls in our example. Assuming we have a Dirichlet distribution on $\theta$ and that our data follows a multinomial sampling distribution:

$$
\begin{aligned}
p(x_{n+1} = x^{(k)}|D) &= \int p(x_{n+1} = x^{(k)}, \theta|D)d\theta \\
&= \int p(x_{n+1} = x^{(k)}|\theta)p(\theta|D)d\theta \\
&= \int \theta^{(k)} p(\theta|D)d\theta \\
&= \int \theta^{(k)} \frac{1}{\beta(\alpha)} \Pi_{i=1}^{N} (\theta^{(i)})^{\alpha_i + n_i - 1} d\theta \\
&= \int \frac{\alpha_N + n_N}{(\sum_{i=1}^{N} \alpha_i + n_i + 1) - 1} \frac{1}{\beta(\alpha\ )} (\theta^{(k)})^{\alpha_k + n_k} \Pi_{i=k} (\theta^{(i)})^{\alpha_i + n_i - 1} d\theta \\
&= \frac{\beta(\alpha\ )}{\beta(\alpha)} \int \frac{1}{\beta(\alpha\ )} (\theta^{(k)})^{\alpha_k + n_k} \Pi_{i=k} (\theta^{(i)})^{\alpha_i + n_i - 1} d\theta
\end{aligned}
$$

where $\alpha = \{\alpha_1 + n_1, \ldots, \alpha_N + n_N\}$, $\alpha\ = \{\alpha_1 + n_1, \ldots, \alpha_k + n_k + 1, \ldots, \alpha_N + n_N\}$, $\beta(\alpha\ ) = \frac{(\Pi_{i \neq k} \Gamma(\alpha_i + n_i)) \times \Gamma(\alpha_k + n_k + 1)}{\Gamma(\sum_{i=1}^{N} \alpha_i + n_i + 1)}$, and $\beta(\alpha) = \frac{\Pi_{i=1}^{N} \Gamma(\alpha_i + n_i)}{\Gamma(\sum_{i=1}^{N} \alpha_i + n_i)}$. The integral is over the pdf of a Dirichlet distribution so the whole integral simplifies to equal 1. Expanding out $\beta(\alpha)$ and $\beta(\alpha\ )$, this is equivalent to

$$
\begin{aligned}
&= \frac{\Pi_{i=1}^{N} \Gamma(\alpha_i\ )}{\Gamma(\sum_{i=1}^{N} \alpha_i\ )} \times \frac{\Gamma(\sum_{i=1}^{N} \alpha_i + n_i)}{\Pi_{i=1}^{N} \Gamma(\alpha_i + n_i)} \\
&= \frac{\Gamma(\alpha_i + n_i + 1)}{\Gamma(\alpha_i + n_i)} \times \frac{\Gamma(\sum_{i=1}^{N} \alpha_i + n_i)}{\Gamma(\sum_{i=1}^{N} \alpha_i + n_i + 1)} \\
&= \frac{\alpha_i + n_i}{\sum_{i=1}^{N} \alpha_i + \sum_{i=1}^{N} n_i}
\end{aligned}
$$

From the above equations, we can find the probability of the next observation being a particular category using only the prior shape parameters, the number of data samples $n$, and $n_i$, the number of times we see that category in the data. Assuming a Dirichlet prior and multinomial likelihood, knowing the counts of seeing each category is sufficient for us to recalculate the probabilities of seeing each category.

There are some distributions where the prior and posterior distributions come out as nicely as the Dirichlet-multinomial example in this chapter. In particular, having exponential family (binomial, Poisson, multinomial, etc.) likelihoods allows for relatively simple integration and a closed form [Heckerman, 2008]. Monte Carlo methods allow us to use distributions outside of the exponential family as likelihoods.

# Chapter 3

# Bayesian Networks and Active Experimental Learning

## 3.1 Bayesian Networks

In this section, we describe how causal relationships are represented as networks and how Bayesian statistics applies to updating these networks. More specifically, we talk about node-specific posterior distributions and how they relate to the overall posterior [Heckerman, 2008].

### 3.1.1 Representation

The goal of a Bayesian network is to model causal relationships between different uncertain variables $X_1, X_2, \ldots, X_n$. For this paper, we assume each $X_i$ is a discrete categorical variable. These uncertain variables are related to

one another with a set of conditional independence assertions. In a network, each uncertain variable $X_i$ is represented by a node, and a directed edge from node $X_i$ to node $X_j$ represents a conditional dependence of $X_j$ on $X_i$. We define Bayesian networks more succinctly in the following definition.

**Definition 3.1** *A* **Bayesian network** *has nodes* $\{X_1, \ldots, X_n\}$*, each representing one of n uncertain variables, and directed edges* $X_i \rightarrow X_j$ *representing conditional dependence of variable* $X_j$ *on* $X_i$ *[Heckerman, 2008].*

Each node $X_i$ has a set of parent nodes which send incoming edges into $X_i$. We denote these parents of $X_i$ as $Pa_i$. Since $X_i$ is by definition conditionally dependent on its parents, the state of $X_i$ can be calculated based solely on its parents' states [Heckerman, 2008]. The probabilities of possible states of $X_i$ can be represented with $\Theta_i$ in a similar way to the die in the previous chapter.

To update networks in a Bayesian manner, we can start with a network prior and use samples of data to update the network. We start by focusing on a simpler problem: how to update posterior distributions of each node ($\Theta_i$). As an initial step, we must understand how to represent the probability of each node in relation to the probability of the entire network. Given a pre-defined structure $S^h$ of variables $X_1, \ldots, X_n$, $p(x_i|pa_i, \theta_i, S^h)$ is the local probability function of observing a single variable $X_i$ at its current state (with the parents $pa_i$ and $\theta_i$ specified by the existing $S^h$). Because each node's local probability function depends only on immediate parents and all

edges are directed, we can assume that the local probabilities of all nodes are mutually independent. Therefore, we can define the joint probability function for the whole graph as the product of all local probability functions [Heckerman, 2008]:

**Definition 3.2** *The* ***joint probability function*** *for a network is*

$$p(x|\theta_s, S^h) = \prod_{i=1}^{n} p(x_i|pa_i, \theta_i, S^h)$$

*, given that the local probabilities* $p(x_i|pa_i, \theta_i, S^h)$ *are mutually independent.*

### 3.1.2 Updating Network Probabilities

Much like for single variables, we use data to update our beliefs about network structure and dependencies. Since the individual variables have independent probabilities, we can update our beliefs about each $X_i$ separately. In order to calculate the probability of $X_i = x_i$, we need to be able to update $\theta_i$, our belief in the true value of $X_i$.

Because $\theta_i$ can have different distributions based on different parent configurations and the state of interest of $X_i$ itself, we can add subscripts to differentiate unique situations [Heckerman, 2008]. $\theta_{ij}^{(k)}$ is the probability of observing $X_i$ at state $k$ with parents $pa_j$. In particular, $\theta_{ij} = (\theta_{ij}^{(1)}, \ldots, \theta_{ij}^{(k)})$ is a vector representing the probabilities of the different k states of $X_i$ with parents $pa_j$. For each $\theta_{ij}$, our goal is to get the posterior distribution $p(\theta_{ij}|D, S^h)$, which depends on a prior $p(\theta_{ij}|S^h)$ and likelihood $p(D|\theta_{ij}, S^h)$. We have the

following variant on Bayes' Rule:

$$p(\theta_{ij}|D, S^h) = \frac{p(\theta_{ij}|S^h)p(D|\theta_{ij}, S^h)}{P(D)}$$

We make the following assumptions to make our calculation of the posterior easier:

1. The data $D$ is **complete**; that is, there is no missing data.

2. The $\theta_{ij}$ vectors are independent from one another and therefore follow **parameter independence**.

Using the previous two assumptions, we can then write the posterior as the following:

$$p(\theta|D, S^h) = \Pi_{i=1}^{n}\Pi_{j=1}^{q_i}p(\theta_{ij}|D, S^h) \tag{3.1}$$

Each $p(\theta_{ij}|D, S^h)$ can be written as a single-variable $\theta$ distribution from the previous chapter. To find each of these single-variable posteriors, we use the methods from Chapter 1 to calculate each posterior using the observed data.

Once we have the posterior distribution, we can calculate the probabilities of seeing a particular node outcome in the network. To obtain predictions for next sample $x_{n+1}$, we average over $\theta_S$, the possible parameters for the

network structure. In the following equations, $j$ and $k$ are dependent on $i$:

$$\begin{aligned} p(x_{n+1} = x^{(k)}|D, S^H) &= E_{P(\theta_S|D,S^h)}(\prod_{i=1}^{n} \theta_{ij}^{(k)}) \\ &= \int (\prod_{i=1}^{n} \theta_{ij}^{(k)}) p(\theta|D, S^h) d\theta_S \\ &= \prod_{i=1}^{n} \int \theta_{ij}^{(k)} p(\theta_{ij}|D, S^h) d\theta_{ij} \end{aligned}$$

### 3.1.3 Learning Network Structure

Calculating structure posterior probabilities works similarly to finding network probabilities; we want to find $p(S^h|D)$ where $S^h$ is the discrete uncertain variable representing the distribution of possible true structures. Theoretically, we start with a prior distribution of all possible structures which could represent our true network [Heckerman, 2008]. We use Bayes' Theorem to find the posterior distribution of the possible true structures:

$$p(S^h|D) = \frac{p(S^h)p(D|S^h)}{p(D)} \tag{3.2}$$

In 3.2, $p(D|S^h)$ is the marginal likelihood of the data, given by the products over $i$ and $j$ of single-variable i-j marginal likelihoods:

$$p(D|S^h) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_k)}{\Gamma(\alpha_{ij} + n_{ij})} \prod_{k=1}^{K} \frac{\Gamma(\alpha_{ijk} + n_{ijk})}{\Gamma(\alpha_{ijk})}$$

From the posterior distribution we calculate using Bayes' Rule, we can

pick the structure with the highest posterior probability $S^h$ as the best representation of the data:

$$S^h = \text{argmax}_{S^h}\{P(S^h|D)\}$$

Finding a structure in this manner is not actually feasible in real life. In the next section, we explain an algorithm which allows us to find a structure with high posterior probability in real life [Heckerman, 2008].

## 3.2 Computationally Finding Structure Fits

The following sections will mostly be concerned with finding a near-best structure in real life, in which we face some obstacles. Unfortunately, for example, finding the best structural fit of a network to data is an NP-hard process [Heckerman, 2008]. The difficulty lies in there being more than exponential in $n$ possible structures for a network with n nodes. In our discrete variable $S^h$, we'd have to consider a very large number of graphs. To facilitate an otherwise intractable process of finding a good-fit DAG (directed acyclic graph) to our data, we use a bootstrapping technique. Before going into detail about the bootstrapping algorithm, we define a Bayesian scoring function and elaborate on how networks are scored.

### 3.2.1 Bayesian Scoring

Because the marginal probability $p(D)$ doesn't vary with network structure, we can use a Bayesian scoring metric proportional to the posterior probability of a network to quantify the goodness-of-fit of a network.

**Definition 3.3** *The* Bayesian score *of a network is*

$$f(S^h : D) = \log p(D|S^h) + \log p(S^h)$$

We try to find network configurations which maximize the Bayesian score.

### 3.2.2 Bootstrapping to Find Good Structure Fits

From the data we have on each uncertain variable in the network, we sample $r$ instances with replacement $N$ times, leading to a set of samples $D_1, \ldots, D_N$. We assign an initial empty graph representation to each bootstrap $D_i$.

From these empty graphs, we can improve our graph fits by considering neighboring graphs. Neighbor graphs are created from existing graphs by making one of three local changes (edge deletion, addition, or direction reversal). Because we need to avoid creating cycles (where we can start at a node and find a path back to that node through directed edges), there are $O(n^2)$ such possible changes for a given graph with $n$ nodes. We calculate the new Bayesian scores caused by each of the local changes (using Def. 3.3) and take the difference of these new scores with the corresponding scores of

the networks prior to the edge change. More formally, if we let $f(S^h : D)$ be the score for the whole graph, we can assume independence between individual nodes plus their corresponding parents and decompose $f(S^h : D)$ [Gamez et al., 2011]:

$$f(S^h : D) = \prod_{i=1}^{n} f(\{X_i, Pa_i)\} : D)$$

Then we can find score differences between neighboring and original graphs using one of the three below processes for each possible edge:

1. Add edge, $X_j \rightarrow X_i$: $f(X_i, Pa_i) \cup \{X_j\}) - f(X_i, Pa_i))$
2. Delete edge, $X_j \rightarrow X_i$: $f(X_i, Pa(X_i) \setminus \{X_j\}) - f(X_i, Pa_i))$
3. Reverse edge directionality (by deleting existing edge and adding an oppositely oriented one), $X_j \rightarrow X_i$: $[f(X_i, Pa_i) \setminus \{X_j\}) - f(X_i, Pa_i))] + [f(X_j, Pa(X_j) \cup \{X_i\}) - f(X_j, Pa(X_j))]$

We choose the neighboring graphs which yields the largest positive difference as our new best graphs $\{G_1, \ldots, G_N\}$. We stop altering the graphs when changes no longer improve the posterior score by more than a user-specified amount $\alpha$ [Gamez et al., 2011].

We then choose the final graphs with the best Bayesian scores as good-fit DAGs to our data.

**Result:** Bootstrap DAGs from Data

Starting parameters: data D, number of DAGs to learn N, stopping threshold $\alpha$, number of DAGs to return F;

**for** *t in 1:N* **do**

  Initialize empty graphs $G_t$, $G_t$;

  **do**

    $G_t = G_t$;

    $G_t$=Find Best Neighboring Graph($G_t$);

  **while** *Score*($G_t$)-*Score*($G_t$) > $\alpha$;

  return $G_t$

**end**

G=order $\{G_1, \ldots, G_N\}$ by score (decreasing);

return G[1:F]

**Algorithm 1:** Main Bootstrapping Algorithm

**Result:** Find Best Neighboring Graph

Starting Parameters: Graph G;

**for** *nodes* ($X_i$, $X_j$) *in* $G$ **do**

  **if** *edge* $i \rightarrow j$ *exists* **then**

    $Score^{ij}_{max}$ = max Bayesian score of {deleting $i \rightarrow j$, reversing $i \rightarrow j$, keeping $i \rightarrow j$};

  **if** *edge* $i \rightarrow j$ *doesn't exist* **then**

    $Score^{ij}_{max}$ = max Bayesian score of {adding $i \rightarrow j$, not adding $i \rightarrow j$};

  Apply edge process which yields $Score^{ij}_{max}$ to G;

**end**

return G

**Algorithm 2:** Finding Best Neighboring Graph

### 3.2.3 Converting DAGs to PDAGs

We regard our "best" graph with some doubt because some directed edges in our graph can be reversed while retaining the same posterior distribution. Because of the ambiguity of some edge directions, it is better to change our DAG (directed acyclic graph) network to a PDAG (partially directed acyclic graph) representation. We convert a DAG to a PDAG by taking the DAG and turning directed edges which follow the following conditions into undirected edges:

1. flipping directionality does not change the number of immoral v-structures (defined in Section 3.3)

2. child node of the directed edge is not a child node of an intervention in S

Using the above conditions to convert DAGs ensures that all DAGs mapped to a single PDAG have the same posterior probability [Ness et al., 2017].

## 3.3 V-Structures

In this section, we go more into detail about the first condition for orienting an edge: that the edge is part of a v-structure. V-structures are a specific directed-edge structure which encode a specific type of dependence structure between three connected variables. In order to describe v-structures, we must first describe dependence (d-separation).

The d in d-separation stands for dependence, and two variables $X$ and $Y$ are d-separated given other variables $O$ if $X$ and $Y$ are independent given $O$ (notation is $X \quad Y \,|O$) [Kuleshov and Ermon, 2019]. In simpler terms, $X$ doesn't contribute any additional information about $Y$ and vice versa if we know $O$ already. We can further formalize d-separation as follows. Two nodes are d-separated given observed nodes $O$ if the two nodes aren't connected by an active path. A path is active if for each triple of consecutive nodes $(X, Y, Z)$ either:

1. $X \rightarrow Z \rightarrow Y$ and $Z \;/\; O$
2. $X \leftarrow Z \leftarrow Y$ and $Z \;/\; O$

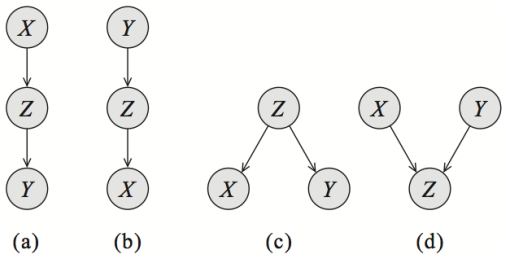


Figure 3.1: The four types of dependency structures

3. $X \leftarrow Z \rightarrow Y$ and $Z \;/\; O$

4. $X \rightarrow Z \leftarrow Y$ and Z or any descendent is in O

The last structure $X \rightarrow Z \leftarrow Y$ is a v-structure, which encodes a different dependency relation ($X \quad Y \,|O$ if $O$ is unobserved and $X \quad Y \,|O$ if $O$ is observed) than the first three structures. A v-structure is called, rather archaically, immoral if there is no edge between the parent nodes.

All variables which are d-separated given $O$ in a DAG $G$ are independent given $O$. We can generalize $G$ as a PDAG (partially-directed acyclic graph) since the dependency relations of the first three structures are identical and the last is distinct. In other words, we can change the direction of any edge in a DAG and preserve its dependency structure unless edge reversal induces a new immoral v-structure or destroys an existing immoral v-structure [Kuleshov and Ermon, 2019].

## 3.4 Summary

We now have a computationally efficient way of finding good structural fits of networks to data, and we know how to class graphs into more generic PDAGs. Bootstrapping structure fits and PDAGs will be useful in finding ideal interventions which further improve structural fits.

# Chapter 4

# Experiments and Information Gain

## 4.1 Methods

In this section, we will go over what experiments are, how the authors of bninfo implement the DAG-to-PDAG algorithm, and how we measure the effectiveness of experiments [Ness, 2015].

## 4.2 Experiments

From a physical science point of view, experiments consist of variables and the relationships between variables. By clarifying relationships between variables, scientists can predict future behavior and interaction between vari-

ables. For this section, we'll assume that variables have finite, discrete states. Variables and their true relationships can be represented by a Bayesian network; more specifically, variables and their relationships to the best of our knowledge can be represented by a PDAG.

As a recurring example, we'll use a problem pertinent to the first robot who autonomously carried out experimental design and hypothesis generation [Flatley, 2009]. Let's say Adam the Robot Scientist wants to predict whether his co-workers will be nice to him on a certain day or not. The variables he's interested in are the following:

- Whether his co-workers are nice to him or not (Nice?)
- Whether it's Friday or not (Friday?)
- Whether his co-worker is a Luddite (Luddite?)
- Whether his co-worker is having a bad day (Bad Day?)
- Whether Adam is happy (Happy Adam?)

The true network representation (4.1) has completely directed edges but is currently unknown. Our goal is to get to the true representation from a PDAG derived from the bootstrapping process (Section 3.2.2). In order to extrapolate the relationships between variables, Adam will need to perform experiments and get data.

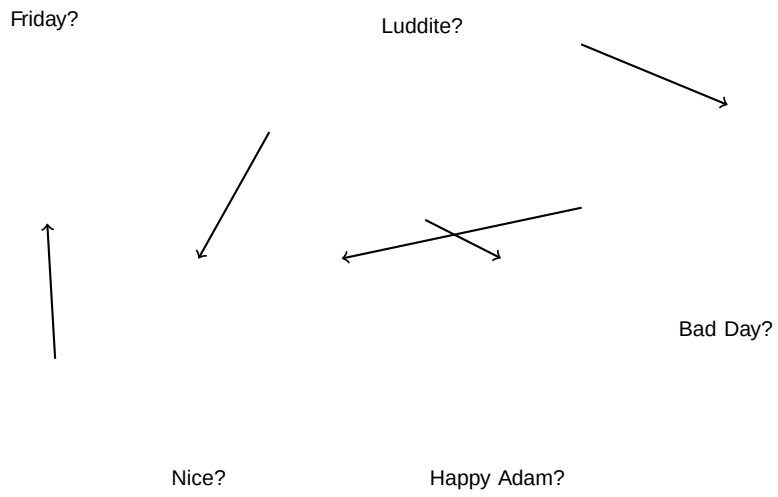


Figure 4.1: A complete network representation of Adam's variables.

### 4.2.1 What is an Experiment?

De nition 4.1 An experiment/intervention in the physical science world involves setting a variable to multiple states and seeing how other variables are a ected (or not a ected) by the di erent states. The outcome of the experiment is the existence of directed edges between the variable and other variables.

In Adam's example network, we can realistically control the states of several of the variables: whether Adam is happy (through some programming), for example, or whether Adam's coworker is having a bad day. If we set these variables to their various possible states, we can observe how these states affect their neighboring variables (Nice? in the case of the Happy node and Nice? + Luddite? in the case of the Bad Day node).

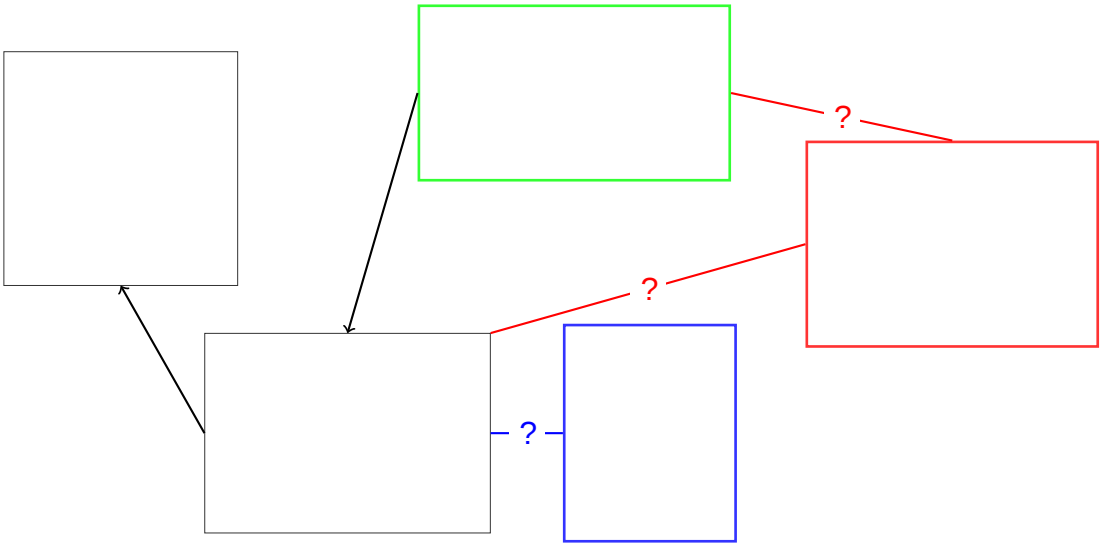


Figure 4.2: A PDAG output of the bootstrapping algorithm; we want to get from the PDAG to Fig. 4.1.

Once we perform an experiment on a variable, we know the directionality of causation between that variable and its neighbors [Pearl, 1995]. Either the variable has no e ect on a neighbor (in which case the edge is determined to be directed towards the variable), or the variable a ects the neighbor (and the edge is determined to be directed from the variable to the neighbor).

Because we would like to predict e ective experiments before we perform them in real life, however, we only have access to the high-posterior possible network ts from the previous chapter's bootstrapping algorithm (Fig. 1). We'd like to compare the PDAG network ts following from the possible network ts alone to the PDAG network ts resulting from di erent experiments on those network ts.

For each possible PDAG network t, we assume that an experiment on

a variable node will orient any incident edges, even if we don't know which direction those edges would be oriented. An experiment on the Bad Day node will orient the connections to the Luddite and Nice nodes, although we don't know in which direction. We can assume the orientation of incident edges because an experiment in real life would clear up ambiguous causality relations between the experiment node and incident nodes.

## 4.3 PDAGs in the Code

In order to convert each DAG into a corresponding PDAG, the code decides which edge orientations to x based on three conditions: whether the edge is part of a v-structure, whether either of its nodes is adjacent to an intervention node, and whether the edge has a prior skewed probability [Ness et al., 2017]. The last condition was proposed by the code's authors in addition to the two more standard ones (mentioned in the previous chapter). We go more into more detail about the skewed-prior condition below.

Ness de nes a skewed-prior edge as one which has an orientation probability other than exactly 0.5, but his de nition is rather stringent. If the probability of an edge is skewed, then that edge is marked as oriented during the DAG-to-PDAG process. The additional skewed-prior condition potentially results in more edges being marked as oriented than if we consider only the orientation conditions (v-structure preservation and intervention adjacency) of Section 3.2.2 alone.

I changed the code to make the de nition of a skewed-prior edge more exible: I can classify edges as unskewed if their probabilities lie within a user-de ned range (" ex") of 0.5.

```
Result: Returns PDAG from DAG
DAG G, interventions S;
for edge e in G do
    if e is in an immoral v-structure then
        Mark e's direction as xed;
    if e's child is adjacent to S then
        Mark e's direction as xed;
    if e has a skewed prior then
        Mark e's direction as xed
end
for edge f in un xed edges do
    Fix f if reversing directionality would change number of
     v-structures;
end
```

Algorithm 3: DAG to PDAG conversion

In the results chapter, I use a ex of 0, but the new de nition of skewedness may be useful for future exploration of the package.

## 4.4 Information Gain

From the previous sections, we see that interventions can improve our knowledge of the network structure: we now quantify the improvement. When we

apply interventions on a DAG, we may end up with more oriented edges than if we didn't apply any interventions. The difference in number of oriented edges caused by an intervention is the information gain of the intervention [Ness et al., 2017].

Definition 4.2 Given a PDAG induced by DAG $D$ and a PDAG induced by $D$ plus interventions $V$, the information gain of $V$ is

$$IG(D, V) = \text{\# edges oriented by \{D,V\}} - \text{\# edges oriented by \{D\}}$$

We can get a more stable estimate of the information gain of an intervention by averaging the information gain over multiple high-probability DAG fits $G = \{G_1, \ldots, G_P\}$. $\{G_1, \ldots, G_P\}$ are the $P$ DAGs with the highest Bayesian scores at the end of the bootstrapping algorithm (Algorithm 1).

Definition 4.3 Given PDAGs induced by DAGs $G = \{G_1, \ldots, G_P\}$, data $D$, and PDAGs induced by $G$ plus interventions $V$, the expected information gain or estimated information gain of $V$ is

$$EIG = \sum_{G_i \in G} IG(G_i, V) \cdot P(G_i | D)$$

The expected/estimated information gain is what we'll be using to quantify how much structural information an additional intervention will give us over the PDAGs derived from the top $P$ bootstrapped DAGs.

## 4.5 Summary

We went over what experiments/interventions are, and how we can quantify how they improve our knowledge of network structure via expected information gain.

# Chapter 5

# Testing Priors

## 5.1 The DREAM network

The DREAM network is a signaling network which represents how cell proteins interact and react to signals from the environment. Each protein is a node in the network and each edge is a regulatory relationship between two proteins [Ness et al., 2017]. It consists of 4 receptors tfna, tgfa, il1a, and igf1 along with other non-receptor nodes.

First, we simulate data from the known network to replicate the scenario where we have data about the network but are unsure of the actual network structure. We then use the bootstrapping algorithm to get initial high-probability DAGs (Algorithm 1). We can then investigate interventions as described in the previous chapter.

Figure 5.1: The DREAM network of cell proteins.

## 5.2 Structure Probabilities

In the code, the authors use an edge-wise notion of structure probability to define their structure priors. Given an edge from node $X_i$ to node $X_j$, they think of edge probability in terms of edge existence and edge directionality [Ness et al., 2017]. The number of edges adjacent to $X_i$ is the degree of $X_i$, the number of adjacent edges directed away from $X_i$ is its out-degree, and the number of adjacent edges directed towards $X_i$ is its in-degree.

Definition 5.1 The probability of existence of edge $\{i, j\}$ is denoted $P(i - j)$ or $\rho_{ij}$

Definition 5.2 The probability of edge orientation $\{i \to j\}$ is denoted

$P(i \to j \mid i \;\; j)$ or $\beta_{ij}$

If we let $\overline{I}_{ij}(G)$ be the indicator function of whether edge $\{i, j\}$ exists in graph $G$ and $\vec{I}_{ij}(G)$ be the indicator function of whether $\{i \to j\}$, then our prior for an entire graph is

$$p(S^h) = c \prod_{ij \in G} (1 \quad \alpha_{ij})^{1 \quad \overline{I}_{ij}(G)} (\alpha_{ij} \quad \beta_{ij})^{\overline{I}_{ij}(G) \vec{I}_{ij}(G)}$$

We incorporate the above prior into the bootstrapping process via the initial use of the prior $p(S^h)$ in the Bayesian score equation [Ness et al., 2017].

## 5.3 Motivation for Exploring Priors

A question which follows from the previous sections is how different prior edge probabilities affect the expected information gain for each potential intervention. The prior edge probabilities affect the initial bootstrapping process of finding good-fit DAGs to the data. In addition, whether each edge probability is considered skewed (Section 4.3) or not affects which edges are marked as oriented before taking into account experiments. Ultimately, marking edges as unskewed can increase the expected information gain for experiments, while priors can also drastically affect the bootstrapped DAG structure.

## 5.4 Methods

Ness considered an uninformative-prior case and an informative prior case. In the uninformative prior case, the authors assume that the probabilities for edge existence and edge directionality each follow a uniform distribution. With the informative prior, they assume a 0.147 probability for the existence of edges incident to receptor nodes (tnfa, tgfa, il1a, and igf1), and a nearly zero probability for them being directed towards the receptor nodes. They use the product of the existence and directionality probabilities as an edge-wise prior in building the initial DAGs and nding information gain of interventions [Ness et al., 2017].

Figure 5.2: Incoming receptor edges with ~0 prior probability of existence.

The authors multiply the existence and orientation probabilities to get an overall probability of a directed edge existing. Using the original, ~0, informative prior to inform the DAGs created through the bootstrapping process, the median out-degree of tnfa (the node with the highest expected information gain) over the DAGs is 5 while the median in-degree of tnfa over the DAGs is 0.

We use di erent distributions as priors in both the bootstrapping and intervention steps. We keep the ex constant at 0. We test di erent prior edge distributions: the original informative prior, a 0.25 receptor-edge-probability distribution, a 0.5 distribution, and a 0.75 distribution.

## 5.5 Results

The higher the probabilities of incoming edges to receptor nodes were, the lower the expected information gain of interventions on the receptor nodes. The decrease in EIG implies that the higher receptor edge priors either impact the DAG structure in a way which reduces the number of adjacent undirected edges in the bootstrapping step or increases the number of adjacent directed edges in the DAG-to-PDAG step.

### 5.5.1 Results by Prior

If the prior says it's extremely unlikely that incoming edges to receptors exist (as it did in the original parameters set by the authors), then either

Figure 5.3: Incoming receptor edges with 0.25 probability of existence.

an edge between any given receptor and another node doesn't exist at all or it's directed the other way (outwards from the receptor). Implementation of either of the two options can be re ected in the step where we bootstrap DAGs from data (Algorithm 1), which may result in a more diverse set of nal, good- t DAGs than the set created with a less extreme prior. The two options also potentially create situations where alternative structures are created to match the data we build from since the simplest explanation (incoming edges to receptors) for correlation may not be probable.

Compared to the original prior (Fig. 5.2), the information gains for graphs with a 0.25 prior of receptor edge probabilities (Fig. 5.3 )are similar in ranking. The median out-degree of tnfa is 5 while the median in-degree of

tnfa is 1. The degrees (Fig. 5.6) of the other nodes seem to support a correlation between out-degree and expected information gain.

Figure 5.4: Incoming receptor edges with 0.5 probability of existence.

If the prior edge probability of incoming edges to receptors is high (Fig. 5.4, 5.5), then the incoming edges to receptors are likely to exist in the DAGs produced by the bootstrapping process. If the edges stay directed during the DAG-to-PDAG process, interventions done on the four receptors do not count the receptor-directed edges towards their information gain.

Compared to the previous two priors, the information gains for graphs with a 0.5 prior of receptor edge probabilities (Fig. 5.4) are drastically different. All four receptor nodes (tnfa, il1a, tgfa, and igf1) now have the four lowest estimated information gains. The median out-degree for tnfa is 4, and

the median in-degree is 3. While the total degree of the tnfa node hasn't changed much, the ratio between out and in degrees has, drastically.

Figure 5.5: Incoming receptor edges with 0.75 probability of existence.

The information gains for graphs with a 0.75 prior of receptor edge probabilities (Fig. 5.5) follow a similar pattern as the previous 0.5 prior. The di erences in information gains between nodes are more pronounced, and the four receptor nodes still have the four lowest estimated information gains. The median out-degree for tnfa is 4, and the median in-degree is 4. In other words, as the probability for incoming edges to receptors gets higher, the ratio of out to in degrees for tnfa (and likely other receptors to a lesser degree) gets lower.

### 5.5.2 Expected Information Gain and Degree

Looking at a summary box plot (Fig. 5.6), there appears to be several main categories of expected information gain trends among di erent nodes. Each box in the top EIG plot represents the distribution of EIG over the top 50 PDAGs produced by the bootstrapping and DAG-to-PDAG algorithms. Each box in the bottom two degree plots represents the distribution of out and in degrees over the top 50 DAGs produced by the bootstrapping algorithm. For the receptors (the  rst four left proteins in our plots), lower incoming probabilities are correlated with higher information gain and higher probabilities with much lower information gain. Most of the other nodes exhibit monotonic trends, where expected information gain increases as incoming receptor edge probabilities rise. Some of these monotonic trend nodes show little change even though they exhibit the trend; these tend to have small expected information gain across the probabilities in comparison to other nodes. The other monotonic trend nodes show much more noticeable increases in expected information gain.

We can look at the in and out degrees of our nodes for all graphs after the bootstrapping process and before the DAG-to-PDAG conversion (Fig. 5.6). The DAG in and out degrees are closely related to the information gains of each node in each graph, which count the undirected adjacent edges in the corresponding PDAGs. The most obvious trend we see is that as the prior edge-probability increases, so do the in-degrees of the receptor nodes, indicating that the higher prior causes some of the prior-probability edges

to appear in our best- t DAGs. The non-receptor nodes have fairly stable median in-degrees.

When we look at the variability of the degrees across receptors, we notice that degrees of the receptor nodes are most variable when the prior edge probabilities are lowest. The di erence in variability supports the idea that our low priors force the bootstrapping process to create alternate, more complex structures to explain our data. If Ness had used a higher probability than the original prior, the DAGs created by the bootstrapping process would be quite di erent.

As the prior edge probabilities get higher, the variability of the non-receptor node degrees and expected information gains gets higher as well. It remains to be determined whether the trends we witness are due to the speci c nodes in question and how the data supports a certain structure, or whether putting directed edge priors in general has the same e ect on children nodes.

Figure 5.6: Box plots comparing expected information gain and degrees across di erent edge priors and nodes.

# Chapter 6

# Concluding Remarks

## 6.1 Discussion

We can hypothesize about why different uniform priors on certain nodes lead to drastically different information gains on those nodes, but it is difficult to quantify how and why. I'd like to further elaborate on the relationship between out-degree of receptor nodes and their expected information gain. Some further goals would be testing priors on different non-receptor nodes and finding an explicit quantitative relationship between in/out degrees and the expected information gain.

## 6.2 Conclusion

The prior edge probabilities play a key role in determining the expected

information gain of the receptor nodes. The importance of the prior edge probabilities is mostly seen in the bootstrapping phase, where different priors produce graphs with different ratios of in to out degrees across receptors. More analysis on the relationship between prior edge probabilities and expected information gain is needed.

# Bibliography

[Flatley, 2009] Flatley, J. (2009). Artificial intelligence solves boring science experiments, makes interns obsolete. https://www.engadget.com/2009/04/03/artificial-intelligence-solves-boring-science-experiments-makes/.

[Gamez et al., 2011] Gamez, J. A., Mateo, J. L., and Puerta, J. M. (2011). Learning bayesian networks by hill climbing: efficient methods based on progressive restriction of the neighborhood. *Data Mining and Knowledge Discovery*, 22(1-2):106.

[Heckerman, 2008] Heckerman, D. (2008). *A Tutorial on Learning with Bayesian Networks*, pages 33–82. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Kuleshov and Ermon, 2019] Kuleshov, V. and Ermon, S. (2019). Stanford cs228 notes. https://ermongroup.github.io/cs228-notes/representation/directed/.

[Ness, 2015] Ness, R. (2015). *bninfo: Queries and information theoritic operations on Bayesian networks.* R package version 1.0.

[Ness et al., 2017] Ness, R. O., Sachs, K., Mallick, P., and Vitek, O. (2017). A bayesian active learning experimental design for inferring signaling networks. In Sahinalp, S. C., editor, *Research in Computational Molecular Biology*, pages 134–156, Cham. Springer International Publishing.

[Pearl, 1995] Pearl, J. (1995). Causal diagrams for empirical research. *Data Mining and Knowledge Discovery*, 82(4):669–710.

[Sarwan, 2016] Sarwan, N. S. (2016). Bayesian statistics explained to beginners in simple english. `https://www.analyticsvidhya.com/blog/2016/06/bayesian-statistics-beginners-simple-english/`.

[Sverchkov and Craven, 2017] Sverchkov, Y. and Craven, M. (2017). A review of active learning approaches to experimental design for uncovering biological networks. *PLOS Computational Biology*, 13(6):1–26.