Lecture 6: Epidemics and other propagation processes in networks

Eduardo López

It is March, 2020 and COVID-19 is rapidly propagating through the planet as I write this. We have just moved from in-person to online teaching **across the globe**. That would be enough justification for this lecture. But it turns out that propagation processes on networks are one of the key reason they are relevant. Not only do networks (social or otherwise) capture the interrelations between the items that function as nodes, but they also act as the medium in which many kinds of things move, such as information, illness, beliefs, and even behavioral traits. So let's discuss this interesting and important concept of propagation on networks.

1 Some basics

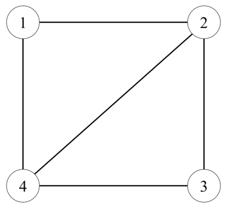
In order to think about propagation processes, we need to lay out the necessary tools to approach the problem.

The first thing to realize is that the propagation of anything on a network is equivalent to a change in the state of a node and/or link. For example, in the process of adoption of innovation, any individual represented by a node has either adopted the innovation or is still pending to adopt it. In information propagation, an individual has either received or is yet to receive the information. And in the context of infectious diseases, an individual is susceptible, infected, or recovered (or removed). Each specific process we focus on may have more than 2 states that need to be modelled (see, e.g. infectious diseases). But the bottom line is that the propagation process is reflected in the way that states for each node change over the course of the process.

The second ingredient to a propagation process involves the rules of propagation, i.e., how a node changes from one state to another and what is the role played by the neighbor nodes of the node in question. Different sorts of rules lead to a variety of outcomes and, more broadly, to classes of outcomes. For example, some of the simpler processes such as epidemic propagation require for an individual to become infected that at least one of its neighbors is infected so that its infection can be passed on; if a node has no infected neighbors, then it cannot become infected. In contrast, adoption of innovation tends to require several neighbors of a node to adopt the new technology in order to "convince" the target node to also adopt the new technology.

Now, there may be other ingredients that are not necessarily present in all models. In some cases, the entire network may also feel the influence of external factors that may affect the propagation process independently of the influence of neighbors. Another mechanism that can be present is network evolution, where some nodes or links may disappear and new ones may join the network.

Let us think of a very simple propagation model on a simple network.



Example 1. *Consider the network shown on the margin. Let us model an epidemic starting from node* 1*. Our epidemic will be very simple: individuals are infected permanently so that once they catch the disease they keep it. Secondly, the transmission rate is 100%, i.e., if a node catches the disease, then its uninfected neighbors are guaranteed to also catch the disease.*

So how do we model this disease? Well, as explained above, we need to define a state for each node. Since we only need to deal with a binary situation (uninfected or infected), we define a vector $\mathbf{x}$ *that indicates the state of each of the nodes in the network. Thus, if node* i *is uninfected,* $x_i = 0$*; if* i *is infected,* $x_i = 1$*. In our example, where patient zero is the node 1, we can write*

$$\mathbf{x} = (1, 0, 0, 0).$$

Implicitly, there is a time variable t *playing a role in this process. We can think of* $\mathbf{x}$ *as the state at time* 0*, or* $\mathbf{x}_0 = (1, 0, 0, 0)$*. Once a step in time has occurred, the state of the system is* $\mathbf{x}_1$*. Subsequent steps are characterized by* $\mathbf{x}_2, \mathbf{x}_3, \ldots$

To update the state of a node i *at time* t*, we have the following equations*

$$x_{i,t} = \theta \left(x_{i,t-1} + \sum_j a_{ij} x_{i,t-1} \right)$$

where $\theta(u)$ is a well-known function called the Heaveside step function, equal to 1 if $u > 0$ and 0 if $u \leq 0$. Thus, $x_i = 1$ at time t if $x_i = 1$ at time $t-1$ and/or if any of its neighbors is 1 is the previous step.

Therefore

$$\mathbf{x}_1 = (1, 1, 0, 1)$$

because from $t = 0$ to $t = 1$, node 1 infects its two neighbors, nodes 2 and 4, and also stays infected itself. In the next step, we have

$$\mathbf{x}_2 = (1, 1, 1, 1)$$

because node 3, which did not become infected in $t = 1$ because it was not directly connected to node 1, is now infected due to either nodes 2 or 4.

After time $t = 2$, all nodes remain infected forever, and therefore we can write the solution to this problem as

$$\begin{aligned} \mathbf{x}_0 &= (1, 0, 0, 0), \\ \mathbf{x}_1 &= (1, 1, 0, 1), \\ \mathbf{x}_t &= (1, 1, 1, 1) \quad (t \geq 2). \end{aligned}$$

This example is actually very useful. It provides us with all the basic ingredients of a propagation process. Nodes have states; the network plays a role; there is a time parameter t; the states at time t depend on the states at time $t-1$ (in more sophisticated models the time dependence can be more complicated). In addition, note that because a node can never recover from being infected, can transfer the infection to its neighbors with probability 1, and the network has only one connected component, the epidemic process ends up with all nodes infected. This is not a result specific to this network, but rather it stems from the rules of the propagation process. This last point is a concrete example of what we mentioned above: the rules of a model determine the class of propagation behavior.

In this sense, we can say that models in which the only states are uninfected and infected, still-to-adopt or adopted, etc. and the rules only provide transitions in one direction, from uninfected to infected, still-to adopt to adopted, etc. the propagation problem always flips all of the network from one state to the other. This result is not affected by the fact that transition probabilities are not equal to 1. If, for instance, we had done the problem in Exmp. 1 by taking a probability 0.1 that a node infects any of its neighbors, the ultimate state of the network is the same: all nodes infected. What changes is how quickly it happens. And what would happen if the probability to infect was equal to 0.000001? The same thing, but taking a much longer time.

These considerations teach us this: the broad characteristics of a propagation model can often be anticipated by a careful analysis of the rules of the model.

Question: before the advent of the chicken pox vaccine, what was the probability that an adult person would answer yes to the question: “have you had chicken pox?”

2 Epidemics: susceptible-infected-removed model

Let us now focus on a basic epidemic disease model, used widely in infectious disease studies. It is called the Susceptible-Infected-Removed (or SIR) model. A network is not required for the model, but it has become commonplace to see versions of it deployed on networks. This is a sensible strategy for many diseases in which a social connection is important for its transmission.

In this model, an individual can be in one of three states: susceptible, infected, or recovered/removed. When applied to a network, these are the three states a node can take at a point in time. The names of the states are rather self-explanatory: a susceptible individual is one who can catch the infectious disease provided the conditions are right, an infected individual is one undergoing the disease, and a recovered/removed individual is one that is no longer infected and cannot catch the illness again.

The dynamics of the model are driven by transitions between each of the states on the basis of some rules. The rules include the probability that an infected individual infects a susceptible one and the time it takes for an infected individual to recover (although this could also be changed into a rate of recovery per time step). As explained above, a susceptible individual needs to be in contact with an infected one to get infected.

The SIR model is an example of so-called compartment(al) models in epidemiology and, more generally, in population dynamics. The states that one assigns to each individual in the system compartmentalizes the population. The rules that allow transitions from one compartment to the other, together with the structure in which the population is organized (say, a network) determine the dynamics of the system.

To develop a concrete understanding of an SIR model, I will discuss one with the following states and rules for transition:

1. An individual in the system (network), can have one of the three states $\mathcal{S}, \mathcal{I}, \mathcal{R}$.

2. The epidemic starts with an initial case, which can be changed to a set of cases, that is arbitrarily assigned the $\mathcal{I}$ state. This case is traditionally called *patient zero* in epidemiology.

3. An infected individual remains infected for a period of time D, the duration of the illness. Simple SIR models like the one we construct here make an infected individual **contagious** from the moment it becomes infected (or typically one time step after infection as we do here), and remains contagious until recovered. The **transmission rate** $\mathcal{T}$ is a number between 0 and 1 which indicates the likelihood in any given time step for an infectious individual to infect a susceptible one when they come in contact (in a static network, two neighbors are always in contact).

4. When an individual transitions from infected to recovered, it no longer infects others.

Specific SIR models can vary in details. In some cases, individuals recover not after a fixed duration D but instead recover with a probability rate after a number of days so that the duration of the illness is not fixed. In other models there may be various ways to go to state $\mathcal{R}$, etc. These variations, however, are such that the behavior of the model remains in the same qualitative class. Stronger changes such as SIS models where people can recover and then be infected again, such as with flu, belong to a different class of model. An SIR model is a good model for illnesses such as measles, chickenpox, the eradicated smallpox, polio, and other illnesses that confer life-long immunity.

For the sake of completeness, note that the way we interpret an SIR model offers some flexibility. While it is natural to think of the infected state as being the time when a person is ill, one could take another view and think of the $\mathcal{I}$ state as the time when a person is contagious. This view can be useful from the standpoint of monitoring the time progression of the contagion of a population, stripping away any other considerations of the state of each individual. However, if one is interested in understanding a contagious disease in the context of other variables such as needs for medical services and consultations, the use of medicines or supplies, time off work, etc. and depending on the specific illness, an SIR model could prove incomplete. Like most situations in research, one has to be very mindful of the *question of interest* before settling on a given model to study. To give another example, it is believed that CODIV-19 should be best modeled with an SEIR model (susceptible, exposed, infected, and recovered) because people can be exposed and have a time between receiving a viral load but before the become infectious; that time period between being exposed and being infected and infectious adds to the overall count of days of illness cycle, and makes contact tracing harder (harder to figure out how one individual may have gotten the infection, the infected-infector contact).

To start a more concrete analysis of SIR models, let us look at the following code for a routine that accepts a network `G`, an infectious period `dur`, a transmission rate `T`, and the identity of patient zero `o`. The outcome of the routine is a set of time series `S`, `EI`, `R`, `I`, and `Rtime`, the total duration of the epidemic captured in the final value of the variable `time`, and a quantity called `Reffavg` that constitutes the average over all individuals of how many susceptibles each infected individual has infected. The time series `S`, `EI`, and `R` track, respectively, the number of susceptible, ever infected (currently or in the past), and recovered individuals at any point in time. The time series `I` corresponds to the number of individuals in the infected state $\mathcal{I}$.

```
def infect(G,o,T,dur):
    import networkx as nx
    import random as rn
    beeninfected={} # Dict, key node, value Boolean
    timeInfected={} # Dic, key node, value time since infected
    for i in G.nodes():
        beeninfected[i]=False # All nodes are uninfected
    initially
        timeInfected[i]=0
    beeninfected[o]=True # Infect node o
    A=[o] # Add o to list of actively infected nodes
    Rtime={} # Time series ratio I[t]/I[t-1]
    Reff={} # Dict, key node, value infected due to node
    EI={} # Time series, ever infected
    S={} # Time series, susceptibles
    R={} # Time series, recovered
    I={} # Time series, currently infected
    time=0 # Time variable, initialized to 0
    EI[time]=1 # Start with one infected
    R[time]=0 # No recovered at start
    S[time]=G.order()-1 # All but one node are susceptible
    I[time]=1 # Start with one infected
    while len(A)>0: # while there are infected nodes, continue
        newA=[] # Actives list for next step
        time=time+1 # Move time forward
# EI, S, R are updated each time step, not counted from scratch
    .
```

```
        EI[time]=EI[time-1] # Update EI
        S[time]=S[time-1] # Update S
        R[time]=R[time-1] # Update R
        for i in A: # Go through each active infected node
            timeInfected[i]=timeInfected[i]+1
            if timeInfected[i]==dur: # Recover at dur
                R[time]=R[time]+1
            elif timeInfected[i]<dur: # Can still infect
                newA.append(i) # Infector may infect in t+1
                I[time]=I.get(time,0)+1 # Current infected
                for j in G.neighbors(i):
                    # only susceptibles can be infected
                    if beeninfected[j]==False:
                    # Infect with probability T
                        if rn.random()<=T:
                            Reff[i]=Reff.get(i,0.)+1
                            beeninfected[j]=True
                            timeInfected[j]=0
                            EI[time]=EI[time]+1
                            S[time]=S[time]-1
                            I[time]=I[time]+1
                            newA.append(j)
        RtimeStep=float(len(newA)-len(A))/len(A)
        Rtime[time]=RtimeStep
        A=newA # Update list of actively infectious
    Reffavg=0. # Effective reproductive rate per node
    for i in Reff.keys():
        Reffavg=Reffavg+Reff[i]
    # Average over Reff to obtain Reffavg
    if len(Reff)==0:
        Reffavg=0.
    else:
        Reffavg=Reffavg/len(Reff)
    # Create output
    OutDic={}
    OutDic['I']=I
    OutDic['EI']=EI
    OutDic['S']=S
    OutDic['R']=R
    OutDic['Rtime']=Rtime
    OutDic['Reffavg']=Reffavg
    OutDic['time']=time
    return(OutDic)
```

To see this code in action, I have run it using the 'ca-CondMat-noself.txt'

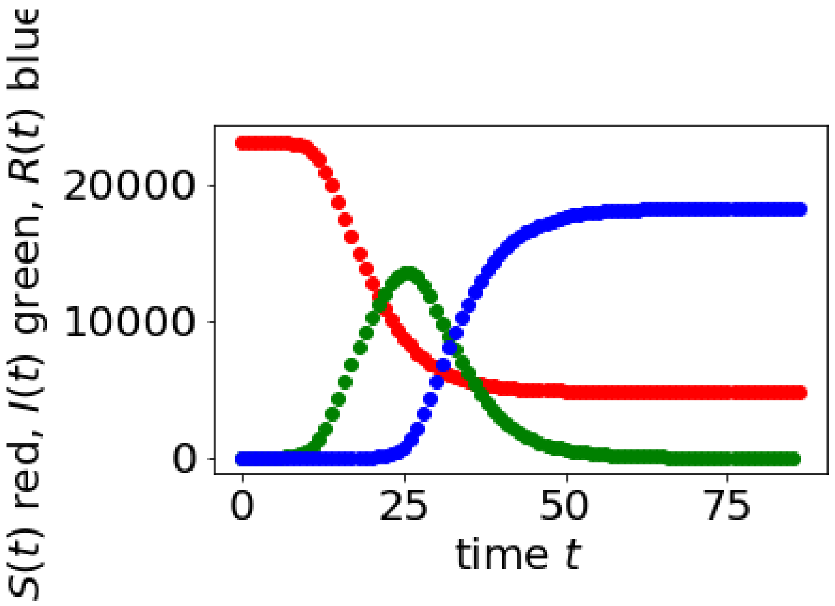


Figure 1: One realization of the SIR model run over the co-authorship network.

and choosing a random node as the starting infection. I use the function `readlist` to import `'ca-CondMat-noself.txt'` into a `networkx` object called `C`. The code is

```
from readlist import readlist
import random as rn
C=readlist('ca-CondMat-noself.txt',4)
Cnodes=list(C.nodes())
rn.shuffle(Cnodes)
EpiC=infect(C,Cnodes[0],0.05,14)
```

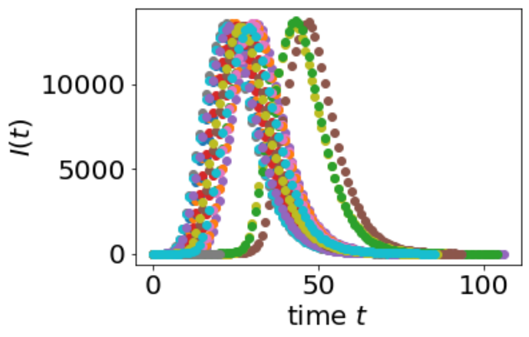
The transmission rate is $\mathcal{T} = 0.05$ and the duration of the illness is $D = 14$. The result of one run if shown in Fig. 1.

The computational SIR model we have presented is stochastic, which means probabilistic. Therefore, even starting with the same node as patient 0 and equal parameters, the outcome of the epidemic need not be the same. Running 20 versions of the same epidemic with the same initial node we obtain various outcomes. The following code produces a set of epidemics all starting from the same randomly chosen node. I save their outcomes and plot them.

```
IR={}
ReffR={}
TotDurR={}
rn.shuffle(Cnodes)
print "patient zero:",Cnodes[0]
for r in range(20):
    Rl=infect(C,Cnodes[0],0.05,14)
    IR[r]=Rl['I']
    ReffR[r]=Rl['Reffavg']
    TotDurR[r]=Rl['time']

for r in IR.keys():
    plt.yscale('log')
    plt.plot(IR[r].keys(),IR[r].values(),'o')
plt.xlabel('time $t$')
plt.ylabel('$I(t)$')
plt.tight_layout()
plt.savefig('SIR-condmat-T005-D14-rea20-log.png')
plt.show()
for r in IR.keys():
    plt.plot(IR[r].keys(),IR[r].values(),'o')
plt.xlabel('time $t$')
plt.ylabel('$I(t)$')
plt.tight_layout()
plt.savefig('SIR-condmat-T005-D14-rea20-lin.png')
plt.show()
plt.plot(ReffR.keys(),ReffR.values(),'o')
plt.xlabel('Realization r')
plt.ylabel('Avg effective rate of infection')
plt.tight_layout()
plt.savefig('SIR-condmat-Reff-T005-D14-rea20.png')
plt.show()
plt.plot(TotDurR.keys(),TotDurR.values(),'o')
plt.xlabel('Realization r')
plt.ylabel('Total epidemic duration')
plt.tight_layout()
plt.savefig('SIR-condmat-EpiDur-T005-D14-rea20.png')
plt.show()
```

In my execution, the random node was `'99746'`. Figures 2 and 3 show the outcomes of my realizations. We can see that indeed even with the same starting node, the trajectories of the epidemics are different. On the other hand, it is important to know that if the epidemic gets to take off (in our case, one realization only marginally gets going with only two individuals



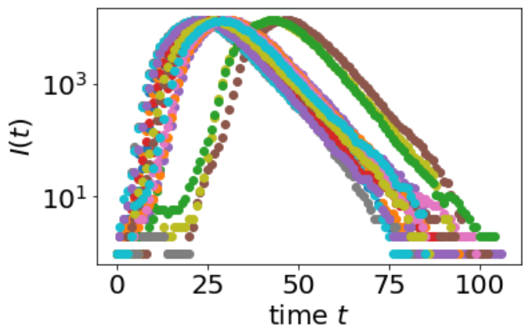


Figure 2: Number of infected at time t, $I(t)$. The top plot is in linear scale, and the bottom plot in logarithmic scale in the vertical axis. The straight shape of the plot vertically for early times denounces the exponential growth phase of the epidemic.

infected altogether), then the trajectories become very similar.

A more extensive study of an epidemic model would need to contemplate a whole spectrum of properties. For example, one should consider how well connected patient zero may be (node i with degree k_i), as well as the local structure of the network for that node. In addition, a full exploration of the effects of $\mathcal{T}$ on the epidemic are critical. Quantities that should be tracked include total number of infected (ever infected), maximum size of the infected (peak of the epidemic), duration, etc. And of the utmost importance, tracking an epidemic requires learning about R_o, the so-called basic reproductive rate of a disease.

R_o has become the most used quantity in understanding epidemic propagation. This is because it allows the researcher to address at least two critical

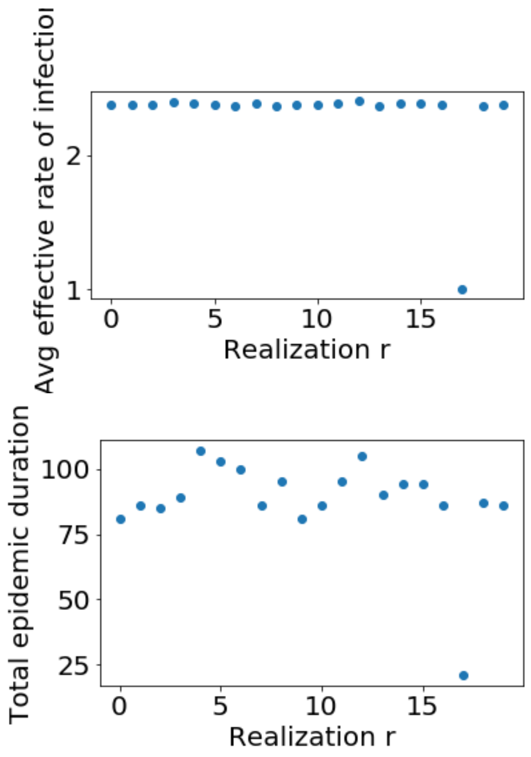


Figure 3: Top: Average effective rate of infection for all individuals in epidemic realization r. Bottom: Total duration in days of epidemic realization r.

aspects of an epidemic, the initial speed of propagation of the epidemic and the percentage of the population that carry the disease if it becomes endemic. Herd immunity (the effect of any given individual being unlikely to become infected because so many others have already reached the $\mathcal{R}$ state) is also correlated to R_o, as well as the peak of the epidemic, but these have other variables involved. In our code, `Reffavg` is one way we track R_o, although the tracking is not over time but rather over individuals, a *local view* of R_o that is not standard. Nevertheless, this approach allows for the study of any plausible relation between k_i and the number of secondary cases due to i.

Our treatment here is quite basic, but it provides a number of needed ingredients that can be built upon to undertake come complex analyses. A very important piece of the puzzle we have ignored is the renewal of susceptibles, simply because we have a fixed rather than an evolving network. If we were to include a dynamic network in which new susceptibles came in, we could also study the pre- and endemic states of an epidemic. The worrying feature of COVID-19 is that it is in the pre-endemic state (not in equilibrium with the population) and thus, it is potentially able to infect more people that what will be the eventual endemic state. The way this matters is the following: prior to the endemic state, a rapidly propagating infectious disease can infect most of the population, and then seemingly disappear, only to re-emerge with the number of susceptibles is large enough for a new propagating phase to come back. But in the initial infection, because so many people get the illness, the full set of complications is seen by a massive population. Currently, this could mean extreme pressure of the health system, beyond what it is able to cope with at both local and global levels.

Some additional thoughts: what role can network properties such as clustering or path length play in an epidemic? Intuitively, one would expect that if a network is highly clustered (in terms of local clustering) then there would be a rapid propagation of illness among those highly related to one another and a slower one across those groups. An interesting exercise to perform is this: two networks with identical n and m, but one more clustered than the other can be compared for a propagating epidemic. How does the clustering affect the propagation?

3 Adoption of innovation

This is another topic in which propagation can take place in a setting such as a network. The theory of innovation is not new at all, but its study in networks is.

At the most fundamental level, adoption is also a propagation process. However, some of the details related to the process are not completely similar to those of epidemics. One key aspect is the fact that rarely does one individual's adoption of an innovation trigger subsequent adoptions of the same innovation by a large fraction of his/her contacts. It is more common that an individual is "convinced" by the fact that many of his/her friends have adopted the innovation and thus the individual follows suit.

A highly cited model of innovation adoption is the one published by Watts [1]. In this model, every individual in a network is assigned a randomly chosen threshold ϕ_i that represents the percentage of the neighbors of node i that need to have adopted an innovation for node i to also adopt it. It turns out that the way the random numbers ϕ_i are drawn (their underlying distribution) "is not very important" (this is clearly an approximate statement). What is important is that ϕ_i changes from node to node. The rule for a node i to adopt the innovation is that

$$\frac{\#\text{ neighbors of } i \text{ that have adopted}}{k_i} \geq \phi_i. \tag{1}$$

To generate this model, I have created the following `python` function

```python
def adopt(G,ol):
    import networkx as nx
    import random as rn
    NodePhi={}
    NodeState={}
    A={}
    time=0
    for i in G.nodes():
        phi=rn.random()
        NodePhi[i]=phi
        NodeState[i]=0
    for o in ol:
        NodeState[o]=1
    A[time]=len(ol)
    # Now execute adoption
    # First, determine if there are nodes satisfying threshold
    check=False
    update=[]
    for i in G.nodes():
        ki=G.degree(i)
        pressi=0.
        for j in G.neighbors(i):
            if NodeState[j]==1:
                pressi=pressi+1
        if (pressi/ki)-NodePhi[i]>0. and NodeState[i]==0:
            update.append(i)
    if len(update)>0:
        check=True
    while check:
        time=time+1
        A[time]=A[time-1]
        for i in update:
            NodeState[i]=1
            A[time]=A[time]+1
        check=False
        update=[]
        for i in G.nodes():
            ki=G.degree(i)
            pressi=0.
            for j in G.neighbors(i):
                if NodeState[j]==1:
                    pressi=pressi+1
            if (pressi/ki)-NodePhi[i]>0. and NodeState[i]==0:
                update.append(i)
        if len(update)>0:
            check=True
    return(A)
```

The algorithm tracks the number of adoptions $\mathcal{A}$ over time t. In fact, there is a bit of idiosyncrasy about time because time progresses only to the extend that there are changes of opinion in the system; when those changes stop, the algorithm also stops along with the time. Time progresses by one unit when all the nodes that have yet to adopt an innovation indeed do so as a consequence of each node having enough neighbors that have already adopted the innovation. Now, whether in one step only a single node changes or many nodes change, the time step increases by one unit. The new nodes that have adopted the innovation change the number of neighbors of the nodes that have not adopted, potentially triggering another wave of adoptions in the next time step.

The adoptions encountered over time, including the overall number of them, can depend upon various details, including the identify of the original

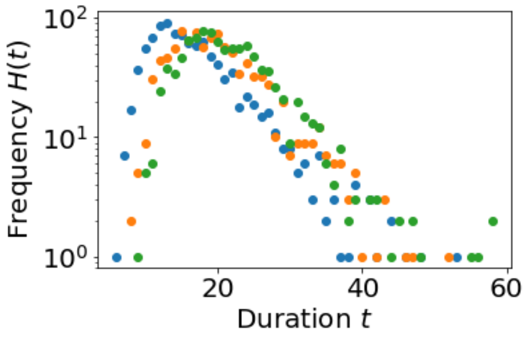


Figure 4: Total times of adoption for $100, 200, 300$ staring nodes.

nodes that adopt, how close they are to one another, how many of them there are, and their degrees.

In Fig. 4 are a number of histograms of final time for 1000 simulations for $100, 200, 300$ initial nodes each.

References

[1] Watts DJ A simple model of global cascades on random networks, PNAS April 30, 2002 99 (9) 5766-5771.

[2]