# SF5: Networks, friendship, and disease

Project leader: Dr George Cantwell

April 25, 2024

## Aims

The aims of the course are:

- To understand and apply concepts from the study of networks.
- Familiarity with the basics of discrete maths and graph theory.
- To run simulations of social networks and to link ideas from probability theory to these simulations.
- To implement graph algorithms and appreciate the need for computational efficiency.

## Demonstrator sessions

The following sessions are mandatory:

- Friday 10th May, 11.00-13.00, JDB Teaching room
- Tuesday 14th May, 9.00-11.00, JDB Teaching room
- Tuesday 21st May, 9.00-11.00, TBC
- Tuesday 28th May, 9.00-11.00, JDB Teaching room

In addition, I will run "office hours" in BE3-11 on Tuesdays from 4pm-5pm (dates: 14/5, 21/5, 28/5, 4/6). You are welcome to come to my office at these times without an appointment to discuss project work.

## Reports

You will be graded based on three reports.

- **Interim report 1, 17/5/24 (10 marks).** Summary of assignments from Wk 1. Two sides. Submit via moodle.
- **Interim report 2, 24/5/24 (10 marks).** Summary of assignments from Wk 2. Two sides. Submit via moodle.

- **Final report, 7/6/24 (50 marks).** A final report covering all relevant material. Ten sides. Submit via moodle.

The interim reports can be brief and assume advanced technical knowledge of the reader. The final report must be self-contained and explain things clearly. It should possible for another student (not in this class) to read and understand the material. There is considerable scope to explore particular topics you enjoyed in more detail, and both initiative and originality will enter into your final mark.

## 1 Week 1: Networks

As scientists, when we look at some complicated system that we don't understand, a reasonable instinct is to conceptually break the system apart into constituent pieces. For example, brains appear to be hopelessly complicated. On the other hand, neurons – the most important cells in the brain – appear to be relatively simple. If we could only understand neurons better, then perhaps we could understand the brain. This kind of approach is known as *reductionism*, and it has been enormously fruitful for answering certain kinds of questions.

However, while it is interesting to break things apart, it is not at all obvious how to put the pieces back together again. Brains think; individual neurons don't. Something different occurs when lots of neurons are wired together in just the right way – something no individual neuron does on its own. The field of *complex systems* or *complexity science* is the attempt to understand how interesting behaviors emerge from the combination of many relatively simple pieces.

In this quest, one of the most useful conceptual tools is the idea of a network. Our brain is a network of neurons – a neural network. Society is a network of people – a social network. The internet is a network of computers – a computer network. Thus, networks provide a common language to describe many different systems.

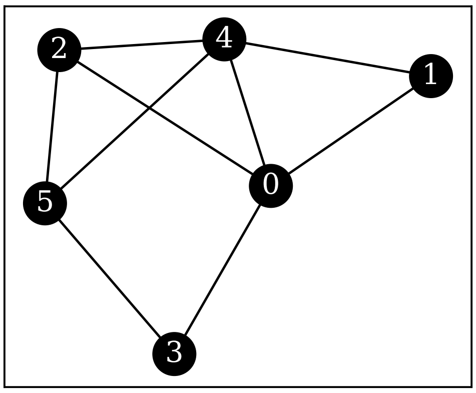

Figure 1: An example graph with 6 nodes, labeled from 0 to 5.

The mathematical concept at the core of network science is what is known as a *graph*. Unlike its standard use in English, the word graph means something quite different to a mathematician. A graph is a discrete structure, made up of nodes and edges between them. For example, in Fig. 1 we have a graph with 6 nodes and 9 edges. We will use both the word "graph" and the word "network" for this kind structure.

A network can be represented in several different ways. One is the *adjacency matrix*, denoted $A$ for adjacency, and with entries defined

$$A_{ij} = \begin{cases} 1 & \text{if edge between nodes } i \text{ and } j \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

The adjacency matrix for the network in Fig. 1 is

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix} \tag{2}$$

Note, the adjacency matrix is binary valued and symmetric. A similar method to represent the same thing is the adjacency list, in which we list each node followed by the nodes they are connected to. In this case,

$$\begin{aligned} 0 &: \{1, 2, 3, 4\} \\ 1 &: \{0, 4\} \\ 2 &: \{0, 4, 5\} \\ 3 &: \{0, 5\} \\ 4 &: \{0, 1, 2, 5\} \\ 5 &: \{2, 3, 4\}\,. \end{aligned}$$

In any network, some nodes may be more important than others. One measure of this importance is the **degree** of a node. The degree of a node is simply a count of how many connections that node has. In a network of friendships, your degree would be your total number of friends. We denote the degree of node $i$ as $k_i$, and it can be found from the adjacency matrix

$$k_i = \sum_j A_{ij}. \tag{3}$$

Nodes with a lot of edges might be considered to be more critical to the functioning of the network. Of course, there are more sophisticated measures of importance too. For example, the PageRank is another quantity to measure the importance of nodes. PageRank was one of the key innovations that made Google's search algorithms successful – it ranked webpages by measuring their importance in a network of webpages and hyperlinks.

## 1.1 The random graph: A model for networks

To begin our studies, we start with perhaps the simplest model for a network – the random graph. In the random graph $G(n, p)$ we have $n$ nodes, each pair of which is connected with probability $p$.

In the notation of the adjacency matrix, we say

$$A_{ij} \sim \text{Bernoulli}(p) \tag{4}$$

where the notation Bernoulli($p$) indicates a random variable that takes the value 1 with probability $p$, and the value 0 with probability $1 - p$. Note, we are considering undirected graphs so $A_{ij} = A_{ji}$. This means that we must only consider adding an edge for each pair once.

While the random graph is a remarkably simple model, there are still many open problems concerning its properties despite decades of intense study.

## 1.2 Network data structures

Many networks have many more zeros than ones in the adjacency matrix. In this case, adjacency lists can be more efficient. A simple Python implementation is as follows.

```python
class Network(object):
    def __init__(self, num_nodes):
        self.adj = {i:set() for i in range(num_nodes)}

    def add_edge(self, i, j):
        self.adj[i].add(j)
        self.adj[j].add(i)

    def neighbors(self, i):
        return self.adj[i]

    def edge_list(self):
        return [(i,j) for i in self.adj for j in self.adj[i] if i<j]
```

Feel free to amend this implementation, make use of `numpy arrays`, or any other data structure, as you deem fit. When dealing with large simulations, it is important to think about efficiency.

## 1.3 Assignments

1. Write a function in Python to sample a network from $G(n, p)$. Each time you run the function, it should output the adjacency matrix of a different network. The total number of edges, $m$, will thus be a random variable. Plot a histogram of $m$ for a large number of runs with $n = 100$ and various values of $p \in (0, 1)$.

2. There are $\binom{n}{2} = \frac{1}{2}n(n-1)$ possible pairs of nodes, each of which will have an edge with probability $p$. Thus, the probability for there being $m$ edges will follow the binomial distribution,
$$P(m) = \binom{\binom{n}{2}}{m} p^m (1-p)^{\binom{n}{2}-m}. \tag{5}$$
Compare this theoretical distribution to your histogram. What is the expected value of $m$? What is the variance?

3. There are $(n-1)$ other nodes that any given node could possibly be connected to. Each connections exists with probability $p$. Derive an expression for the probability $P(k)$ that a node has exactly $k$ edges. This expression will depend on both $n$ and $p$. What is the expected value of the degree $k$? The distribution for degrees is known as the *degree distribution*.

4. Consider $G(n, \lambda/(n-1))$ for some constant $\lambda$. That is, suppose that we set $p = \lambda/(n-1)$ where $\lambda$ is some constant. This means that as we increase $n$, we decrease $p$. For this case, what is the expected degree? How does it depend on $n$? What about the variance? Using your expression for the degree distribution from before, derive an expression for the probability of having degree $k$, $P(k)$, in the limit that $n \to \infty$ with $p = \lambda/(n-1)$.

5. The naive sampling algorithm for $G(n,p)$ loops through all $\frac{1}{2}n(n-1)$ pairs of nodes. This may be inefficient. We can improve it with a two step algorithm. First, we chose a value $m$ to denote how many edges our network will have. To be consistent with $G(n,p)$, we must pick $m$ according to Eq. (5). Next, given that our network will have $m$ edges, we add these uniformly at random to the network. Write code to implement this two-stage algorithm.

6. You should now have two algorithms to sample $G(n,p)$, one from Q1 and one from Q4. To benchmark the algorithms, estimate the run time for each one to generate from $G(n, 10/(n-1))$, with $n = 64$, $n = 128 \ldots, n = 1024$. You can measure the time a piece of code takes to run in python using the `timeit` module, e.g. by running

   ```
   timeit.timeit(lambda: sample_gnp(64, 10/63), number=100)
   ```

   The *time complexity* of an algorithm is a description of how long that algorithm takes to run as a function of $n$. For large $n$, a linear time algorithm, denoted $\mathcal{O}(n)$, will be preferable to say, a quadratic time algorithm, denoted $\mathcal{O}(n^2)$. Plot the average run time for your $G(n, 10/(n-1))$ algorithms against $n$ on a log-log plot. One should be approximately quadratic (slope of 2), the other linear (slope of 1).

7. Even if two nodes are not directly connected, there might be a longer path between them. Any two nodes that are connected by a path of any length are said to be in

the same component. However, in general a network will be fragmented into multiple components.

Write code to efficiently find all the nodes that can be reached from node 1. For $n = 4096$, as $p$ varies from 0 to 0.001, plot how the average size of the component that contains node 1 changes. At $p \simeq 1/(n-1)$, we see a change from small components (approximately 0% of the total network) to much larger ones.

## 2 Week 2: Friendship paradox

Whenever nodes have a numerical property, we can calculate the average value of the property within the network. We will denote network averages using the notation $\langle \dots \rangle$, for example if each node $i$ has an associated property $x_i$, then

$$\langle x \rangle = \frac{1}{n} \sum_i x_i. \tag{6}$$

Note, this is generally *not* the same as an expected value in our model – this is an empirical average over the nodes of a network.

In a network of friendships, the degree of a node corresponds to the number of friends that person has. Thus, the quantity $\langle k \rangle$ is the average number of friends that people in the network have,

$$\langle k \rangle = \frac{1}{n} \sum_i k_i \tag{7}$$

$$= \frac{1}{n} \sum_{i,j} A_{ij}. \tag{8}$$

Of course, some people may be very popular indeed, with many more friends than average. Likewise, some people may be much less popular, with only a very small number of friends. Most of us, however, can expect to be in-betweeners. If you are averagely popular, then you would expect to have roughly $\langle k \rangle$ friends.

Do you ever feel as though you are less popular than your friends? Well it turns out that on average, your friends are more popular than average! This claim sounds counter-intuitive, almost paradoxical. It is a result that is known as the *friendship paradox*, although it is not really a paradox but a simple mathematical fact.

Let's expand on the claim. Node $i$ has $k_i$ friends. Each such friend $j$ has in turn $k_j$ friends. Let $\kappa_i$ denote the average number of friends that node $i$'s friends have. By this definition, we have

$$\kappa_i = \frac{1}{k_i} \sum_j A_{ij} k_j. \tag{9}$$

Just as the degree $k_i$ is a measure of how popular node $i$ is, the value $\kappa_i$ is a measure of how popular node $i$'s friends are. If $k_i$ is less than $\kappa_i$, then node $i$ is less popular than their friends, on average.

So how does $k_i$ actually compare to $\kappa_i$? This may differ from node to node, but we can calculate the average. We have

$$\begin{aligned}\langle\kappa\rangle &= \frac{1}{n}\sum_i \kappa_i &(10)\\ &= \frac{1}{n}\sum_{i,j} A_{ij}\frac{k_j}{k_i}. &(11)\end{aligned}$$

Now, because we have summed over the indices $i$ and $j$, we are free to reverse them, to get

$$\langle\kappa\rangle = \frac{1}{n}\sum_{i,j} A_{ji}\frac{k_i}{k_j}. \qquad (12)$$

If we add Eq. (11) to Eq. (12) and divide by 2 we get

$$\begin{aligned}\langle\kappa\rangle &= \frac{1}{2}\left(\langle\kappa\rangle + \langle\kappa\rangle\right) &(13)\\ &= \frac{1}{2}\left(\frac{1}{n}\sum_{i,j} A_{ji}\frac{k_i}{k_j} + \frac{1}{n}\sum_{i,j} A_{ji}\frac{k_i}{k_j}\right) &(14)\\ &= \frac{1}{n}\sum_{i,j} A_{ij}\,\frac{1}{2}\left(\frac{k_i}{k_j} + \frac{k_j}{k_i}\right). &(15)\end{aligned}$$

**Exercise.** Prove that

$$\frac{1}{2}\left(x + x^{-1}\right) \geq 1 \qquad (16)$$

for all $x > 0$. Hint: multiply by $x$ and factorize.

Using the inequality in Eq. (16), but with $x = k_i/k_j$, we finally have

$$\begin{aligned}\langle\kappa\rangle &= \frac{1}{n}\sum_{i,j} A_{ij}\,\frac{1}{2}\left(\frac{k_i}{k_j} + \frac{k_j}{k_i}\right) &(17)\\ &\geq \frac{1}{n}\sum_{i,j} A_{ij} = \langle k\rangle. &(18)\end{aligned}$$

Because we made no assumption about the structure of the network, the result that $\langle\kappa\rangle \geq \langle k\rangle$ is true for all networks. Or in words: No matter the structure of your friendship network, people, on average, have fewer friends than their friends do. So, if you are ever feeling less popular than your friends then take comfort in the fact that you quite probably are, but this is to be expected.

## 2.1 The configuration model

In the $G(n,p)$ random graph model, every node has roughly the same degree. This is unrealistic. The *configuration model* is a method to generate networks that (approximately) follow a given degree distribution.

Our goal is to create a network in which each node $i$ has $k_i$ edges, for an arbitrary sequence $k_i$. In order to do this, we create a list `S` that contains each $i$ a total of $k_i$ times. I.e., if $k_3 = 6$ then the list should contain 6 copies of the value 3. Then, randomly shuffle the entries of `S`, and reshape it into an $m \times 2$ matrix (dropping the last entry if the list length is odd).

```
S = np.array([i for i in range(n) for _ in range(k[i])])
S = np.random.permutation(S)
if len(S) % 2:
    S = S[:-1]
S = S.reshape(-1, 2)
```

This matrix $S$ is now our list of edges.

The reason this algorithm works is that each node $i$ will be contained in the edge list $k_i$ times.[1] And, if node $i$ occurs in $k_i$ different edges, then that means that node $i$ has degree $k_i$, as required.

The caveat, however, is that this algorithm can create duplicate edges and self-loops (edges between a node and itself). For the purpose of this project, we will simply neglect these cases – whenever you see a duplicate or self edge, ignore it and move on.

## 2.2 Assignments

1. Write code to sample from the configuration model where the degrees $k_i$ are sampled from a Poisson distribution.

   Defining the geometric distribution, $p_k = p(1-p)^k$ for $k \geq 0$, write code to sample a configuration model with a geometric degree distribution (be careful, `np.random.geometric` defines the distribution differently).

   For both cases, sample a network with $10,000$ nodes and mean degree 10. Verify that the degree distributions approximately match the Poisson/geometric distribution probability mass functions.

2. Suppose our graph has degree distribution $p_k$. This means that a randomly chosen node will have degree $k$ with probability $p_k$. However, a randomly chosen friend will follow a different distribution. First, verify the friendship paradox numerically.

   To do this, let's reuse the two networks from above with $10,000$ nodes, mean degree 10, and a Poisson/geometric degree distribution. For these networks, repeat the following: chose a node at random, then chose a neighbor of that node at random, and then record

[1] One node may appear $(k_i - 1)$ times, if it was the very last item that got dropped in order to make `S` an even length. For a large network, this should lead to a negligible effect on the degree distribution.

both of their degrees. After you do this many times you can make a histogram of the degrees. (Note: If the first node has degree 0, then it has no friends, so ignore it and try again). Compare the two distributions of degrees – do you observe the "friendship paradox"?

3. Let $\Delta_i = \kappa_i - k_i$. This is the amount by which node $i$'s friends are more popular than itself, on average. Make histogram for $\Delta_i$, and show that the mean is larger than 0. (Again, think carefully about how to deal with degree 0 nodes.)

4. Let $q_k$ be the probability distribution for a random chosen friend to have degree $k$. Argue that for the configuration model,

   $$q_k = \frac{k p_k}{\sum_{k'} k' p_{k'}}. \tag{19}$$

   Verify this formula for for the Poisson and geometric degree distributions.

   Consider the difference between the expected degree of a friend, $\sum_k k q_k$, and the expected degree of a randomly chosen node, $\sum_k k p_k$. Show how this quantity is determined by the variance of the degree distribution. What does this look like for the Poisson and geometric distributions?

5. The generating function for distribution $p_k$ is defined

   $$G(z) = \sum_k z^k p_k. \tag{20}$$

   Show that the generating function for $q_k$ is

   $$\sum_k z^k q_k = \frac{z\, G'(z)}{G'(1)} \tag{21}$$

   where $G'(z) = \frac{\mathrm{d}G(z)}{\mathrm{d}z}$. Derive explicit expression for these for both Poisson and geometric distribution. Relate these to the friendship paradox.

6. Last week, we looked at the size of the component that contained a given node. To find the nodes in the component, we first start at the node and then add all of its neighbors. There will be $k$ immediate neighbors with probability $p_k$. After we add the neighbors, we look at their neighbors. Each neighbor will have $k-1$ extra neighbors with probability $q_k$ (we lose one because it is the original node). This process then repeats until there are no more nodes left to explore. If $\sum_k k q_k > 1$ then we expect the process to keep growing for several rounds.

   Generate Poisson/geometric graphs with mean degrees between 0 and 2. Measure the size of the component containing an arbitrary node, and show that this starts to grow earlier for the geometric degree distribution than the Poisson distribution. Compare simulation results to analytic predictions.

The presence of nodes with unusually high degree means that most people will be considerably less popular than their friends. However, these high degree nodes also help to hold the network together.

7. **(Challenge)** So far, the analyses have all assumed that the degrees of neighbors are uncorrelated. In reality, popular people are more likely to be friends with other popular people. How would this effect things? Explore this with simulations and/or analytical arguments.

## 3 Week 3: Spreading processes

Networks provide a framework to describe the structure of social groups. But they allow us to do more than simply *describe* structure – the allow us to model processes.

For example, if a new disease is circulating in a community, we would want to estimate how fast and how far that disease will spread. We can do this by modeling how the disease would spread in the relevant network. There is a very large body of literature on the use of networks to model dynamic processes, and we will study an important example.

Suppose a disease (e.g., influenza) worked as follows: Every time someone gets infected, they will remain infectious for one week and then recover. During this week, they will spread the disease to each of their contacts, independently and with probability $\lambda$. Whenever the infection successfully spreads, the new individual will be infectious for the next week. In turn, this person will infect each of his or her contacts with probability $\lambda$, and so on. However, once someone recovers we will assume they cannot be reinfected.

For this process, we can classify each person at a given time as being either Susceptible/Infectious/Recovered, and so these models are called SIR models. Many of the assumptions are overly simplistic. For example, we assume that a recovered person can never be reinfected. And, we assume that an infectious person is infectious for exactly one time period (e.g., one week). Nevertheless, simplified models still help us to both understand and control processes. This model allows us to see how network structure and the parameter $\lambda$ interact to create typical patterns of spread.

Having defined the evolution, to run a simulation we simply need to define an initial condition. For example, to start the outbreak we could assume that almost all of the individuals are susceptible, but a small fraction of individuals, chosen uniformly at random, begin as infectious.

### 3.1 Assignments

1. Write code to simulate the SIR process described above. Run the process until extinction – until every node is either in state $S$ or $R$, and no one infectious remains. On a large network with an intermediate mean degree (e.g., $n = 10,000$ nodes and $k = 20$ mean degree), and for a few different values of of $\lambda$, plot the values of $S$, $I$, and $R$ against time.

2. Every node that was infected at any time will end up in state $R$. So, once the process has ended, the number of nodes in state $R$ will be equivalent to the total number of cases in the outbreak. Run simulations for a range of values of $\lambda$, and plot $R$ against $\lambda$. You will need to average over many simulations to get a smooth curve.

   Below some threshold value of $\lambda$, the process is "sub-critical", meaning an initial outbreak only infects a very small fraction. Above the threshold, the process is "super-critical". The value of the threshold depends on the specific network. From your simulations, attempt to find the threshold.

3. Let $s_i$ be the probability that node $i$ is never infected. We will now derive an approximation for $s_i$. If node $i$ is never infected, then none of its neighbors $j$ ever infect it.

   Assuming independence, then probability that node $j$ becomes infected and passes the infection to $i$ is $(1 - s_j)\lambda$. Hence, the probability that $j$ does *not* infect $i$ is 1 minus this quantity $1 - \lambda + s_j\lambda$. If $i$ does not get infected, then it must not have been infected by any of its neighbors, and hence

   $$s_i = \prod_j (1 - \lambda + s_j\lambda)^{A_{ij}}. \tag{22}$$

   This equation can be solved by iteration – first, fix $s_i$ to random values, and then iterate the equation until it converges. Compare the predictions of this equation to simulations.

   Note that $s_i = 1$ for all $i$ is always a fixed point, though it may be unstable. By taking the log of both sides of Eq. (22), relate the stability of this $\boldsymbol{s} = 1$ fixed point to the eigenvalues of $A$. Relate this to the simulations and the epidemic threshold.

4. If we want to estimate the size of the outbreak, the naive method of simulating the whole process can be improved.

   When we simulate SIR, then whenever a node gets infected we flip a biased coin for each neighboring susceptible node to see if the infection spreads. All of these coin flips could be done in advance.

   Using a *disjoint set* (see scipy documentation), we can make one loop over all edges. First, initialize the set

   ```
   from scipy.cluster.hierarchy import DisjointSet
   C = DisjointSet(range(n))
   ```

   Then, for each edge $i, j$, we merge them with probability $\lambda$ by running `C.merge(i,j)`. At the end, the value of `C.subset_size(i)` is equivalent to the number of nodes that would become infected, if $i$ were the initial infection.

   Run this algorithm to estimate both the mean and the standard deviation of cluster size for the approximate range $\lambda \in (0, 3\lambda_c)$, where $\lambda_c$ is the critical value of $\lambda$. Plot both the cluster size $\mu$ and the coefficient of variation $\sigma/\mu$, against $\lambda$. What happens to $\sigma/\mu$ near $\lambda_c$?

5. **(Challenge)** We can greatly increase the speed of these simulations further. To do this, we loop through every edge $i, j$ in a random order, merge cluster $i$ and $j$ with `C.merge(i,j)`, and record the current sizes of the components. After we have worked through $k$ edges on the list, we will have a correct sample conditioned on getting exactly $k$ edges selected. Call the size of the cluster containing $i$ after $k$ edges, $c_i(k)$. In a single loop through all $m$ edges, we get an estimate of $c_i(k)$ for all $k \in 0, 1, \ldots, m$.

   For a given $\lambda$, the probability of $k$ edges being selected from a total of $m$ in the full process is $\binom{m}{k}\lambda^k(1-\lambda)^{m-k}$. Thus, the size of the cluster containing $i$, if we use parameter $\lambda$ is
$$c_i(\lambda) = \sum_{k=0}^{m} \binom{m}{k} \lambda^k (1-\lambda)^{m-k} c_i(k). \tag{23}$$
   This means that when we measure $c_i(k)$, we can effectively simulate all values of $\lambda$ in one sweep.

   Implement this algorithm and compare the results and the speed to the full simulation algorithm.

## 4 Week 4: Putting it all together

As we have seen, for $\lambda > \lambda_c$ a disease that starts with only a few infectious individuals will spread to a considerable proportion of the whole network. By simulating the process many times, we can estimate relevant quantities. This week, we will look at how the structure of the network interacts with the dynamics of the spreading process.

Let $x_i = 1 - s_i$ be the probability that node $i$ becomes infected at some point. For example, if we run 1000 simulations, and node $i$ becomes infected in 250 of them, then we can estimate $x_i = 0.25$. Just as the degree $k_i$ is a simple measure of the structural centrality of a node, $x_i$ is a simple measure of the dynamical centrality. If $x_i$ is relatively large, then node $i$ is relatively central in the network from the perspective of the disease.

### 4.1 Assignments

1. Through simulation, write code to estimate the vector $x_i$ for given $\lambda$.

2. For a couple of large networks – one with Poisson degree and one with geometric – compare $x_i$ to the degree $k_i$. This is most interesting when $\lambda$ is slightly above its critical value, because if $\lambda$ is too small or too large $x_i$ will be close to zero or one for all $i$.

3. A randomly chosen node will become infected with probability $\langle x \rangle$. However, a randomly chosen neighbor will become infected with probability

$$\frac{1}{n}\sum_{i,j}\frac{A_{ij}x_j}{k_i} = \left\langle \frac{Ax}{k} \right\rangle \tag{24}$$

Show that there is a "disease paradox": your friends are more likely to have more diseases than you do! Combine simulation and analytic arguments as you see fit.

Is the issue more or less pronounced on networks with larger variance in the degree distribution?

4. It is important to vaccinate people to avoid large outbreaks. In this final part, we explore herd immunity.

First, suppose you randomly vaccinate 40% of the population. To simulate this, you can simply remove those nodes from the network. Then, run the simulations for a range of values of $\lambda$ and plot the outbreak size against $\lambda$. Repeat and compare this for 0% and 20% vaccination rate. For some values of $\lambda$, the disease spreads in an unvaccinated population, but does not when enough people are vaccinated.

For any $\lambda$, there is some value of the vaccination rate that will make the disease subcritical. While it is usually impractical or impossible to vaccinate everyone, everyone benefits when enough people are vaccinated.

Next, we will will use a different protocol to vaccinate. As before, we will randomly pick nodes. However, this time we will not vaccinate those nodes that we picked. Instead, we will let them nominate one friend to be vaccinated. After 40% of the network is vaccinated, stop vaccinating and run the simulation. How does this compare to the previous protocol? How is this related to the friendship/disease paradox?

5. **(Challenge)** In an unvaccinated population, we argued that the probability of not getting infected approximately obeyed $s_i = \prod_j (1 - \lambda + s_j\lambda)^{A_{ij}}$. Derive appropriate generalizations of this for the two vaccination strategies above. Test the approximations against your simulations.

## Further Reading

[1] M. E. J. Newman. *Networks.* Oxford University Press, second edition, 2018.

[2] S. N. Dorogovtsev and J. F. F. Mendes. *The Nature of Complex Networks.* Oxford University Press, 2022.

[3] H. J. Jensen. *Complexity Science: The Study of Emergence.* Cambridge University Press, 2023.

[4] G. T. Cantwell, A. Kirkley, and M. E. J. Newman. The friendship paradox in real and model networks. *Journal of Complex Networks*, (2):cnab011, 2021.

[5] R. Pastor-Satorras, C. Castellano, P. Van Mieghem, and A. Vespignani. Epidemic processes in complex networks. *Reviews of Modern Physics*, 87:925–979, 2015.