# Where's The Network?

Written by James D. Wilson

University of San Francisco

We've talked about many situations where networks arise; however, by this point you should ask yourself: when is network analysis actually beneficial? Even more, when should we view the underlying data as a network in the first place? In all situations, when considering a network model we suppose that we observe measurements on some collection of **actors** $\mathcal{A} = \{1, \ldots, n\}$. Whether or not one models $\mathcal{A}$ with a network model largely depends on your view. In the words of the famous statistician George Box once stated "all models are bad, some are useful." On the other hand, a network collaborator of mine once claimed that he viewed "every problem as a network partitioning problem." [1]

It is true that there are many situations in which a network representation of a system is natural. However, in some applications, we instead observe an $n \times p$ matrix of measurements, $\mathbf{X}$, on $\mathcal{A}$. In these situations, classical multivariate statistical analysis can be used rather than, or in addition to, network analysis. As a statistician, I tend to view network modeling and analysis as a methodology to *augment* multivariate analysis. That is, network analysis provides additional insights about a complex system beyond multivariate analysis alone. We will investigate exactly what network analysis can offer throughout the remainder of this class.

Here, we consider several problems and manifestations of measurements on $\mathcal{A}$ and when and how to construct a network in these situations.

## Naturally Occurring Networks

Often, network models provide the simplest and most natural representation of the actors $\mathcal{A}$. In particular this is the case whenever the measurements on $\mathcal{A}$ are **relational**. That is, we observe measurements that directly describe the interactions or relationships of members of $\mathcal{A}$. Networks are naturally occurring in a variety of applications. Arguably, the most common of these applications arise in sociology where researchers seek to describe interactions among a group of individuals. We provide a concrete example of a naturally occurring static network below.

[1] This quote, from Professor Peter J. Mucha at UNC Chapel Hill, is probably taken a bit out context, but he certainly is well-versed in community detection and network partitioning.

**Example 1.** Suppose that we wish to analyze the friendships among my $n$ Facebook friends. Then the collection of actors is given by $\mathcal{A} = \{\text{friends of mine on Facebook}\}$ and measurements are represented by the collection $\mathcal{M} = \{\mathbb{I}(u \text{ and } v \text{ are friends}) : u, v \in \mathcal{A}\}$. This system is therefore naturally represented by a static undirected and unweighted network, with node set $V = \{1, \dots, n\}$ and associated adjacency matrix $\mathbf{A}$ whose entries are of the form:

$$A_{uv} = \begin{cases} 1 & \text{if } u \text{ and } v \text{ are friends} \\ 0 & \text{otherwise} \end{cases}$$

Now, we consider a slightly more complicated example where the system is naturally represented by a multilayer network, or hypergraph.

**Example 2.** Suppose that we still want to analyze my Facebook friends, so that $\mathcal{A}$ is the same as in Example 1. However, this time we have three measurements that describe the relationships of my friends, including

- *Friendships*: $\mathcal{M}_1 = \{\mathbb{I}(u \text{ and } v \text{ are friends}) : u, v \in \mathcal{A}\}$
- *Dating Relationships*: $\mathcal{M}_2 = \{\mathbb{I}(u \text{ and } v \text{ have ever dated}) : u, v \in \mathcal{A}\}$
- *Classmates*: $\mathcal{M}_3 = \{\mathbb{I}(u \text{ and } v \text{ have shared a class in college}) : u, v \in \mathcal{A}\}$

This system is readily modeled using a multilayer network with three layers. Here, the node set again is $V = \{1, \dots, n\}$ for all layers, and layers are represented by the associated sequence of adjacency matrices $\{\mathbf{A}_1, \mathbf{A}_2, \mathbf{A}_3\}$, whose entries match the indicator variables from the measurements $\mathcal{M}_1$, $\mathcal{M}_2$, and $\mathcal{M}_3$.

## Networks as First Order Similarities

Unfortunately, it is not always immediate how a network can be used to represent a collection of actors $\mathcal{A}$. Consider for instance the case where we observe some mixture of continuous and discrete *non-relational* measurements on each actor $u \in \mathcal{A}$. As is standard in practice, we can represent the measurements on actor $u$ by a $p$-dimensional vector $\mathbf{x}_u$, where $p$ is the number of measurements. Generally, one represents the system using an $n \times p$ *design* matrix $\mathbf{X}$ whose rows are the observations $\{\mathbf{x}_1, \dots, \mathbf{x}_n\}$. To distinguish rows from columns, we will denote the $v$th column of $\mathbf{X}$ as $\mathbf{x}^v$.

**Practical Note 1.** In many instances one treats the variables (columns) of the matrix $\mathbf{X}$ as the actors of interest. This decision ultimately depends on the context of the application. The notion of similarity that we discuss here and in the following section can in generally be applied to either the rows (observations) of columns (variables) of $\mathbf{X}$. Throughout, we will interchangeably do this according to the application.

If our goal is to construct a network model for $\mathcal{A}$, we can directly rely on representing the relationships between each pair of actors using the **similarity** of their measurements. Let $S(u, v)$ denote the similarity of nodes $u, v \in \mathcal{A}$. Common examples of similarities include:

- **Correlation**: $S(u, v) = corr(\mathbf{x}_u, \mathbf{x}_v) \in [-1, 1]$
- **Gaussian similarity**: $S(u, v) = \exp\left(\sum_{j=1}^{p} -(x_{u,j} - x_{v,j})^2/2\sigma^2\right) \in [0, 1]$
- **k Nearest Neighbor**: $S(u, v) = \mathbb{I}(u \in \text{Ne}_k(v)) \in \{0, 1\}$
- **$\epsilon$-neighborhood**: $S(u, v) = \mathbb{I}(d(\mathbf{x}_u, \mathbf{x}_v) < \epsilon) \in \{0, 1\}$

Above, $d(\cdot, \cdot)$ is treated as a distance metric on $\mathbb{R}^p$, and $\text{Ne}_k(v)$ is the collection of the $k$ nodes that are closest to $v$ according to some distance metric on $\mathbb{R}^p$.

In the situation where we observe the measurements $\mathbf{X}$, the network model that represents $\mathcal{A}$ is the weighted network whose adjacency matrix is given by the entries $A_{uv} = S(u, v)$.

We provide a common example of using similarity to represent a gene-gene interaction network below.

**Example 3.** Suppose that we would like to analyze the interactions among a collection of $p$ genes for which we observe expression levels across $n$ individuals. This data is represented by the $n \times p$ design matrix $\mathbf{X}$, whose $u$th column $\mathbf{x}^u$ is the $n$-dimensional vector of expressions associated with gene $u$. The collection of actors is given by $\mathcal{A} = \{\text{genes } g_1, \ldots g_p\}$. One common way to represent an adjacency matrix for this system is to take the collection of pair-wise correlations among the genes:

$$A_{uv} = corr(\mathbf{x}^u, \mathbf{x}^v).$$

The resulting network is a weighted and undirected network with edge weights between -1 and 1.

It is important to note that $S(u, v)$ quantifies a *first order dependence* between nodes $u$ and $v$ - namely, it considers only the distance between the observations $\mathbf{x}_u$ and $\mathbf{x}_v$ with no further analysis of higher moments of the data. Sometimes a more formal understanding of the dependency between vertices is desired; however, the benefit of the similarity approach is that it provides a "quick and dirty" description of the relational structure observed in the design matrix $\mathbf{X}$ that is scalable to large networks.

## Coding Example

Consider the exercise in Section 7.3.1 of [1]. The data that we consider is a the gene expression values for 153 genes over 40 experimental conditions of *E. coli*. Below, we load the data, which is available in the `sand` package in `R`, and construct an unweighted graph where an edge represents whether or not the correlation between two genes is statistically significant at a 0.05 level.

Formally, to identify edges, we first calculate the empirical correlation between genes ($\widehat{\rho}_{uv}$) and then transform these values according to Fisher's transform given by:

$$z_{uv} = \frac{1}{2} \log \left[ \frac{1 + \widehat{\rho}_{uv}}{1 - \widehat{\rho}_{uv}} \right].$$

Under the null hypothesis $H_0 : \rho_{uv} = 0$, and assuming that the gene expression values are multivariate normal, the transformed data is well approximated by a Gaussian random variable with mean 0 and variance $(n-3)^{-1}$. So, we use this normal distribution to approximate which values are statistically significant under multiple testing and then construct a network from these significant values. This is done in the code below.

```
library(igraph)

#load the E.coli data
data(Ecoli.data)

#Get a heatmap of the gene expression data (when column centered and
#standardized)

heatmap(scale(Ecoli.expr), Rowv = NA)

#the above heatmap orders the columns (genes) according to hierarchical
#clustering and reveals relationships among the genes.
```

```
#Calculating the correlations between genes and normalizing
#according to the Fisher transform

correlations <- cor(Ecoli.expr)
#by default, the above takes correlations of the columns of the matrix

#Fisher transform
z <- 0.5 * log((1 + correlations) / (1 - correlations))

n <- dim(Ecoli.expr)[1] #number of observations

#calculate p-values from Normal distribution
z.vec <- z[upper.tri(z)] #coercing the upper triangle of z into a vector
corr.pvals <- 2*pnorm(abs(z.vec), 0, sqrt(1 / (n-3)), lower.tail = FALSE)

#Now, adjust for the p.values using the Benjamini-Hochberg multiple
#testing correction (correct thing to do when testing multiple values)

corr.pvals.adjusted <- p.adjust(corr.pvals, "BH")

#how many values are significant at a 0.01 level?
length(corr.pvals.adjusted[corr.pvals.adjusted < 0.01])
#this is quite large and should be investigated more thoroughly

#To complete the example, we'll now make a network out of this
e.coli.adjacency <- matrix(0, 153, 153)
#first fill in the upper triangle
e.coli.adjacency[upper.tri(e.coli.adjacency)] <- corr.pvals.adjusted < .01

#now fill in the lower triangle by taking the transpose
e.coli.adjacency <- e.coli.adjacency + t(e.coli.adjacency)

#convert this into an igraph object
e.coli.graph <- graph.adjacency(e.coli.adjacency, mode = "undirected")

#plot using the Kamada-Kawai layout
igraph.options(vertex.size=3, vertex.label=NA,
               edge.arrow.size=0.5)
plot(e.coli.graph, layout = layout.kamada.kawai)

#matrix image
image(Matrix(e.coli.adjacency))
```

In the next section, we discuss the probabilistic way of thinking about dependence in a graph through what is known as graphical models.

## Graphical Models

### Brief Review of Conditional Probability

Before we begin, we briefly discuss conditional probability and the notion of conditional independence of variables.

Let $X, Y$ be discrete random variables taking values in the countable sets $\mathcal{X}$ and $\mathcal{Y}$, respectively, and having joint probability mass function

$$p_{X,Y}(x,y) \;=\; \mathbb{P}(X=x, Y=y) \;\; x \in \mathcal{X},\; y \in \mathcal{Y}$$

By summing $p_{X,Y}(x,y)$ over $y \in \mathcal{Y}$ we recover the marginal probability mass function $p_X(x)$ of $X$. The marginal probability mass function $p_Y(y)$ is obtained similarly, by summing $p_{X,Y}(x,y)$ over $x \in \mathcal{X}$. The **conditional probability mass function** of $Y$ given $X$ is defined by

$$p_{Y|X}(y \mid x) \;=\; \frac{p_{X,Y}(x,y)}{p_X(x)} \;=\; \mathbb{P}(Y=y \mid X=x)$$

when $p_X(x) \neq 0$. If $p_X(x) = 0$ then it is common to define $p_{Y|X}(y \mid x) = 0$ as well. The conditional probability mass function of $X$ given $Y$ is defined in the same way. In general, this notion of conditional distributions extend to jointly continuous random variables and the **conditional probability density function** of $Y$ given $X$ is defined as

$$f_{Y|X}(y \mid x) = \frac{f_{X,Y}(x,y)}{f_X(x)} \quad x, y \in \mathbb{R}$$

where $f_X$ is the marginal probability density function describing the continuous random variable $X$ and $f_{X,Y}$ is the joint continuous density function for $(X, Y)$.

Graphical models rely on the notion of conditional independence, which we now (briefly!) describe. Suppose that $(X, Y, Z)$ are jointly distributed discrete random variables with associated probability measure $\mathbb{P}$. The variables $X$ and $Y$ are said to be **conditionally independent given** $Z = z$ if the following factorization holds

$$\mathbb{P}(X=x, Y=y \mid Z=z) = \mathbb{P}(X=x \mid Z=z)\; \mathbb{P}(Y=y \mid Z=z) \tag{1}$$

In words, conditional independence of $X$ and $Y$ implies that *given the value of of the random variable $Z$, the outcomes of $X$ and $Y$ do not influence one another.* Conditional independence for jointly continuous random variables is defined analagously. $X$ and $Y$ are said to be **conditionally dependent given** $Z = z$ if (1) does not hold.

## Gaussian Graphical Models

Gaussian graphical models are a popular and well-studied way to "construct" a network on observed multivariate data $\mathbf{X}$. Given an $n \times p$ design matrix $\mathbf{X}$, a **graphical model** is the graph $G = (V, E)$ on the actors $\mathcal{A} = \{\text{variables } 1, \ldots, p\}$. Let $\mathbf{x^u}$ denote the $u$th column of $\mathbf{X}$, which represents the $u$th variable. Let $\mathbf{X}_{-(uv)}$ denote all the columns of $\mathbf{X}$ excluding the $u$th and $v$th column. Then the graphical model is defined by the adjacency matrix $\mathbf{A}$ containing the entries:

$$A_{uv} = \begin{cases} 1 & \text{if } \mathbf{x}^u \text{ is conditionally dependent upon } \mathbf{x}^v \text{ given } \mathbf{X}_{-(uv)} \\ 0 & \text{otherwise} \end{cases}$$

The graphical model defined by $\mathbf{A}$ is often referred to as a *conditional independence* or *Markov* graph because it posits that two nodes are connected to one another if and only if the represented variables are conditionally dependent upon one another given the remainder of the network.

In general, calculating conditional probabilities is a difficult problem. Estimating a graphical model requires the calculation (or more likely, the estimation) of the joint probability distribution of $p$ variables. Without simplifying assumptions, this task is practically impossible. This leads us to one simplifying assumption that leads to a special type of graphical models known as Gaussian graphical models, described as follows. Suppose that the variables $\{X^1, \ldots, X^p\}$ are characterized by a multivariate Gaussian distribution with variance covariance matrix $\Sigma$. In this case, we have the following important result:

**Fact 1.** For all $u, v = 1, \ldots, p$, $X^u$ is conditionally independent of $X^v$ if and only if $\omega_{uv} = 0$. Notably, $\rho_{uv} = \omega_{uv}/\sqrt{\omega_{uu}\omega_{vv}}$ is the **partial correlation** between $X^u$ and $X^v$ given $\mathbf{X}_{-(uv)}$. The value $\omega_{uv}$ is the $u$, $v$th entry of the precision matrix $\Omega = \Sigma^{-1}$.

The **Gaussian graphical model** for the variables $\{X^1, \ldots, X^p\}$ with covariance matrix $\Sigma$ is the graph $G = (V, E)$ defined by $V = \{1, \ldots, p\}$ and $E$ is defined by the adjacency matrix $\mathbf{A}$ with entries

$$A_{uv} = \begin{cases} 1 & \text{if } |\omega_{uv}| > 0 \\ 0 & \text{if } |\omega_{uv}| = 0 \end{cases}$$

To fit a Gaussian graphical model to an observed design matrix $\mathbf{X}$, one typically calculates the empirical covariance matrix of the columns of $\mathbf{X}$, $\widehat{\Sigma}$ and constructs a graph by placing edges between nodes with large entries in $\widehat{\Omega} = \widehat{\Sigma}^{-1}$. In particular, one generally uses statistical testing techniques (not described) here to formally test the following hypotheses for each pair of nodes:

$$H_0 : \omega_{uv} = 0 \qquad vs. \qquad H_1 : \omega_{uv} \neq 0$$

**Practical Note 2.** In general, calculating the inverse of a matrix is computationally difficult (on the order of $O(n^{2.3737})$ in the best case scenario). So, fitting a Guassian graphical model to data with a large number of variables is computationlly challenging. This has brought about a significant amount of research in graphical models in statistics and computer science. See for example [2] for a book-level treatment on methods and approaches to fitting graphical models.

## Coding Example

Below, we provide a computational example of how to fit a Gaussian graphical model to the *E. coli* data from before. This relies on utilizing the package `huge` (high-dimensional undirected graph estimation) to perform the hypothesis tests above using a modeling-based procedure. We consider two selection procedures below (from Ch. 7.3.3 of [1]); however, we suggest digging more into the package and use of this method for better understanding.

```
install.packages("huge")
library(huge)

#create a huge object for analysis from the observed data matrix
#this is fitting the Gaussian graphical model to this data
huge.out <- huge(Ecoli.expr)

#Now, we need to select which partial correlation values are
#statistical significant. There are two primary ways of doing this
#The first is known to be prone to under-selection. This is seen
#in the empty graph formed using the "ric" criterion below

huge.opt1 <- huge.select(huge.out, criterion = "ric")
summary(huge.opt1$refit)

#This second method is known to select a moderate number of edges
#this is based on subsampling for variance stabilization
huge.opt2 <- huge.select(huge.out, criterion = "stars")
summary(huge.opt2$refit)
```

```
#Create a graph object and plot the result
GGM.graph <- graph.adjacency(huge.opt2$refit, mode = "undirected")
summary(GGM.graph) #153 nodes and 759 edges

igraph.options(vertex.size=3, vertex.label=NA,
               edge.arrow.size=0.5)
plot(GGM.graph, layout = layout.kamada.kawai)
```

## References

[1] Eric D Kolaczyk and Gábor Csárdi. *Statistical analysis of network data with R*, volume 65. Springer, 2014.

[2] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.