Introduction to Networks

Levi Lee

December 23, 2016

Why Networks?

A *network* is a collection of inter-related things. Because they focus on relationships among elements, networks have wide applications in many fields of study. In this handout, we focus on social networks, in which the individual elements are either individuals or groups (though they need not be human), called *actors* and the links between the actors are called *ties*. These ties can be friendships, collaborations, email exchanges and so on. Social networks have been a growing field of study since the 1930s. Even before the computer age, people have observed some unique relationships among individuals already. One famous example is Milgram's letter experiments in which Milgram wanted to explore how many people it required for a letter to reach between any two people. He discovered that the median number of people the letters had to go through was about six, giving rise to the popular term "six degrees of separation." We will see other examples of (smaller-scale) social networks in later sections.

What is a Network?

Basic Terminology and Definitions

While a network is collection of vertices and edges, more formally, we define a *graph* as an object $G = (V, E)$ such that E is the set of edges and V is the set of vertices. Each edge connects two vertices, and so for any two vertices $i, j \in V$, if i and j have a connection then the ordered pair $\{i, j\} = e \in E$. In this case, we say i is *adjacent* to j and that i and j are *neighbors*. A vertex is *incident* on an edge if that vertex is the endpoint of that edge. For example, a vertex i would be incident on $e = i, j$ if j does not connect with any other vertex in V. Here, we make no distinction in the ordering of vertices. That is, i, j is the same as j, i. Such a graph is considered *undirected*.

The *order* of a graph G is defined to be the number if vertices in G, denoted $N_V = |V|$. Similarly, the *size* of G is the number of edges in G, denoted, $N_E = |E|$

Graphs that have ordering to its edges, that is, $\{i, j\} \in E$ is different from $\{j, i\} \in E$, are called *directed graphs* (or *digraphs*), and the edges in a directed graph are called *directed edges* (or *arcs*).

We can also represent a network with just the connections that are present (or not present) between nodes by using an adjacency matrix. Given a graph $G = (V, E)$, the adjacency matrix is an N_V x N_V matrix in which:

$$A_{ij} = \begin{cases} 1 & \text{if } \{i, j\} \in E \\ 0 & \text{otherwise} \end{cases}$$

It is also possible to assign edge weights to individual elements of the adjacency matrix; for example, assigning an edge weight w_e to $e = \{i, j\} \in E$. Matrix $\mathbf{A}$ is symmetric if G is a non-directed graph, but may not necessarily be if G is a directed graph. This is because $\{i, j\} \in E$ does not necessarily guarantee that $\{j, i\} \in E$ for directed graphs. Similarly, the incidence matrix $\mathbf{B}$ is defined as:

$$B_{ij} = \begin{cases} 1 & \text{if vertex } i \text{ is incident to vertex } j \\ 0 & \text{otherwise} \end{cases}$$

While the adjacency matrix $\mathbf{A}$ is one data structure that fully characterizes a graph, size can become an issue very quickly, especially in the context of data structures and algorithms. For a graph with N_E vertices,

it's corresponding adjacency matrix will have N_E^2 elements. Thus, we can also consider an alternative: an *edge list* is a two-column list of all the edges and their corresponding vertices present in a graph.

To describe how to move about in a network, we define a *walk* in graph G to be a sequence that begins and ends with two vertices and alternates between vertices and edges. The sequence $(a_0, e_1, a_1, e_2, ..., v_{l-1}, e_l, v_l)$ denotes one path from vertex a_0 to vertex a_l, where for all $e_i \in E$, e_i connects points a_{i-1} and $a_i \in V$. Using this notation, we define the *length* of the walk to be the value l. More refined than a walk, a *trail* is defined to be a walk in which the sequence does not traverse through the same edge more than once. And even more refined still is a *path* in which a walk does not travel through the same vertex twice. A vertex j is *reachable* from another vertex i is there is a walk from i to j. G is *connected* if all vertices in G are is reachable by some other vertex in G.

A *circuit* is a trail in which the starting and ending vertices is the same. A *cycle* is a walk with length of at least three that starts and ends at the same vertex but all other vertices are unique. An acyclic graph contains no cycles. The length of the shortest path(s) between any two vertices is defined to be the *distance* (or *geodesic distance*). This distance is infinite if no path between two particular vertices exist. The *diameter* is the longest (geodesic) distance achieved in a graph. In a digraph, a *directed walk* is analogous to a walk, but uses arcs to get from one point to another.

Informally, a *subgraph* is part of a graph. $H = (V_H, E_H)$ is a subgraph of $G = (V_G, E_G)$ if V_H is contained in V_G and E_H is contained in E_G. In this case, the collection of edges in H can be any subset of the edges that connect the vertices of V_H found in G. An *induced subgraph* of G, which we denote $G' = (V', E')$ where V' is contained in V is a predetermined subset of vertices, and E' contained in E is the collection of all the edges that connect all the vertices in V' found in G. Additionally, the basic set-theoretic concepts of union, disjoint union, intersection, difference, and complement all extend naturally to graphs.

A graph is *complete* if every vertex is has an edge with all other vertices in the graph. An undirected graph is *simple* if it has no *loops* and *multi-edges*, meaning vertices do not have an edge that points to itself and any pair of vertices do not have more than one edge between them. Edges on a simple graph are called *proper edges*.

A *component* is a subgraph that is maximally connected. In a digraph, *weakly connected* means that it the underlying graph (i.e. creating a simple graph by replacing all arcs with regular edges) is also connected. A digraph is *strongly connected* if every vertex in the digraph itself is reachable by another vertex in the graph.

A *clique* is a graph or subgraph that is complete. A graph is *regular* if each vertex has an equal number edges connecting to it. That is, each vertex has the same *degree*.

Families of Graphs

A graph is a *tree* if it connected but does not contain any cycles. The disjoint union of trees create forests. A digraph that is also a tree is known as a *directed tree*. Directed trees usually have a *root vertex*, in which it is the only vertex that has direct paths to every other vertex on the graph. In these rooted trees, the root vertex is an *ancestor* while the vertex in directed paths of length one away is a *descendant*. An ancestor that is immediately above a vertex is a *parent*, and a descendant immediately below a vertex is a *child*. A *leaf* is a vertex without any descendants. *Rooted trees* are directed acyclic graphs (DAGs). Family trees and pedigrees are examples of tree graphs.

A *k-star* is a tree that has only one root and k leaves. Trees are *directed acyclic graphs* (DAGs), meaning it is a directed graph that has no cycles. A *bipartite graph* is a graph that consists of two disjoint sets of vertices that does not have edges within each sets pf vertices but only to opposing sets. Movie recommendations on media subscription services are an example of bipartite graphs. Based on the viewing history of the user, the subscription service tries to link users with movies to watch. However, it is never the case that users are paired with users or movies are paired with movies; they only connect with each other. We can get induced graphs, called *projections*, from the bipartite graph that explains the relationship between each disjoint set of points.

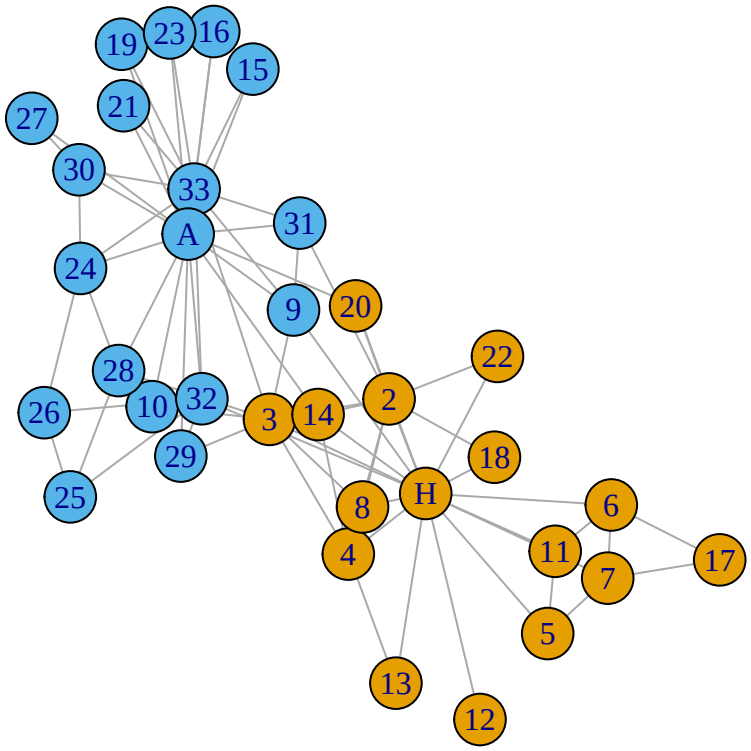
Vertex, Edges, and Graph Attributes

A graph can contain other information or data, called *attributes*, beyond just the collection of nodes and links. Incorporating these attributes, otherwise known as *decorating* a graph, can occur in both the nodes, links and even the entire graph itself. Both edge and vertex attributes can be discrete or continuous. An example of discrete is a binary case of positive or negative relationship between pairs of vertices. An example of continuous would then be the strength (i.e. weight) of the relationship between pairs. A *weighted graph* is a graph whose edges include weights (that are nonzero and between 0 an 1)

Example 1: Zachary's Karate Network

One famous (or infamous) social network, called "the karate club of Zachary," is one observed by anthropologist Zachary in the 1970s. Nodes indicate the 34 individual members of the karate club while the links indicate the 78 social ties among pairs of the members. As shown by the colors of the nodes, the clubs, at this particular moment in time, is being divided into two clusters centering around two nodes (the master and disciple of the dojo).

```
data(karate)
plot(karate)
```




```
vcount(karate) #34 people
```

```
## [1] 34
```

```
ecount(karate) #78 friednships
```

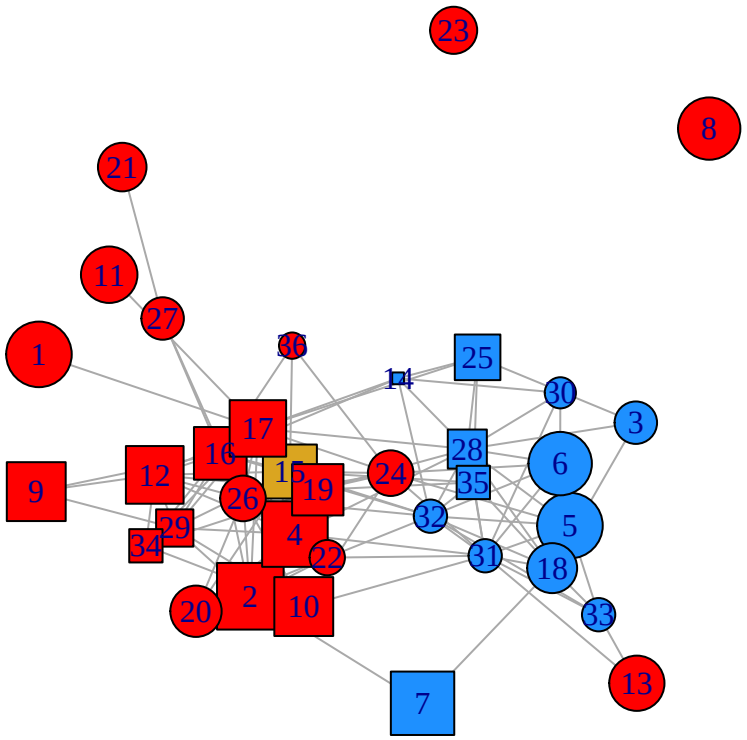
```
## [1] 78
```

It is possible to display more data than just the actors and their relationships. As the network shows, we can provide more information by assigning other visual cues–such as node shapes to determine gender, node size to indicate belt rank, and edge weights to determine the strength of the social tie–onto each member.

Example 2: Lawyers

Consider a network of lawyers that reflects the relational data collected by Lazega from a New England law firm consisting of over 70 lawyers to study cooperation in organizations. This network has 36 lawyers and 115 collaborations. Each lawyer is individually labeled with smaller numbers indicating seniority. Shapes for each lawyer indicate the type of practice: litigation (circles) and corporate (squares). The shapes' sizes indicate the relative number of years of having been in with the law firm. Color is used to indicate office location (red, blue, and yellow). Edges indicate collaboration between partners. The network below provides what is called a "visual summary" of the lawyer network data.

```
data(lazega)
# Office location indicated by color.
colbar <- c("red", "dodgerblue", "goldenrod")
v.colors <- colbar[V(lazega)$Office]
# Type of practice indicated by vertex shape.
v.shapes <- c("circle", "square")[V(lazega)$Practice]
# Vertex size proportional to years with firm.
v.size <- 3.5*sqrt(V(lazega)$Years)
# Label vertices according to seniority.
v.label <- V(lazega)$Seniority
# Reproducible layout.
set.seed(42)
l <- layout.fruchterman.reingold(lazega)
plot(lazega, layout=l, vertex.color=v.colors,
     vertex.shape=v.shapes, vertex.size=v.size,
     vertex.label=v.label)
```



```
vcount(lazega) #36 people
```

```
## [1] 36
```

```
ecount(lazega) #115 collaborations
```

```
## [1] 115
```

We do not have to stop here; we can continue adding layers into this visual in order to display even more information. For example, if the appropriate data was recorded it is possible quantify and qualify the collaborations between the lawyers. We can apply weights or counts to give an idea of how frequently any pair of lawyers interacted with each other in this resource exchange; we can even apply line types (solid, dashed, double-dashed, etc.) to indicate how this collaboration took place: was it done in person, through the phone, through email, or some combination of the three?

Modeling Real-World Networks

One topic of study that has been gaining ground is using graph models to model observed networks. In other words, now that we have observed social networks such as the karate club and law firm, which graph model can generate networks that are visually and structurally similar to a network that we observed? We can do this through a simulation study. That is, we generate many different networks from different graph models and compare the descriptive statistics between the simulated graphs and the observed graphs.

Being able to model observed networks has many application across many fields. By accurately predicting how networks form, it is possible to understand how friendships across people form or even how information spreads across people. More generally, we can even improve traffic flow, provide better product suggestions to online shoppers, and even present the spread of diseases.

What is a Model?

In a way, a *graph model* takes in fixed parameters that generates a graph that can vary in structure with each iteration. Equivalently, it is also possible to consider a model for a network graph as a collection, or *ensemble*, $\{\mathbb{P}_\theta(G), G \in \mathcal{G} : \theta \in \Theta\}$ in which G is a collection or ('ensemble') of possible graphs, P_θ is a probability distribution on G, and θ is a vector of parameters, ranging over possible parameters in Θ.

Descriptive Statistics

Descriptive statistics for networks are analogous to those seen in elementary statistics. These characterize networks that we observe and generate from graph models. We describe a look at a few below. Because we are currently only interested in simulating for accuracy, we omit the mathematical derivations of certain statistics and rely solely on R and the `igraph` package to handle the calculations. For more information, Kolaczyk (2004) and Newman (2010).

Transitivity The *transitivity*, otherwise known as the *clustering coefficient*, provides a sense of how well-connected the graph is. A higher clustering coefficient implies a graph having many nodes close to each other with links connecting all of them.

Average Path Length As mentioned earlier, *average path length* is the average of the shortest paths of all distinct pairs of nodes in the network.

Diameter As mentioned earlier as well, the *diameter* of a graph is the longest of all shortest paths between distinct pairs of nodes.

Centrality *Centrality* provides a measure of the "importance" to each node in the network. It is similar to the measures of central tendency seen in elementary statistics. Many different types of centrality have been proposed, some of which are discussed below. Here, we assume G is undirected.

Vertex degree is perhaps the most common type of centrality. One of the properties of networks is that the degree of the nodes of a graph follow a *power law distribution*. This property indicates that a few nodes with particularly high degrees occur very frequently (compared to if it followed a Poisson distribution, for example). Because of this, nodes with higher vertex degrees are considered to be more central to the network than nodes with lower vertex degrees.

Closeness centrality measures how close a vertex is to other vertices by based on the inverse of the total distance of the vertex from all others.

$$c_{Cl}(i) = \frac{1}{\sum_{i \in V} dist(i,j)}$$

where $dist(i,)$ is the geodesic distance between the vertices $i, j \in V$. To compare across graphs and with other centrality measures, this measure is often normalized to lie in the interval $[0,1]$ by multiplying by a factor of $N_V - 1$.

Betweenness centrality: measures are aimed at summarizing the extent to which a vertex is located between other pairs of vertices. The most commonly used version of betweenness centrality is

$$c_B(i) = \sum_{g \neq h \neq i \in V} \frac{\sigma(g,h|i)}{\sigma(g,h)}$$

where $\sigma(g,h|i)$ is the total number of shortest paths between g and h that pass through i, and $\sigma(g,h) = \sum_V \sigma(g,h|i)$. If the shortest paths are unique, $cB(i)$ just counts the number of shortest paths going through i. This normalized by dividing by a factor of $\frac{(N_V-1)(N_V-2)}{2}$

Eigenvector centrality is based on the idea of 'status,' 'prestige,' or 'rank;' the more central the neighbors of a vertex are, the more central that vertex itself is. One definition of such a centrality measure is:

$$c_{Ei}(i) = \alpha \sum_{\{i,j\} \in E} c_{Ei}(u)$$

.

The vector $c_{Ei} = (c_{Ei}(1), ..., c_{Ei}(N_V))^T$ is the solution to the eigenvalue problem $\mathbf{A}\mathbf{c}_{Ei} = \alpha^{-1}\mathbf{c}_{Ei}$, where $\mathbf{A}$ is the adjacency matrix for network graph G.

Classical Random Graph Models

The following graph models are called "classical" because they have been extensively studied. Derivations will be omitted here. See Newman (2010) for more details.

Here, we will look at the two examples seen earlier–Zachary's karate club and Lazega's lawyers–and consider three network statistics–average path length, diameter, and clustering coefficient. By generating random graph from their respective graph models and compare these descriptive statistics, we can determine how accurate each graph model is.

```r
average.path.length(karate) #2.4082
```

```
## [1] 2.4082
```

```r
diameter(karate) #13
```

```
## [1] 13
```

```r
transitivity(karate, type="global") #0.2556818
```

```
## [1] 0.2556818
```

For the karate club network, the average path length is 2.4 people, the diameter is 13 people, and the clustering coefficient is 0.256.

```r
average.path.length(lazega) #2.144385
```

```
## [1] 2.144385
```

```r
diameter(lazega) #5
```

```
## [1] 5
```

```r
transitivity(lazega, type="global") #0.3887689
```

```
## [1] 0.3887689
```

For the lawyer network, the average path length is 2.1 people, the diameter is 5 people, and the clustering coefficient is 0.389.

Erdos-Renyi-Gilbert

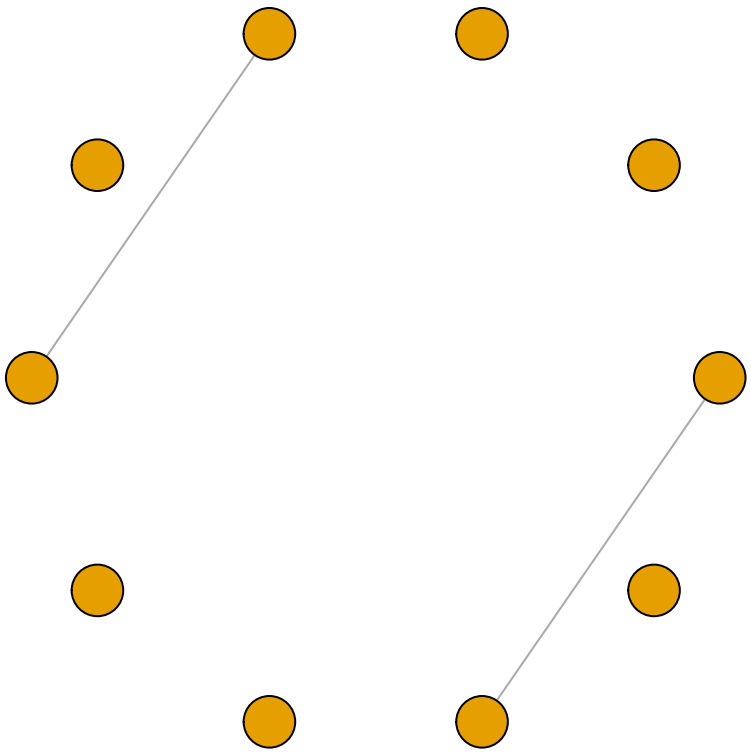
The Erdos-Renyi-Gilbert model is the most studied of the graph models. It is also one of the simplest, taking only two parameters. The model $G(N_V, N_E)$, first suggested by Gilbert, takes in N_V, the number nodes, and N_E the number of edges. Erdos and Renyi considered the model of the form $G(N_V, p)$, where instead of using the number of edges, the probability of an edge forming between any pairs of nodes is fixed. Our focus is on the latter model.

Below is an example of a network constructed from the $G(N_V, p)$ model, in which the number of nodes are fixed at 10, and the change of an edge forming between pairs of nodes is 0.02. Should the model generate another network, chances are the result would look very different. This is why is is also possible to think of a graph model as a probability distribution over an ensemble of networks, and all networks in the ensemble have equal probability of being chosen. More often than not, the graph generates a *giant component*, in which a component forms that is much larger than the rest.

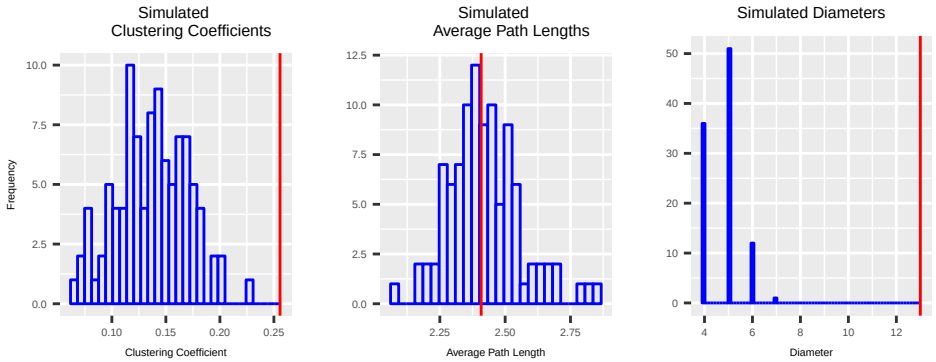
```r
set.seed(42)

# N = 10, P = 0.02
g.er <- erdos.renyi.game(10, 0.02)

plot(g.er, layout=layout.circle, vertex.label=NA)
```

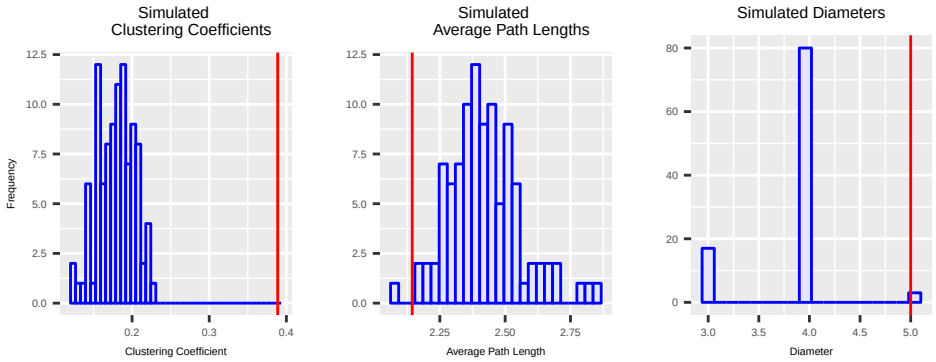


Modeling Zachary's Karate Club with Erdos-Renyi Knowing that the karate network has 34 people and 78 friendships, we simulate 100 networks of parable magnitude. The N_V parameter is already given as 48. To find the probability p we take the quotient of the number of edges observed and the number of edges possible; the former is 78, the latter is $\binom{34}{2}$. We can generate histograms that summarize the network statistics that we choose to explore and determine how well the model did compared to the statistics seen in the observed network.



The histograms above show that the model does not do a good job modeling the clustering coefficient and the diameter aspects of the karate network; both observed values are much greater than those of the networks simulated. However, the network does an excellent job of modeling the average path length; the observed value sits at the center of the distribution of the simulated average path lengths.

Modeling Lazega's Lawyers with Erdos-Renyi The network of lawyers consists of 36 people and 115 collaborations. Again, N_V is given as 36 while p is $\frac{115}{\binom{36}{2}}$.

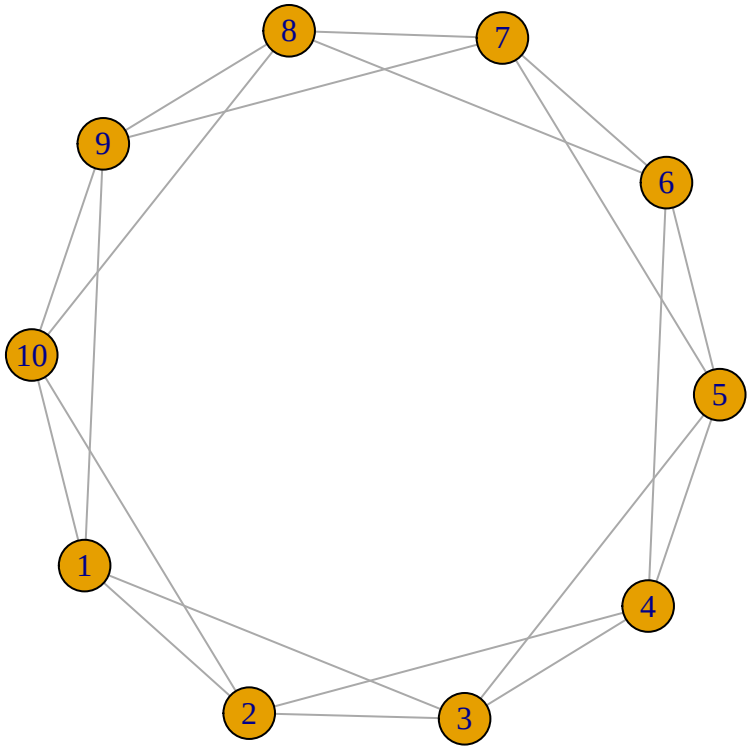


Here, all the Erdos-Renyi model does not accurately model the lawyers network in all three aspects. Simulated graphs underestimate the clustering coefficients and diameter, just like in the karate network, but they also overestimate the average path length.

Watts-Strogatz

Watts and Strogatz noticed that many networks have high levels of clustering and only require small distances between mode nodes. As seen in the Erdos-Renyi model, simulated graphs of parable magnitude tend to have smaller than expected clustering coefficients. In the Watts-Strogatz model, we start with N_V number of vertices arranged in a circular fashion, r number of beginning neighbors for each node, and the probability p of a node being moved to another pair of vertices.

```
# lattice
# dimenesion = 1, size = 10, neighborhood = 2, p = 0
g.lat10 <- watts.strogatz.game(1, 10, 2, 0)
plot(g.lat10)
```



In the plot above, we have 10 nodes, each starting with 2 neighbors.

```
# substantial clustering
transitivity(g.lat10, type="global")
```

```
## [1] 0.5
```

```
# nontrivial diameter and path lengths
diameter(g.lat10)
```

```
## [1] 3
```

```
average.path.length(g.lat10)
```

```
## [1] 1.666667
```

Because every node is connected is every other link in a similar way, the clustering coefficient is relatively high. Clustering and average path length are nontrivial, and can be rather high as well.

```
# rewire to add shortcuts
# dimenesion = 1, N = 10, neighborhood = 2, p = 0
g.ws10 <- watts.strogatz.game(1, 10, 2, 0.05)

# increased clustering
transitivity(g.ws10, type="global")
```

```
## [1] 0.4354839
```

```
# reduced diameter and path lengths
diameter(g.ws10)
```

```
## [1] 3
```

```
average.path.length(g.ws10)
```

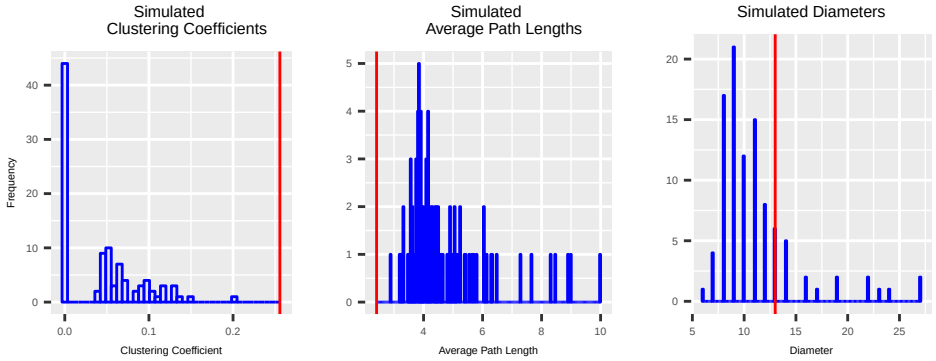
```
## [1] 1.622222
```

By adding a probability of rewiring, the clustering coefficient increases, while diameter and average path length decreases. This makes sense because this rewiring essentially creates shortcuts in the network framework.

Modeling Zachary's Karate Club with Watts-Strogatz We start with $N_V = 34$ nodes again, as originally observed in the karate network. Because we will be rewiring edges, one can argue that the beginning number of neighbors should be so that the total number of edges is approximately equal to the total number of edges observed. This is approximately $r = 2$. For the probability of rewiring, because there is no justifiable probability, we will simply set the probability equal to the uniform probability distribution on the unit interval.

```
g.ws.karmod.coef <- rep(NA, numsim)
g.ws.karmod.apl <- rep(NA, numsim)
g.ws.karmod.dia <- rep(NA, numsim)

for (i in 1:numsim) {
  g.ws.karmod <- watts.strogatz.game(dim = 1,
                                     size = vcount(karate),
                                     nei = 0,
                                     p = runif(1, min=0, max=1))
  g.ws.karmod.coef[i] <- transitivity(g.ws.karmod, type="global")
  g.ws.karmod.apl[i] <- average.path.length(g.ws.karmod)
  g.ws.karmod.dia[i] <- diameter(g.ws.karmod)
}
```

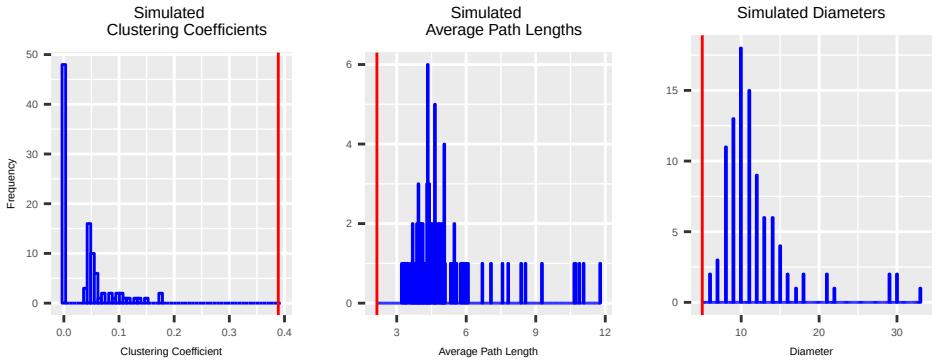


The clustering coefficients of the simulated graphs are quite random on the unit interval; the Watts-Strogatz model does not do a good job of modeling the clustering coefficient of the karate network. Similarly, the simulated networks of this model tend to underestimate the average path length and diameters of the karate network.

Modeling Lazega's Lawyers with Watts-Strogatz We have $N_V = 36$, and $N_E = 115$. So we can start with 36 nodes and $r = 3$ neighbors. We again use a uniform probability distribution on the unit interval for p.

```
g.ws.lawmod.coef <- rep(NA, numsim)
g.ws.lawmod.apl <- rep(NA, numsim)
g.ws.lawmod.dia <- rep(NA, numsim)

for (i in 1:numsim) {
  g.ws.lawmod <- watts.strogatz.game(dim = 1,
                                     size = vcount(lazega),
                                     nei = 0,
                                     p = runif(1, min=0, max=1))
  g.ws.lawmod.coef[i] <- transitivity(g.ws.lawmod, type="global")
  g.ws.lawmod.apl[i] <- average.path.length(g.ws.lawmod)
  g.ws.lawmod.dia[i] <- diameter(g.ws.lawmod)
}
```



Here, the simulated graphs from the Watts-Strogatz model underestimates the clustering coefficient of the lawyer network and over estimated the average path length and diameter. Thus, this model does not do a good job overall of modeling the lawyer network.

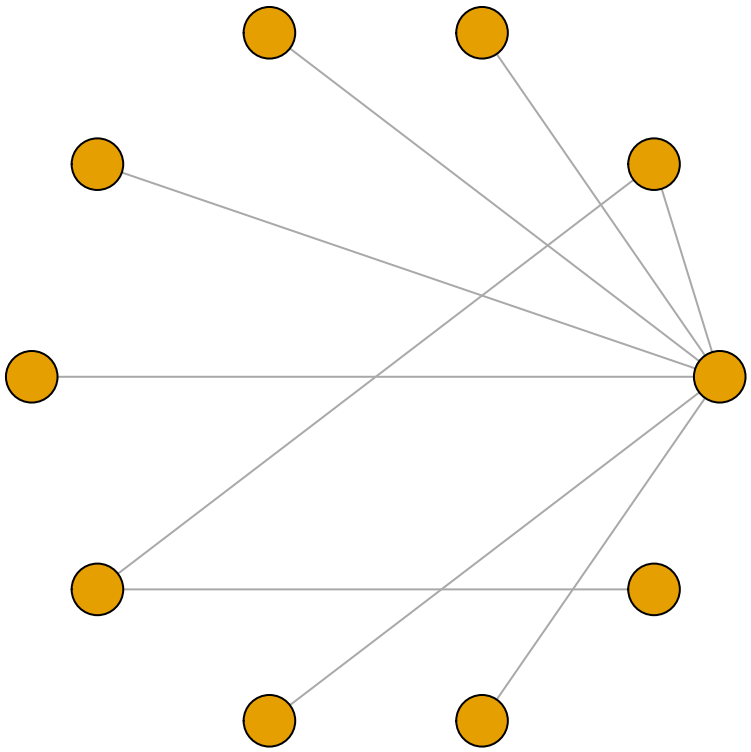
Babarasi-Albert

Babarasi and Albert were interested in how networks grow and change over time. In this model, we start with an initial graph $G^{(0)}$ of $N_V^{(0)}$ vertices and $N_E^{(0)}$ edges. At each stage t = 1, 2, ..., the current graph that is $G^{(t-1)}$ is modified to create a new graph $G^{(t)}$. A new vertex of degree $m >= 1$ is added to m different vertices with probability such that it favors vertices with higher degrees.

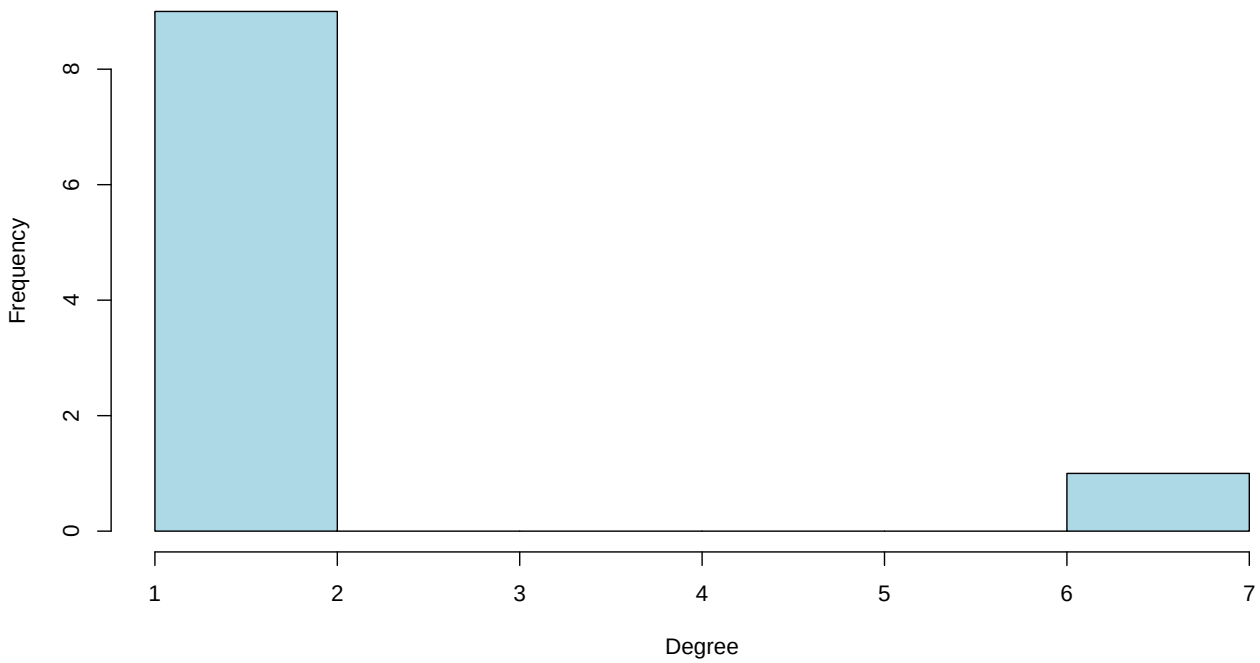
In the graph below, a link is added to the network for each node added to the graph.

```
# N = 10 vertices, m = 1 new edge added for each new vertex
g.ba <- barabasi.game(10, directed=FALSE)

plot(g.ba, layout=layout.circle, vertex.label=NA)
```



```r
# some nodes with very high degree
hist(degree(g.ba), col="lightblue", xlab="Degree", ylab="Frequency", main="")
```




```r
summary(degree(g.ba))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00    1.00    1.00    1.80    1.75    7.00
```

```r
average.path.length(g.ba)
```

```
## [1] 2.222222
```

```r
diameter(g.ba)
```

```
## [1] 4
```

```r
transitivity(g.ba, type="global")
```

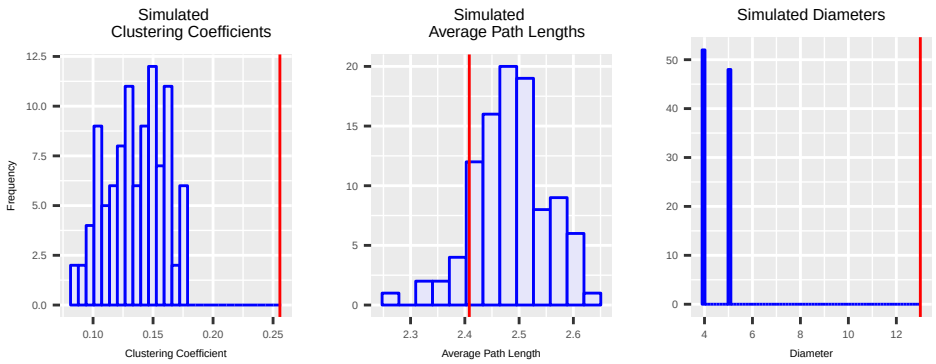
```
## [1] 0
```

As the histogram above shows, this model outputs some nodes with high degree.

Modeling Zachary's Karate Club with Babarasi-Albert We start with $N_V = 34$ nodes yet again, and with 78 edges observed, we can create a similar network by adding two links with every node added.

```
g.ba.karmod.coef <- rep(NA, numsim)
g.ba.karmod.apl <- rep(NA, numsim)
g.ba.karmod.dia <- rep(NA, numsim)

for (i in 1:numsim) {
  g.ba.karmod <- barabasi.game(n = vcount(karate), m = 2, directed=FALSE)
  g.ba.karmod.coef[i] <- transitivity(g.ba.karmod, type="global")
  g.ba.karmod.apl[i] <- average.path.length(g.ba.karmod)
  g.ba.karmod.dia[i] <- diameter(g.ba.karmod)
}
```

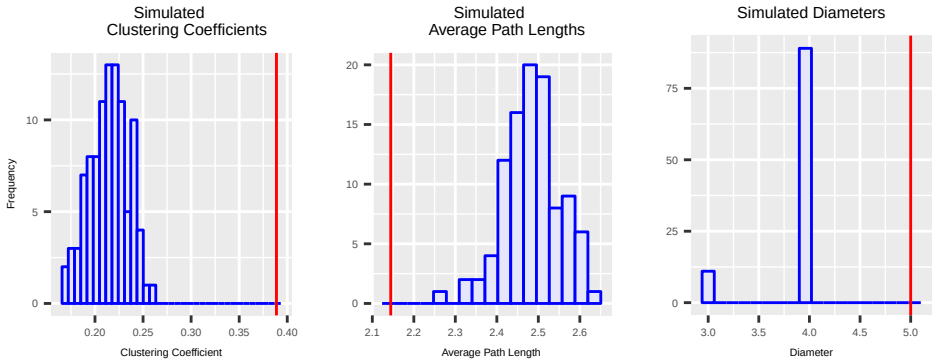


The simulated graphs from the Barabasi-Albert model do not seem to do a good job with the karate network as well. While the clustering coefficient and diameter were underestimated, the average path length was slightly overestimated.

Modeling Lazega's Lawyers with Babarasi-Albert With $N_V = 36$ lawyers and $N_E = 115$ collaboration, we can generate graphs with 36 nodes, each time, adding 3 links.

```
g.ba.lawmod.coef <- rep(NA, numsim)
g.ba.lawmod.apl <- rep(NA, numsim)
g.ba.lawmod.dia <- rep(NA, numsim)

for (i in 1:numsim) {
  g.ba.lawmod <- barabasi.game(n = vcount(lazega), m = 3, directed=FALSE)
  g.ba.lawmod.coef[i] <- transitivity(g.ba.lawmod, type="global")
  g.ba.lawmod.apl[i] <- average.path.length(g.ba.lawmod)
  g.ba.lawmod.dia[i] <- diameter(g.ba.lawmod)
}
```



The simulated graphs from the Barabasi-Albert model do not seem to do a good job with the lawyer network as well. While the clustering coefficient and diameter were underestimated, the average path length was highly overestimated.

Further Work

So far, we have gone through a brief overview of networks, including some basic terminology and some network statistics. We also introduced three heavily-studied graph models and compared the accuracy of how well they model observed networks. This was done by simulating networks that were similar to that of the observed network and compared different network statistics to that of the observed network. In the end, it seems that none of these models were able to generate statistics that were close to that of the original model. We can see if these three models would be accurate in predicting other network statistics (e.g. centrality measures). It is also possible (and perhaps more reasonable) to explore other graph models that can model the observed networks. There area myriad of graph models to choose from, ranging from exponential random gram models, which also have been extensively studied for some time, to the R-MAT generator, a network that has only been proposed in the last decade. See Chakrabarti (2006) for more details. Accurately modeling observed networks has many applications, the challenge of doing so comes from the variety of networks seen in the world, each with vastly different properties. By accurately modeling social networks, we can better understand how information and diseases travel among people or also how friendships form and change over time.

References

Chakrabarti, Deepayan, and Christos Faloutsos. “Graph mining: Laws, generators, and algorithms.” ACM computing surveys (CSUR) 38.1 (2006): 2.

Kolaczyk, Eric D. Statistical Analysis of Network Data: Methods and Models. New York: Springer, 2009. Print.

Kolaczyk, Eric D., and Gabor Csardi. Statistical Analysis of Network Data with R. New York: Springer, 2014. Print.

Newman, M. E. J. Networks: An Introduction. Oxford: Oxford UP, 2010. Print.