Disease Transmission and Epidemic Thresholds on Generalized Random Hypergraph Models

Grisham Paimagam
Project Mentor: Anthony Della Pella

September 13, 2025

1. Introduction

Random graphs models are models that arise when removing edges randomly with a certain probability distribution, or choosing each possible graph with a probability distribution. The earliest and most well-known random graph model is the Erdos-Renyi model. This model, commonly referred to as $G_{n,p}$, chooses a particular "possible" edge between two vertices (where the vertex set has size of n) to be present within the resulting graph with fixed probability p. Many results have been found regarding the threshold values of p such that certain properties appear within $G_{n,p}$ with almost absolute certainty.

A hypergraph is a generalization of a graph. In normal graphs, an edge will connect only two vertices. However, in a hypergraph, an edge (denoted as *hyperedge*) can connect any subset of the vertex set. For example, a hyperedge can connect 3 vertices, all the vertices, or simply just 1 vertex. There are relatively few works that have explored the notion of a "random hypergraph", and what the possible models for a random hypergraph would look like.

2. Literature Review

Random graphs has been a relatively new field of interest, and was first introduced by thee collection of papers by Erdos and Renyi: Erdos *et al.* [1960], Erdős & Rényi [1961], and Erdős & Rényi [1966], as well as the relatively simple Erdos-Renyi random graph model. Most research that is done on random graphs examines specific or general properties that occur in the random graph. More specifically, they examine which values of p could nearly guarantee the existence of a property and which values of p could nearly guarantee the absence of a property. A very recent example of this is the Kahn-Kalai conjecture, Kahn & Kalai [2007], and the proof of the conjecture, Park & Pham [2024].

As said in the introduction, hypergraphs are generalizations of normal graphs. Ouvrard [2020] gives an introduction and review to hypergraphs if more background knowledge is needed. What makes a hypergraph useful is its ability to model more complex connections that are more often seen in the real world. Instead of one edge simply connecting two vertices, a hyperedge can group many vertices together, adding further detail. This increase in detail makes hypergraphs an especially attractive topic to apply to real-world studies such as epidemiology, or genetics.

Now, combining these two concepts gives birth to the notion of a random hypergraph. Because of the complexity of a hypergraph, a random hypergraph can be of many forms. There are relatively few works that deeply explore possible ideas of random hypergraph, but a few of them are: Barthelemy [2022], Bergman & Leskelä [2024], and Ghoshal *et al.* [2009]. The first paper listed focuses mainly on the types of random hypergraphs, giving more of an overview rather than focusing on specific properties of each random hypergraph model. The third paper focuses on the real-world applications of generalizing random graphs to random hypergraphs. On the other hand, the second paper focuses on a topic that is nearly identical to this paper, as they focus on the one of the models shown here and same notion of connectivity. However, they do not go into depth about strength between hyperedges or vertices, and they don't develop concrete bounds for p, the probability.

3. Motivation for Models

The models shown throughout the paper are motivated by simply taking how random graph models are constructed and forcing them to fit into a hypergraph setting. In normal graphs, we see that a possible edge will *exist* in the resulting graph by random chance. In order to replicate this in a hypergraph setting, we want a possible subset of the vertex set to appear by random chance. Of course, we also need the sum of the probabilities of each particular hyperedge occurring to be equal to 1, so we have a valid probability distribution. Led by this, we have a couple of options: Either have each of the $2^{|V|}$ possible edges appear with a fixed probability p (which would result in a simple hypergraph); have each of the $2^{|V|}$ possible edges appear with a more complex probability distribution $\mathcal{P}$ where the probability is not constant (again, this would result in a simple hypergraph); or fix the number of hyperedges, then one by one for each hyperedge, add a vertex with a fixed probability p (resulting in a not necessarily simple hypergraph). Of course, these aren't the only possible random hypergraph models, but they are the most natural and easy to work with. Future work can most likely be done on more generalized probability distributions rather than just having a fixed probability.

4. Connection Between Vertices

Build a (not necessarily simple) hypergraph $\mathcal{H}$ with L edges and N vertices as follows: For each edge E_i, add a vertex v to E_i with probability p, where p is fixed. If X is the event that $E_i = E$ for at least one i, where E is a possible edge in $\mathcal{H}$, then clearly we have

$$\mathbb{P}[X] = 1 - \mathbb{P}[E_i \neq E] = 1 - \left(1 - p^{|E|}(1-p)^{k-|E|}\right)^n$$

Define *strength* to be the number of edges that contain two fixed vertices u, v. The expected value of strength of any two vertices u, v is given by

$$\begin{aligned} \mathbb{E}[\text{strength}] &= \sum_{E \subset V/\{u,v\}} \left(1 - \left(1 - p^{|E|+2}(1-p)^{k-|E|-2}\right)^n\right) \\ &= 2^{|V|-2} - \sum_{E \subset V/\{u,v\}} \left(1 - p^{|E|+2}(1-p)^{k-|E|-2}\right)^n \end{aligned}$$

Redefine X to be $1 - p^{|E|+2}(1-p)^{k-|E|-2}$ and note that

$$1 - \frac{1}{2^{N-2}} \cdot \mathbb{E}[\text{strength}] = \mathbb{E}[X^n] \geq a^N \cdot \Pr[X \geq a]$$

For large k, $\Pr[X \geq a] = 1$ for all possible values of p. Therefore we have that $\mathbb{E}[\text{strength}] \leq 2^{N-2} \cdot (1 - a^L)$ implying that if L is small and N is large then the expected strength is extremely small. We can get an exact bound which is as follows: If

$$2 \log_{p-p^2}(1-a) \leq N \leq \log_2 \left(\frac{4\delta}{1-a^L}\right)$$

then $\mathbb{E}[\text{strength}] \leq \delta$

Branching out from this crude bash, we can actually get a very nice expression for $\mathbb{E}[\text{strength}]$. We use the indicator variables $x_1, x_2, \ldots, x_L$, where x_i denotes if the edge E_i contains both selected vertices. Note that

$$\begin{aligned} \mathbb{E}[\text{strength}] &= \mathbb{E}[x_1 + x_2 + \cdots + x_L] \\ = \mathbb{E}[x_1] + \mathbb{E}[x_2] \cdots &+ \mathbb{E}[x_L] = L \cdot \mathbb{E}[x_1] = p^2 \cdot L \end{aligned}$$

Definition 4.1. Two vertices are *connected* if they share at least one edge

Definition 4.2. The strength of the connection between two vertices is the number of edges they share. Two vertices are *j-connected* if the strength is at least j.

Define the strength of two selected vertices to be S. We can apply Markov's Inequality to see that $\mathbb{E}[S] \geq j\mathbb{P}[S \geq j]$. We can create another (normal) graph G_j, where each vertex in G represents a vertex in $\mathcal{H}$, and two vertices in G_j are connected if and only if their strength in $\mathcal{H}$ is at least j. Note that G_j can be modeled as the common Erdos-Renyi random graph model, where two vertices are connected with probability $\mathbb{P}[S \geq j]$. It is well-known that in a $G_{n,p}$ model, if $np > 1$, then G will almost surely contain a giant connected component, while if $np < 1$, then G will almost surely not contain a connected component. Applying this to G_j, we have that if $\mathbb{P}[S \geq j] > \frac{1}{|V|}$ then $\mathcal{H}$ will almost surely contain a giant j-connected component. On top of this, if we have $\mathbb{E}[S] = p^2 \cdot L < \frac{j}{|V|}$ then $\mathcal{H}$ will not contain a j-strong connected component of vertices with size more than $O(\log |V|)$. This is simply equivalent to

$$p < \sqrt{\frac{j}{|V| \cdot |E|}} \tag{1}$$

We can also generate results based on j-disconnectivity for $\mathcal{H}$. Note that if $\mathbb{E}[S] = p^2 \cdot |V| < \frac{j \cdot \ln L}{L}$, then G_j will almost certainly be disconnected, and as a result $\mathcal{H}$ will be j-disconnected. This gives rise to the following theorem and lemma:

Theorem 4.3. *If $p < \sqrt{\frac{j \cdot \ln L}{L \cdot N}}$, then $\mathcal{H}$ will be j-disconnected.*

Lemma 4.4. *If $p < \sqrt{\frac{\ln L}{L \cdot N}}$ then $\mathcal{H}$ will be disconnected.*

Let $\mathcal{E}_i$ be the edge set of G_i. Note that, trivially, $\mathcal{E}_i \subset \mathcal{E}_j$ when $i \leq j$. We examine specifically the transition $\mathcal{E}_i \mapsto \mathcal{E}_{i+1}$, and see that this process simply removes edges in the normal graph G one step at a time. The probability the connection between a pair of vertices is lost in G when we go from $\mathcal{E}_i$ to $\mathcal{E}_{i+1}$ is the probability the strength of that pair of vertices is exactly equal to i. We have

$$\begin{aligned} \mathbb{P}[v_a \underset{S=i}{\longleftrightarrow} v_b] &= \frac{\binom{n}{i} \cdot p^{2i} \cdot (1-p^2)^{n-i}}{2^n} \\ &= P_i \end{aligned}$$

Note that the probability an edge has been lost when the graph undergoes the transition $G_0 \mapsto G_j$ is

$$\sum_{i=0}^{j-1} P_i = \mathcal{P}_j(p)$$

where P_j is a n-degree polynomial in terms of p. If we fix p, and let $\chi = \mathcal{P}_j(p)$, we note that going from $G_0 \mapsto G_j$ removes an edge from a complete $K_{|V|}$ graph with probability χ, meaning that any possible edge is included in the final graph G_j with probability $1 - \chi$. Therefore this transition can be represented as a classic Erdos-Renyi graph model $G_{n,p}$ where $n = |V|$ and $p = 1 - \chi$. Therefore, by classical Erdos-Renyi results, if

$$\chi = \mathcal{P}_j(p) > \frac{|V| - 1}{|V|}$$

then there is sure to not be a giant component in G_j. Because of this there will not be a j-strength component of size larger than $O(\log |V|)$. The converse is also true by classical

Erdos-Renyi results, making $\frac{|V|-1}{|V|}$ the threshold value for $\mathcal{P}_j(p)$. In the previous section, we proved the upper bound of $p < \sqrt{\frac{j}{|V|\cdot|E|}}$. However, we never addressed what happens if $p > \sqrt{\frac{j}{|V|\cdot|E|}}$. Finding an exact solution to $\mathcal{P}_j(p) = \frac{|V|-1}{|V|}$ is incredibly difficult for large values of j, but we can produce results if we focus on small values of j. Note that this idea of $P_j(p)$ is equivalent to the $\mathbb{P}[S \geq j] \leq \frac{1}{|V|}$ notion that was discussed previously but not dissected in depth.

5. Connection Between Edges

Another method we can pursue is examining the connectivity of edges. We have the following definitions:

Definition 5.1. Two edges are *connected* if they share at least one vertex

Definition 5.2. The *strength* of the connection of two hyperedges is the number of vertices they share. If two hyperedges are not connected, then they have a strength of 0.

We say $E_i \sim E_j$ if and only if those edges are connected. Note that

$$\mathbb{P}[E_i \sim E_j] = 1 - (1 - p^2)^{|V|}$$

If we focus only on the "connected" notion, then our random model can be simply defined as a graph where two vertices are connected with probability $\mathcal{P} = 1 - (1 - p^2)^{|V|}$, and are not connected with probability $1 - \mathcal{P}$. From here on out, we let $1 - p^2 = \chi$. Clearly, the vertices in the normal graph represent the edges in the hypergraph. This can be related to the Erdos-Renyi graph model to examine connected components. By properties of Erdos-Renyi, we have that the threshold value of $\mathcal{P} = 1 - \chi^{|V|}$ for having a giant connected component is $\frac{1}{|E|}$. This means that if p is fixed, then the "threshold" value for the amount of vertices is $\log_\chi\left(\frac{|E|-1}{|E|}\right)$. Basically, this means that if $|V| < \log_\chi\left(\frac{|E|-1}{|E|}\right)$, then there is sure to not exist a connected component of size larger than $O(\log|E|)$. On the other hand, if $|V|$ is larger than this value, then there is sure to be a giant connected component of hyperedges.

We now branch out into *strong connected components*, where we only focus on pairs of hyperedges with a certain strength.

Definition 5.3. Two edges are j-connected, if they have a strength of at least j

Definition 5.4. A j-strong connected component is a connected component of hyperedges, where every hyperedge is j-connected to at least one other hyperedge in the component.

Let S denote the strength between two edges. Note that $\mathbb{E}[S] \geq j\mathbb{P}[S \geq j]$ by Markov's Inequality. Define a normal graph G now where the vertices represent the edges in $\mathcal{H}$, and two vertices are connected in G if and only if their corresponding edges are j-connected. By well-known results from the Erdos-Renyi model, if $\mathbb{P}[S \geq j] > \frac{1}{|E|}$ then, G will almost surely contain a giant component. Also we have if $\mathbb{E}[S] < \frac{j}{|E|}$ then G will almost surely not contain a j-strong connected component of size more than $O(\log |E|)$.

Now, we claim that $\mathbb{E}[S] = p^2 \cdot |V|$. Note that each vertex has a p^2 probability of being in both edges. We can employ linearity of expectation and indicator variables. Let x_i be the indicator variable for if the ith vertex is in both edges. We have

$$\mathbb{E}[S] = \mathbb{E}[x_1 + x_2 \cdots + x_{|V|}] =$$
$$\mathbb{E}[x_1] + \mathbb{E}[x_2] \cdots + \mathbb{E}[x_{|V|}] = p^2 \cdot |V|$$

Because of the previous claim, we have that, in fact, if

$$p < \sqrt{\frac{j}{|E| \cdot |V|}}$$

there will almost surely not contain a j-connected component of hyperedges of size more than $O(\log |E|)$. Note that this is the exact same upper bound seen in the previous section: Eq. (1), which gives us the following theorem:

Theorem 5.5. *If* $p < \sqrt{\frac{j}{|E| \cdot |V|}}$, *then there will not exist a* j*-connected component of either vertices or edges of size more than* $\max(O(\log |E|), O(\log |V|))$

We also have that if $\mathbb{E}[S] = p^2 \cdot N < \frac{j \cdot \ln L}{L}$ then $\mathcal{H}$ will almost certainly be j-disconnected. In fact, letting $j = 1$ gives the previous situation discussed with connectivity, and gives another bound for p.

6. Code Implementation And Pictures

The following pictures depict the resulting normal graph that arises when we represent each hyperedge in Model 2 as a vertex, and connect two vertices in the normal graph if the two edges that correspond with the vertices have at least a specific amount of strength. The vertices that are clustering towards the outer boundary are not part of a connected component, and the vertices that are clustered towards the middle are part of a connected component.

```python
import networkx as nx
import matplotlib.pyplot as plt
import random
from networkx.generators.random_graphs import erdos_renyi_graph
```

```python
e = 400
v = 400
p = 0.006
#Sets the parameters for the graph: e is the number of edges, v is the
    number of vertices, and p is the probability
k = 1
#The strength threshold for determining if two vertices will be connected
    or not
g = nx.Graph()
arr = [ [0 for _ in range(v)] for _ in range(e)]
for i in range(e):
  g.add_node(i)
counter = 0
for i in range(e):
  for j in range(v):
    x = random.uniform(0,1)
    if (x <= p):
      arr[i][j] = 1
    else:
      arr[i][j] = 0
#Sets up the array by randomly assigning arr[i][j] to 1 or 0 if the jth
    vertex is included in the ith edge or not
for i in range(e):
    for j in range(i):
      strength = 0
      for l in range(v):
        if ((arr[i][l] == 1) and (arr[j][l] == 1)):
          strength = strength + 1
      if strength > k:
        g.add_edge(i, j)
#Checks each edge to see if their strength is above the threshold given at
    the beginning. The program will connect the two corresponding vertices
    in the final normal graph if it is.
largest_cc = max(nx.connected_components(g), key=len)
print(len(largest_cc))
print(g.nodes)
print(g.edges)
nx.draw(g)
plt.show()
#Prints the graph and certain values
```

Listing 1: Code representing the Model 2 graph where each hyperedge is represented by a vertex

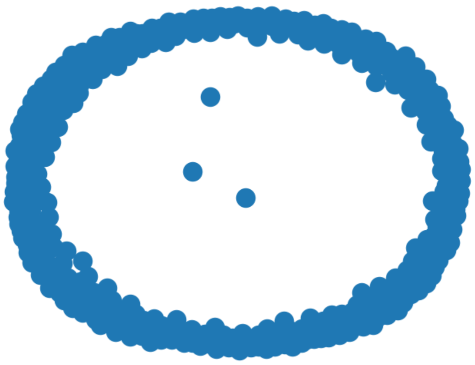

Figure 1: $V = 400$, $E = 400$, $p = 0.0025 = \sqrt{\frac{j}{|V| \cdot |E|}}$, $j = 1$, The largest connected component is of size 2, which is slightly lower than $O(\log |E|)$

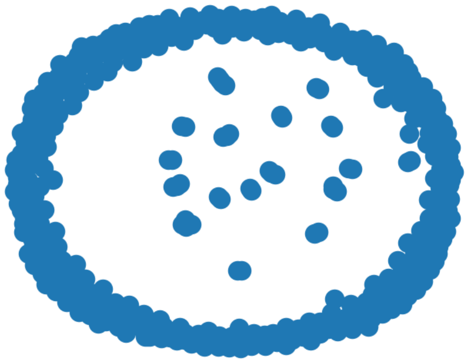

Figure 2: $V = 400$, $E = 400$, $p = 0.008$, $j = 1$, The largest connected component is of size 5, which is approximately higher than $O(\log |E|)$. There is also a significantly higher number of connected components (vertices clustered in the middle)

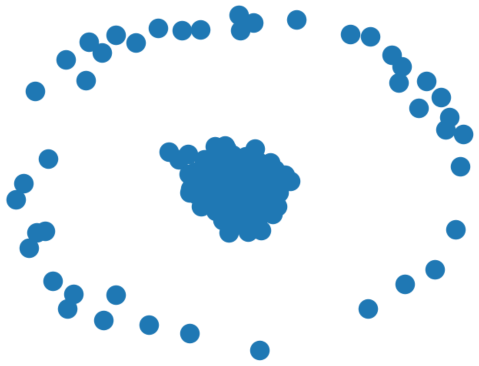

Figure 3: $V = 400$, $E = 400$, $p = 0.02$, $j = 1$, The largest connected component is of size 356, which is approximately $O(|E|)$. Nearly all of the vertices are apart of the very giant connected component in the middle.

7. Disease Transmission On Networks

In this section, we will discuss the transmission of diseases in a population, by modeling this process with a random hypergraph model. In previous papers, such as Newman [2002] and Zhao & Magpantay [2024], epidemics and disease spread are modeled using a random graph and percolation model. This section aims to improve the accuracy of these papers by using hypergraphs to increase the complexity and better model what happens in the real world. We examine three models in this section, ordered by simplicity.

Define a hypergraph $\mathcal{H}$, with $N = |V|$ vertices, and $L = |E|$ hyperedges. We assume both L and N are extremely large. Each hyperedge represents a group of people that are in close contact with each other, such as families or workspaces.

Each node in the hypergraph is apart of one of the following 3 groups at once: Susceptible, Infected, and Removed. Susceptible nodes are nodes that have not yet been infected and are capable of being infected; infected nodes are nodes that are currently infected; while removed nodes are nodes that have either recovered from the infection and cannot transmit the disease to others, or they have died. An infected node can only transmit the disease to a node that is in the same hyperedge as the infected node.

Definition 7.1. The transmissibility T of a disease, is the average probability that an edge carries the disease (or is "occupied") from an infected node i to susceptible node j.

Theorem 7.2 (Zhao & Magpantay [2024])**.**

$$T = \frac{\beta}{\beta + \gamma}$$

where β is the probability, per unit time, that the disease is transmitted from an infected node i to susceptible node j, while γ is the probability, per unit time, that an infected node recovers.

Proof. First, we examine the transmissibility between a fixed pair of infected and susceptible nodes i and j. Denote this value as T_{ij}. Now, denote the recovery time of node i to be τ_i, where $\mathbb{E}[\tau] = \frac{1}{\gamma}$. We have that

$$1 - T_{ij} = \lim_{\Delta t \to 0} (1 - \beta \Delta t)^{\frac{\tau_i}{\Delta t}} = e^{-\beta \tau_i}$$

where both sides represent the probability that the disease is NOT transmitted over a continuous interval.

Now, we wish to find $T = \mathbb{E}[T_{ij}]$. In order to do this, we make the assumption that the probability distribution of τ is $f_\tau(t) = \gamma e^{-\gamma t}$. Note that

$$\int_0^\infty f_\tau(t)\, dt = 1 \quad \text{and} \quad \int_0^\infty t f_\tau(t)\, dt = \frac{1}{\gamma} = \mathbb{E}[\tau]$$

so this probability distribution is valid. Now we can directly calculate

$$\begin{aligned} T = \mathbb{E}[T_{ij}] &= 1 - \mathbb{E}[e^{-\beta\tau_i}] \\ &= 1 - \int_0^\infty e^{-\beta t} \cdot f_\gamma(t)\, dt \\ &= 1 - \gamma \int_0^\infty e^{-(\beta+\gamma)t}\, dt \\ &= \frac{\beta}{\beta+\gamma} \end{aligned}$$

□

Now, we create a normal graph G, with the same vertex set V. We define this to be the *projection* graph of $\mathcal{H}$. We connect two nodes i and j if and only if there exists a hyperedge e in $\mathcal{H}$ such that $i, j \in e$. Otherwise, we leave these two nodes disconnected. Recall the following theorem:

Theorem 7.3 (Newman [2002]). *The critical transmission threshold, T_c, for the emergence of a occupied giant component (e.g. the disease has spread to the majority of the population) for a regular random graph and percolation model is given by*

$$T_c = \frac{\langle k \rangle}{\langle k^2 \rangle - \langle k \rangle} \tag{2}$$

where $\langle k^n \rangle$ represents the nth moment of the degree distribution.

This threshold also generally holds true for extremely large, but finite, graphs so we assume this is applicable to our graph G.

Theorem 7.4. *The expected number of vertices in multiple hyperedges is bounded above by $\mathbf{L}^2\mathbf{Z}^2\mathbf{N}^{-1}$ where Z represents the expected size of a hyperedge.*

Proof. Let ϵ_{ij} represent how many vertices are in both edges e_i and e_j. We have

$$\mathbb{E}[\text{vertices in} \geq 2 \text{ hyperedges}] < \mathbb{E}\left[\sum_i \sum_j \epsilon_{ij}\right]$$

$$= \sum_i \sum_j \mathbb{E}[\epsilon_{ij}] = \binom{L}{2} \cdot \mathbb{E}[\epsilon_{ij}] = \binom{L}{2} \cdot \mathbb{E}[S]$$

where S represents the strength between two hyperedges. Assume $\mathfrak{L}$ represents the size distribution of the hyperedges. Now we have

$$\begin{aligned}\mathbb{E}[S] = N \cdot \sum_{i=0}^{N}\sum_{j=0}^{N} [\mathfrak{L}(i)\mathfrak{L}(j) \cdot \mathbb{P}[v \in E_a \text{ if } |E_a| = i] \cdot \mathbb{P}[v \in E_b \text{ if } |E_b| = j]] \\ = N \cdot \sum_{i=1}^{N}\sum_{j=1}^{N} \left[\mathfrak{L}(i)\mathfrak{L}(j) \cdot \frac{ij}{N^2}\right] \\ = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{N} \mathfrak{L}(i)\mathfrak{L}(j) \cdot ij \\ = \frac{1}{N}\left(\sum_{i=1}^{N}\mathfrak{L}(i) \cdot i\right)\left(\sum_{j=1}^{N}\mathfrak{L}(j) \cdot j\right) \\ = \frac{(\mathbb{E}[\mathcal{Z}])^2}{N}\end{aligned}$$

where $\mathbb{E}[\mathcal{Z}]$ represents the expected size of a hyperedge. □

7.1. Uniform Random Hypergraph Model

In this subsection, we look at one of the simplest random hypergraph model: uniform hypergraphs. In a d-uniform random hypergraph, all hyperedges have a fixed size of d, and each hyperedge is formed by randomly selecting d vertices.

We denote the events $\langle k \rangle$ and $\langle k^2 \rangle$ as X, Y respectively, as well as assume N outgrows d significantly as $N \to \infty$

Lemma 7.5. *We have the following approximations:*

$$\begin{aligned}\mathbb{E}[X] = (N-1) \cdot \left(1 - \left(1 - \frac{d(d-1)}{N(N-1)}\right)^L\right) \approx \frac{Ld^2}{N} \\ \mathbb{E}[Y] \approx \frac{Ld^2}{N} + \frac{Ld^3}{N} + \frac{L^2d^4}{N^2}\end{aligned}$$

Proof. We have

$$\begin{aligned}
\mathbb{E}[X] = \frac{1}{N}\sum_i \mathbb{E}\left[\deg(v_i)\right] = \mathbb{E}[\deg(v)] \\
= (N-1)\cdot \mathbb{P}[\text{Two vertices are connected}] \\
= (N-1)\cdot(1-\mathbb{P}[\text{Two vertices are not connected}]) = (N-1)\cdot\left(1-\left(1-\frac{\binom{N-2}{d-2}}{\binom{N}{d}}\right)^L\right) \\
= \left(1-\left(1-\frac{d(d-1)}{N(N-1)}\right)^L\right) \approx \frac{Ld^2}{N}
\end{aligned}$$

where the last approximation is valid as we assume $d, N \to \infty$ and $\frac{d}{N} \ll 1$ for large d, N.

By similar logic, we have

$$\mathbb{E}[Y] = \frac{1}{N}\sum_i \mathbb{E}[\deg(v_i)^2] = \mathbb{E}[\deg(v)^2]$$

The key here is to fix a vertex ℓ and then use indicator variables $\epsilon_1, \epsilon_2, \dots, \epsilon_{N-1}$ where

$$\epsilon_j = \begin{cases} 1, & \text{if } j, \ell \text{ connected} \\ 0, & \text{otherwise} \end{cases}$$

So now we have,

$$\begin{aligned}
\mathbb{E}[\deg(v)^2] = \mathbb{E}\left[\left(\sum_{i=1}^{N-1}\epsilon_i\right)^2\right] = \sum_{i=1}^{N-1}\mathbb{E}[\epsilon_i^2] + \mathbb{E}\left[2\sum_{i\neq j}\epsilon_i\epsilon_j\right] \\
= (N-1)\cdot\mathbb{E}[\epsilon_1^2] + 2\binom{N-1}{2}\mathbb{E}[\epsilon_1\epsilon_2] = \mathbb{E}[X] + 2\binom{N-1}{2}\mathbb{E}[\epsilon_1\epsilon_2]
\end{aligned}$$

For simplicity and brevity, we let $\alpha = \frac{\binom{N-3}{d-3}}{\binom{N}{d}} = \frac{\binom{d}{3}}{\binom{N}{3}}$ and $\lambda = \frac{\binom{N-2}{d-2}}{\binom{N}{d}} - \frac{\binom{N-3}{d-3}}{\binom{N}{d}} = \frac{\binom{d}{2}}{\binom{N}{2}} - \frac{\binom{d}{3}}{\binom{N}{3}}$. Observe that α is the probability that a hyperedge contains 3 fixed vertices, and λ is the probability that a hyperedge contains 2 of the fixed vertices, but not the 3rd. Note that

$$\mathbb{E}[\epsilon_1\epsilon_2] = \mathbb{P}[\epsilon_1 = 1 \text{ and } \epsilon_2 = 1]$$

so we aim to find the complement of this probability, which we will denote as C. Observe

$$\begin{aligned}
\mathbb{P}[C] = \mathbb{P}[\epsilon_1 = 0 \text{ and } \epsilon_2 = 0] + 2\mathbb{P}[\epsilon_1 = 0 \text{ and } \epsilon_2 = 1] \\
= (1-\alpha)^L(1-\lambda)^{2L} + 2(1-\alpha)^L(1-\lambda)^L\left(1-(1-\lambda)^L\right)
\end{aligned}$$

Making the approximation $\left(1 - \left(1 - \frac{\binom{d}{i}}{\binom{N}{i}}\right)^L\right) \approx \frac{Ld^i}{N^i}$ for $i = 2, 3$ simplifies our expression down to $1 - \frac{Ld^3}{N^3} - \frac{L^2d^4}{N^4}$ as we expand and ignore higher order terms as they will vanish as $N, d, L \to \infty$. Finally, taking the complement of this, we get

$$2\binom{N-1}{2}\mathbb{P}[\epsilon_1 = 1 \text{ and } \epsilon_2 = 1] \approx N^2 \cdot \left(\frac{Ld^3}{N^3} + \frac{L^2d^4}{N^4}\right) = \frac{Ld^3}{N} + \frac{L^2d^4}{N^2}$$

as we can make the approximation $2\binom{N-1}{2} \approx N^2$ as N is large. The proof is complete. $\square$

Lemma 7.6. *Define* $f(X,Y) = \frac{X}{Y-X}$*. We can approximate* $\mathbb{E}[f(X,Y)]$ *by*

$$\mathbb{E}[f(X,Y)] \approx f(\mu_X,\mu_Y) + \frac{1}{2}\left(\frac{\partial^2 f}{\partial X^2}\bigg|_{(\mu_X,\mu_Y)}\right)\text{Var}[X] + \frac{1}{2}\left(\frac{\partial^2 f}{\partial Y^2}\bigg|_{(\mu_X,\mu_Y)}\right)\text{Var}[Y] + \left(\frac{\partial^2 f}{\partial X \partial Y}\bigg|_{(\mu_X,\mu_Y)}\right)\text{Cov}[X,Y]$$

where $\mathbb{E}[X] = \mu_X$ *and* $\mathbb{E}[Y] = \mu_Y$

Proof. Let $f(X,Y) = \frac{X}{Y-X}$. We have

$$\mathbb{E}[T_c] = \mathbb{E}\left[\frac{X}{Y-X}\right] = \mathbb{E}[f(X,Y)]$$

so we utilize a Taylor Series approximation at the point $(\mathbb{E}[X], \mathbb{E}[Y])$ as follows:

$$f(X,Y) \approx f(\mu_X,\mu_Y) + \frac{\partial f}{\partial X}\bigg|_{(\mu_X,\mu_Y)}(X-\mu_X) + \frac{\partial f}{\partial Y}\bigg|_{(\mu_X,\mu_Y)}(Y-\mu_Y) + \frac{1}{2}\frac{\partial^2 f}{\partial X^2}\bigg|_{(\mu_X,\mu_Y)}(X-\mu_X)^2 + \frac{1}{2}\frac{\partial^2 f}{\partial Y^2}\bigg|_{(\mu_X,\mu_Y)}(Y-\mu_Y)^2 + \frac{\partial^2 f}{\partial X \partial Y}\bigg|_{(\mu_X,\mu_Y)}(X-\mu_X)(Y-\mu_Y).$$

Taking the expectation of both sides, the first degree terms are annihilated.

$$\mathbb{E}[f(X,Y)] \approx f(\mu_X,\mu_Y) + \frac{1}{2}\left(\frac{\partial^2 f}{\partial X^2}\bigg|_{(\mu_X,\mu_Y)}\right)\text{Var}[X] + \frac{1}{2}\left(\frac{\partial^2 f}{\partial Y^2}\bigg|_{(\mu_X,\mu_Y)}\right)\text{Var}[Y] + \left(\frac{\partial^2 f}{\partial X \partial Y}\bigg|_{(\mu_X,\mu_Y)}\right)\text{Cov}[X,Y]$$

$\square$

Lemma 7.5 and 7.6 can be used to create an accurate approximation for the expected epidemic threshold for $\mathcal{H}$. However, if we assume $d \propto \sqrt{N}$ and $L \propto N$ as $N \to \infty$, then we can create sharper results as follows. We assume $d = C_d\sqrt{N}$ where C_d remains constant and $L = C_L N$ where C_L remains constant.

Lemma 7.7. *Let Z be the event that any two fixed vertices are connected. We have*

$$\lim_{N\to\infty} \mathbb{P}[Z] = 1 - \exp\{-C_d^2 \cdot C_L\}$$

Proof.

$$\lim_{N\to\infty} \mathbb{P}[Z] = \lim_{N\to\infty} \left(1 - \left(1 - \frac{d(d-1)}{N(N-1)}\right)^L\right) = \lim_{N\to\infty} \left(1 - \left(1 - \frac{d^2}{N^2}\right)^L\right)$$
$$= \lim_{N\to\infty} \left(1 - \left(1 - \frac{C_d^2}{N}\right)^{C_L N}\right) = 1 - \exp\{-C_d^2 \cdot C_L\}$$

□

Therefore, any two vertices in G are connected to probability $1-\exp\{C_d^2 \cdot C_L\}$ and when the percolation process is applied, the edges are kept with probability $T = \frac{\beta}{\beta+\gamma}$, so the edges are kept in the final "transmission graph" G_T with probability $\frac{\beta}{\beta+\gamma} \cdot \left(1 - \exp\{C_d^2 \cdot C_L\}\right)$. Note that this process is essentially equivalent to applying a percolation process on a complete K_N graph with the same vertex set as G, except that the "transmissibility" is $\mathcal{T} = \frac{\beta}{\beta+\gamma} \cdot \left(1 - \exp\{-C_d^2 \cdot C_L\}\right)$, instead. Therefore by Eq. (2), the critical transmissibility value is

$$T_c = \frac{\langle k \rangle}{\langle k^2 \rangle - \langle k \rangle} = \frac{N-1}{(N-1)^2 - (N-1)} = \frac{1}{N-2}$$

so we have the following results:

Theorem 7.8. *Assuming that $d \propto \sqrt{N}, L \propto N$ with constants of proportionality of C_d and C_L, the disease will die out and there will be no epidemic if and only if*

$$\frac{\beta}{\beta+\gamma} \cdot \left(1 - \exp\{-C_d^2 \cdot C_L\}\right) < \frac{1}{N-2} \approx \frac{1}{N}$$

Otherwise, an epidemic will occur.

Theorem 7.9. *Assuming that $d \propto N^\alpha, L \propto N^{2-2\alpha}$ with constants of proportionality of C_d and C_L, the disease will die out and there will be no epidemic if and only if*

$$\frac{\beta}{\beta+\gamma} \cdot \left(1 - \exp\{C_d^2 \cdot C_L\}\right) < \frac{1}{N-2} \approx \frac{1}{N}$$

Otherwise, an epidemic will occur.

If d and L do not satisfy such proportionalities with N, then we could always use the exact formula (albeit uglier) for the probability that two edges are connected, which is $1-\left(1-\frac{d(d-1)}{N(N-1)}\right)^L$.

7.2. Non-Uniform Random Hypergraph Model

For this section, we adopt the model explored in sections 4 and 5. To recap, the model is defined as follows:

Definition 7.10. Create a hypergraph $\mathcal{H}$ with N vertices, given by $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$ and L edges, given by $\mathcal{E} = \{E_1, E_2, \ldots, E_L\}$. We generate edge E_i by going through each vertex, and adding it to E_i with probability p. More formally, $v_j \in E_i$ with probability p for all (i, j).

We write $v_i \sim v_j$ if vertices i and j are connected. Note that

$$\mathbb{P}[v_i \sim v_j] = 1 - (1-p^2)^L$$

so utilizing the same strategy we used in the previous model, two vertices are connected in the final "transmission" graph G_T with probability $T\left(1-(1-p^2)^L\right)$, so applying Eq. (2) to a complete K_N graph, we have the following theorem:

Theorem 7.11. *The disease will die out and there will be no epidemic if and only if*

$$\frac{\beta}{\beta+\gamma} \cdot \left(1-(1-p^2)^L\right) < \frac{1}{N-2} \approx \frac{1}{N}$$

Now, we shift to a matrices point of view. Now, instead of having one or very few initial infected vertices, we infect each vertex right at the start with probability P_0. Let p_i denote the probability that v_i eventually gets infected, and let ζ_i be the probability v_i gets infected by transmission from an adjacent vertex. We have

$$\begin{aligned} 1-\zeta_i &= \left(\sum_{m=0}^{\deg(v_i)} T^m \cdot (1-T)^{\deg(v_i)-m} \cdot [S_m]\right) \\ &= \prod_{j\sim i}(1-Tp_j) \approx 1 - T\sum_{j\sim i} p_j \end{aligned}$$

where S_m is an mth symmetric sum of the set $\{1-p_j \mid j \sim i\}$ and the last step stems from the approximation of $(1-a)(1-b) \approx 1-a-b$ where $a, b \ll 1$. Now, we have

$$\begin{aligned} 1-p_i = (1-P_0)\cdot(1-\zeta_i) &= 1-P_0-\zeta_i(1-P_0) \\ &= 1-P_0-T(1-P_0)\sum_{j\sim i} p_j \end{aligned}$$

Putting this in matrix form we have $\mathbf{P} = \mathbf{P}_0 + T(1-P_0)\mathbf{AP}$ where $\mathbf{P}_0$ is a N-dimensional vector with every entry being P_0, $\mathbf{A}$ is the adjacency matrix of $\mathcal{H}$ (where we use the same definition of connectivity between vertices as before), and $\mathbf{P}$ is the infection probability vector for all nodes (where the ith entry is p_i). Note that the above equality can be rearranged to $\mathbf{SP} = \mathbf{P}_0$ where $\mathbf{S}$ is the *system* matrix represented by

$$\mathbf{S} = \mathbf{I} - T(1-P_0)\mathbf{A}$$

where $\mathbf{I}$ is the identity matrix.

Theorem 7.12.

$$\|\mathbf{P}\| \leq \left| \frac{P_0 \cdot \sqrt{N}}{[1 - T(1-P_0)]\, \lambda_{N,A}} \right|$$

Proof. First, we prove the following lemma (a similar lemma is seen in Wang *et al.* [2003]):

Lemma 7.13. *The eigenvectors of* $\mathbf{S}$ *are equivalent to the eigenvectors of the adjacency matrix* $\mathbf{A}$*, and the eigenvalues of* $\mathbf{S}$ *are given by* $\lambda_{i,S} = [1 - T(1-P_0)]\, \lambda_{i,A}$*, where* $\lambda_{i,A}$ *is an eigenvalue of* $\mathbf{A}$*.*

Proof. Take an eigenvector $\mathbf{u}_{i,S}$. We have that

$$\begin{aligned} \mathbf{S}\mathbf{u}_{i,S} &= (\mathbf{I} - T(1-P_0)\mathbf{A})\mathbf{u}_{i,S} \\ (\lambda_{i,S} - 1)\mathbf{u}_{i,S} &= -T(1-P_0)\mathbf{A}\mathbf{u}_{i,S} \end{aligned}$$

so $\mathbf{u}_{i,S}$ is an eigenvector of $\mathbf{A}$. Now, assume the eigenvalue of this eigenvector with respect to $\mathbf{A}$ is $\lambda_{i,A}$. We now have that

$$(\lambda_{i,S} - 1)\mathbf{u}_{i,S} = -T(1-P_0)\lambda_{i,A}\mathbf{u}_{i,S}$$

so we require $\lambda_{i,S} = [1 - T(1-P_0)]\, \lambda_{i,A}$ □

Order the eigenvalues of $\mathbf{S}$, such that $|\lambda_{1,S}| \geq |\lambda_{2,S}| \geq \cdots \geq |\lambda_{N,S}|$. We have that

$$\begin{aligned} \|\mathbf{P}\| = \left\|\mathbf{S}^{-1}\mathbf{P}_0\right\| &\leq \left\|\mathbf{S}^{-1}\right\| \cdot \|\mathbf{P}_0\| \\ = \left|\frac{1}{\lambda_{N,S}}\right| \cdot P_0\sqrt{N} &= \left| \frac{P_0 \cdot \sqrt{N}}{[1 - T(1-P_0)]\, \lambda_{N,A}} \right| \end{aligned}$$

so we are done. □

This system of equations can also easily be used for situations where only one node is infected initially. This setup would allow for the system of equations as follows (assume N is the infected node):

$$\begin{cases} 1 - p_i = \prod_{j \sim i}(1 - Tp_j), & i \neq N \\ p_N = 1 \end{cases}$$

However, in this situation we would not be able to express the system of equations in matrix form. From this we receive the following corollary

Corollary 7.14. *In a hyperedge with size N (or equivalently, a K_N graph), the probability a node gets infected p satisfies*

$$1 - p = (1 - T)(1 - Tp)^{N-2}$$

Proof. Note that, by symmetry, all nodes that are not initially infected have the same exact probability of eventually getting infected. Therefore $p = p_1 = p_2 = \cdots = p_{N-1}$, and substituting this, as well as $p_N = 1$, gives the desired result. □

Now using the approximation $(1 - Tp)^{N-2} \approx 1 - (N - 2)Tp$, we have that

$$p = \frac{T}{1 - (T - T^2)(N - 2)}$$

7.3. An Extra-Realistic Model

In this model the set of hyperedges, E, is split into three groups: small, medium, and large. Small groups represent groups like families, while medium groups represent groups like workspaces or friend groups, and large groups represent groups like concert crowds or event attendees. The distribution of the L hyperedges that go into these groups are represented by the variable weights $w = \{\omega_1, \omega_2, \omega_3\}$, respectively. The size distribution for each group is as follows: small hyperedges are Poisson with $\lambda = 3$, medium hyperedges are uniform on $[5, 15]$, and large hyperedges are power-law with bounds $[20, 200]$ and $\alpha = 2.5$. The probability distribution of the size of any hyperedge can therefore be given by

$$\mathfrak{L}(|e|) = \omega_1\mathcal{P}_1 + \omega_2\mathcal{P}_2 + \omega_3\mathcal{P}_3$$

where $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3$ represent the small, medium, and large probability distributions respectively.

The same methods we used for the previous two models can also be used for this model. If we use the same projection graph method that we utilized in previous models, we have the following theorem for this particular model:

Theorem 7.15. *The disease will die out and there will be no epidemic if and only if*

$$\frac{\beta}{\beta + \gamma} \cdot \mathbb{P}[i \sim j] < \frac{1}{N - 2} \approx \frac{1}{N}$$

Otherwise, an epidemic will occur.

We can also characterize $\mathbb{P}[i \sim j]$, with the following. Observe that

$$\mathbb{P}[i \sim j] = 1 - \left(1 - \sum_{i=0}^{N} \mathfrak{L}(i) \cdot \frac{\binom{N-2}{i-2}}{\binom{N}{i}}\right)^{L}$$
$$= 1 - \left(1 - \sum_{i=0}^{N} \mathfrak{L}(i) \cdot \frac{\binom{i}{2}}{\binom{N}{2}}\right)^{L}$$

$\mathfrak{L}$ can be decomposed into the smaller size distributions, $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3$ as necessary. It is also notable we can use an eigenvalue point of view through Theorem 7.12 as it also applies to this model (as well as any other random hypergraph model).

7.4. Containment Strategies

In this section, we explore various containment strategies that could be implemented to control the spread of the epidemic. For all of these strategies, we assume that we work with the *Non-Uniform Random Hypergraph Model* for simplicity. First, we examine the most basic strategy: random vaccination. In this scenario, we vaccinate a node with probability q, where vaccination essentially moves the node to the "Removed" state immediately. A vaccinated node cannot transmit or receive the disease. Analyzing this situation is quite simple if we use similar methods that were utilized in previous sections. As usual, let G be the projection graph of hypergraph $\mathcal{H}$. However, in this case, we slightly tweak the definition of a projection graph, as follows:

Definition 7.16. The vertex set of the projection graph G is all vertices v in $\mathcal{H}$ that are not vaccinated, and two vertices in G are connected if and only if they share a hyperedge in $\mathcal{H}$

Now, we have that

$$\mathbb{P}[i \sim j] = q^2 \cdot \left(1 - (1 - p^2)^L\right)$$

where $i \sim j$ represents the event that vertices i, j exist and are connected in the projection graph G. From this, we have the following theorem

Theorem 7.17. *The disease will die out and there will be no epidemic if and only if*

$$\frac{\beta}{\beta + \gamma} \cdot \left(1 - (1 - p^2)^L\right) < \frac{1}{q^2(N-2)} \approx \frac{1}{q^2 N}$$

Otherwise, an epidemic will occur.

It follows that if we randomly vaccinated about qN people in a population of N people increases the epidemic threshold by a factor of q^{-2}.

Now, we focus on a more targeted vaccination approach. For this scenario, we initially infect each node with probability P_0. Let $f\colon \mathbb{R} \to [0,1]$ be a increasing function that determines the probability that a specific node gets vaccinated. More specifically, for nodes in a hyperedge of size x, they are vaccinated with probability $f(x)$, meaning that larger hyperedges will have larger vaccination probabilities. Note that these vaccination probabilities overlap for each node, so if a node is part of hyperedges with size $\{x_1, x_2\}$ then they are vaccinated with probability $1-(1-f(x_1))(1-f(x_2))$. More generally, we have that

$$q_i = 1 - \prod_{E_j, v_i \in E_j} (1 - f(|E_j|)) \approx \sum_{E_j, v_i \in E_j} f(|E_j|)$$

where q_i represents the probability that the ith node gets vaccinated. Like always, we use p_i to denote the probability that vertex i eventually gets infected. Adding onto this, define ω_i to be the probability v_i gets infected by transmission from an adjacent vertex in this scenario. We have that

$$\begin{aligned} 1-\omega_i &= \prod_{j\sim i}(1 - Tp_j + Tp_j q_j) = 1 - T\sum_{j\sim i} p_j(1-q_j) \\ &= 1 - T\sum_{j\sim i} p_j + T\sum_{j\sim i}\left(p_j\left[\sum_{E_k, v_j\in E_k} f(|E_k|)\right]\right) \end{aligned}$$

so it follows that

$$\omega_i = T\sum_{j\sim i} p_j - T\sum_{j\sim i}\left(p_j\left[\sum_{E_k, v_j\in E_k} f(|E_k|)\right]\right)$$

Now, we characterize p_i with the following:

$$p_i = 1-(1-q_i)(1-P_0)(1-\omega_i) \approx P_0 + \omega_i + q_i$$

Putting this in matrix form, we get that

$$\mathbf{P} = \mathbf{P}_0 + T(\mathbf{AP} - \mathbf{A}(\mathbf{HP})) + \mathbf{Q}$$

where $\mathbf{H}$ is an $N\times L$ matrix with entry (i,j) being $f(|E_j|)$ if $v_i \in E_j$ and 0 otherwise, and $\mathbf{Q}$ is the vaccination vector with the ith entry being q_i. Comparing this matrix equation to the system matrix found before can yield the degree of containment.

8. Code for Disease Models

8.1. Model 1

```python
import numpy as np
import matplotlib.pyplot as plt
from itertools import combinations
import hypernetx as hnx
# --- Parameters ---
n = 20  # Number of vertices
d = 5  # Size of each hyperedge
m = 10  # Number of hyperedges
beta = 0.1  # Probability of infection per contact
gamma = 0.1  # Probability of recovery
time_steps = 100  # Number of simulation steps

# --- Generate d-uniform hypergraph ---
def generate_d_uniform_hypergraph(n, d, m):
    """Generate a d-uniform hypergraph with n vertices and m hyperedges."""
    hyperedges = [tuple(np.random.choice(n, d, replace=False)) for _ in
    range(m)]
    return hyperedges

hyperedges = generate_d_uniform_hypergraph(n, d, m)

# --- Create adjacency list ---
def create_adjacency_list(hyperedges):
    """Create an adjacency list from hyperedges."""
    adjacency_list = {i: set() for i in range(n)}
    for edge in hyperedges:
        for u, v in combinations(edge, 2):
            adjacency_list[u].add(v)
            adjacency_list[v].add(u)
    return adjacency_list

adjacency_list = create_adjacency_list(hyperedges)

# --- Initialize SIR states ---
S = np.ones(n, dtype=bool)  # Susceptible
I = np.zeros(n, dtype=bool)  # Infected
R = np.zeros(n, dtype=bool)  # Removed

# Initial infection
initial_infected = np.random.choice(n, 1, replace=False)[0]
I[initial_infected] = True
S[initial_infected] = False
print(f"Initial infected vertex: {initial_infected}")
print(f"Number of neighbors of the initial infected vertex: {len(
    adjacency_list[initial_infected])}")
print(f"Neighbors of initial infected vertex: {list(adjacency_list[
    initial_infected])}")
def draw_hypergraph(hyperedges):

    edge_dict = {f"e{i}": edge for i, edge in enumerate(hyperedges)}
    H = hnx.Hypergraph(edge_dict)
    hnx.draw(H, with_node_labels=False, with_edge_labels=False)

# Draw the hypergraph
draw_hypergraph(hyperedges)
# --- Simulation ---
def simulate_sir(n, adjacency_list, beta, gamma, time_steps):
    """Simulate the SIR model on a d-uniform hypergraph."""
    S = np.ones(n, dtype=bool)  # Susceptible
    I = np.zeros(n, dtype=bool)  # Infected
    R = np.zeros(n, dtype=bool)  # Removed

    # Initial infection
    initial_infected = np.random.choice(n, 1, replace=False)[0]
    I[initial_infected] = True
    S[initial_infected] = False

    # Record fractions of S, I, R over time
    S_frac, I_frac, R_frac = [], [], []

    for t in range(time_steps):
        new_infections = []

        # Attempt infection
        for u in range(n):
            if S[u]:  # If node u is susceptible
                # Calculate the probability of infection from all infected neighbors
                infected_neighbors = [v for v in adjacency_list[u] if I[v]]
                prob_infection = 1 - (1 - beta) ** len(infected_neighbors)
                if t==0:
                  print(infected_neighbors)

                if np.random.rand() < prob_infection:

                    new_infections.append(u)
        if t==0:
          print(f"Initial infected vertex: {initial_infected}")
          print(f"Number of neighbors of the initial infected vertex: {len(adjacency_list[initial_infected])}")
          print(f"Neighbors of initial infected vertex: {list(adjacency_list[initial_infected])}")
        # Recover infected nodes
        recoveries = np.random.rand(n) < gamma
        R[I & recoveries] = True
        I[I & recoveries] = False

        # Update new infections
        for v in new_infections:
            I[v] = True
            S[v] = False

        # Record fractions
        S_frac.append(np.sum(S) / n)
        I_frac.append(np.sum(I) / n)
        R_frac.append(np.sum(R) / n)

        # Print counts
        print(f"Time Step {t}: Susceptible = {np.sum(S)}, Infected = {np.sum(I)}, Removed = {np.sum(R)}")

    return S_frac, I_frac, R_frac

# --- Run Simulation ---
S_frac, I_frac, R_frac = simulate_sir(n, adjacency_list, beta, gamma, time_steps)
def probability_zero_neighbors(adjacency_list, n):
    """Calculate the probability that a randomly selected vertex has 0 neighbors."""
    zero_neighbor_count = sum(1 for neighbors in adjacency_list.values() if len(neighbors) == 0)
    return zero_neighbor_count / n

p_zero_neighbors = probability_zero_neighbors(adjacency_list, n)
print(f"Probability that a randomly selected vertex has 0 neighbors: {p_zero_neighbors:.4f}")
# --- Plot Results ---
def plot_sir(S_frac, I_frac, R_frac, time_steps):
    """Plot the SIR dynamics."""
    plt.figure(figsize=(10, 6))
    plt.plot(range(time_steps), S_frac, label="Susceptible", color="blue")
    plt.plot(range(time_steps), I_frac, label="Infected", color="red")
    plt.plot(range(time_steps), R_frac, label="Removed", color="green")

    plt.xlabel("Time Steps")
    plt.ylabel("Fraction of Population")
    plt.legend()
    plt.show()

plot_sir(S_frac, I_frac, R_frac, time_steps)
```

8.2. Model 2

```python
import numpy as np
import matplotlib.pyplot as plt
from itertools import combinations

# --- Parameters ---
N = 100  # Number of vertices
```

```python
L = 200  # Number of hyperedges
p = 0.1  # Probability a vertex is part of a hyperedge
beta = 0.1  # Probability of infection per contact
gamma = 0.1  # Probability of recovery
time_steps = 100  # Number of simulation steps

# --- Generate Random Hypergraph ---
def generate_random_hypergraph(N, L, p):
    """Generate a random hypergraph with N vertices and L hyperedges."""
    hyperedges = []
    for _ in range(L):
        edge = [v for v in range(N) if np.random.rand() < p]
        if edge:  # Avoid empty edges
            hyperedges.append(edge)
    return hyperedges

hyperedges = generate_random_hypergraph(N, L, p)

# --- Create adjacency list ---
def create_adjacency_list(hyperedges, N):
    """Create an adjacency list from hyperedges."""
    adjacency_list = {i: set() for i in range(N)}
    for edge in hyperedges:
        for u, v in combinations(edge, 2):
            adjacency_list[u].add(v)
            adjacency_list[v].add(u)
    return adjacency_list

adjacency_list = create_adjacency_list(hyperedges, N)

# --- Initialize SIR states ---
S = np.ones(N, dtype=bool)  # Susceptible
I = np.zeros(N, dtype=bool)  # Infected
R = np.zeros(N, dtype=bool)  # Removed

# Initial infection
initial_infected = np.random.choice(N, 1, replace=False)[0]
I[initial_infected] = True
S[initial_infected] = False
print(f"Initial infected vertex: {initial_infected}")
print(f"Number of neighbors of the initial infected vertex: {len(
    adjacency_list[initial_infected])}")
print(f"Neighbors of initial infected vertex: {list(adjacency_list[
    initial_infected])}")

# --- Simulation ---
def simulate_sir(N, adjacency_list, beta, gamma, time_steps):
    """Simulate the SIR model on a random hypergraph."""
    S = np.ones(N, dtype=bool)  # Susceptible
    I = np.zeros(N, dtype=bool)  # Infected
    R = np.zeros(N, dtype=bool)  # Removed

    # Initial infection
    initial_infected = np.random.choice(N, 1, replace=False)[0]
    I[initial_infected] = True
    S[initial_infected] = False

    # Record fractions of S, I, R over time
    S_frac, I_frac, R_frac = [], [], []

    for t in range(time_steps):
        new_infections = []

        # Attempt infection
        for u in range(N):
            if S[u]:  # If node u is susceptible
                # Calculate the probability of infection from all infected
    neighbors
                infected_neighbors = [v for v in adjacency_list[u] if I[v]]
                prob_infection = 1 - (1 - beta) ** len(infected_neighbors)
                if np.random.rand() < prob_infection:
                    new_infections.append(u)

        # Recover infected nodes
        recoveries = np.random.rand(N) < gamma
        R[I & recoveries] = True
        I[I & recoveries] = False

        # Update new infections
        for v in new_infections:
            I[v] = True
            S[v] = False

        # Record fractions
        S_frac.append(np.sum(S) / N)
        I_frac.append(np.sum(I) / N)
        R_frac.append(np.sum(R) / N)

        # Print counts
        print(f"Time Step {t}: Susceptible = {np.sum(S)}, Infected = {np.
    sum(I)}, Removed = {np.sum(R)}")

    return S_frac, I_frac, R_frac

# --- Run Simulation ---
S_frac, I_frac, R_frac = simulate_sir(N, adjacency_list, beta, gamma,
    time_steps)

# --- Plot Results ---
def plot_sir(S_frac, I_frac, R_frac, time_steps):
    """Plot the SIR dynamics."""
    plt.figure(figsize=(10, 6))
    plt.plot(range(time_steps), S_frac, label="Susceptible", color="blue")
    plt.plot(range(time_steps), I_frac, label="Infected", color="red")
    plt.plot(range(time_steps), R_frac, label="Removed", color="green")

    plt.xlabel("Time Steps")
    plt.ylabel("Fraction of Population")
    plt.legend()
    plt.show()

plot_sir(S_frac, I_frac, R_frac, time_steps)
```

8.3. Model 3

```python
import random
import numpy

def simulate(num_hyperedges, weights, num_nodes, initial_infected,
    time_stamps, beta_value, recovery_rate):
  hyperedge_sizes = []

  num_small = int(weights[0]*num_hyperedges)
  num_medium = int(weights[1]*num_hyperedges)
  num_large = num_hyperedges - num_small - num_medium
  params = {
          'small': {'lambda': 3},
          'medium': {'a': 6, 'b': 15},
          'large': {'gamma': 2.5, 'min_size': 20, 'max_size': 200}
          }
  lambda_param = params['small']['lambda']
  small_size = numpy.random.poisson(lam=lambda_param, size=num_small)
  a,b = params['medium']['a'], params['medium']['b']
  medium_size = numpy.random.randint(a, b, size=num_medium)
  gamma = params['large']['gamma']
  min_size, max_size = params['large']['min_size'], params['large']['
    max_size']
  large_size = []
  for _ in range(num_large):
      size = int(numpy.random.zipf(gamma))
      while size < min_size or size > max_size:
          size = int(numpy.random.zipf(gamma))
      large_size.append(size)

  hyperedge_sizes.extend(small_size)
  hyperedge_sizes.extend(medium_size)
  hyperedge_sizes.extend(large_size)

  hyperedges = []
  for size in hyperedge_sizes:
    hyperedge = random.sample(range(num_nodes), size)
    hyperedges.append(hyperedge)

  node_states = {node: 'S' for node in range(num_nodes)}
  history = []
  for node in initial_infected:
    node_states[node] = 'I'
  for t in range(time_stamps):
    new_states = node_states.copy()
    for hyperedge in hyperedges:
        infected = [node for node in hyperedge if node_states[node] == 'I']
        susceptible = [node for node in hyperedge if node_states[node] == 'S']

        for node in susceptible:
            if random.random() < 1 - (1 - beta_value) ** len(infected):
                new_states[node] = 'I'


        for node in infected:
            if random.random() < recovery_rate:
                new_states[node] = 'R'


    node_states = new_states
    state_counts = {
        'S': list(node_states.values()).count('S'),
        'I': list(node_states.values()).count('I'),
        'R': list(node_states.values()).count('R')
    }
    history.append(state_counts)
  return history


simulate(50000, [0.5, 0.3, 0.2], 10000, [1,2], 1000, 0.001, 0.98)
```

9. Acknowledgement

I would like to thank my mentor Dr. Anthony Della Pella, for introducing me to the research topic of hypergraphs as an extension of normal graphs, and for offering feedback during my research process. I am very grateful for his support throughout this project.

References

Barthelemy, Marc. 2022. Class of models for random hypergraphs. *Physical review e*, **106**(6), 064310.

Bergman, Elmer, & Leskelä, Lasse. 2024. Connectivity of random hypergraphs with a given hyperedge size distribution. *Discrete applied mathematics*, **357**, 1–13.

Erdős, Pál, & Rényi, Alfréd. 1966. On the existence of a factor of degree one of a connected random graph. *Acta math. acad. sci. hungar*, **17**(359-368), 192.

Erdős, Paul, & Rényi, Alfréd. 1961. On the strength of connectedness of a random graph. *Acta mathematica hungarica*, **12**(1), 261–267.

Erdos, Paul, Rényi, Alfréd, *et al.* 1960. On the evolution of random graphs. *Publ. math. inst. hung. acad. sci*, **5**(1), 17–60.

Ghoshal, Gourab, Zlatić, Vinko, Caldarelli, Guido, & Newman, Mark EJ. 2009. Random hypergraphs and their applications. *Physical review e—statistical, non-linear, and soft matter physics*, **79**(6), 066118.

Kahn, Jeff, & Kalai, Gil. 2007. Thresholds and expectation thresholds. *Combinatorics, probability and computing*, **16**(3), 495–502.

Newman, Mark EJ. 2002. Spread of epidemic disease on networks. *Physical review e*, **66**(1), 016128.

Ouvrard, Xavier. 2020. Hypergraphs: an introduction and review. *arxiv preprint arxiv:2002.05014*.

Park, Jinyoung, & Pham, Huy. 2024. A proof of the kahn–kalai conjecture. *Journal of the american mathematical society*, **37**(1), 235–243.

Wang, Yang, Chakrabarti, Deepayan, Wang, Chenxi, & Faloutsos, Christos. 2003. Epidemic spreading in real networks: An eigenvalue viewpoint. *Pages 25–34 of: 22nd international symposium on reliable distributed systems, 2003. proceedings.* IEEE.

Zhao, S, & Magpantay, FMG. 2024. Disease transmission on random graphs using edge-based percolation. *arxiv preprint arxiv:2401.06872*.