SUPER AND SUB CRITITCAL CASES APPLIED TO THE BIRTH-DEATH MODEL

JONATHON D'ARCY

Abstract. This report acts as a continuation of section 2 in [1]. In particular it aims to simulate the probability of epidemic sizes in the Birth-Death model under varying conditions. It was found that in the critical version a linear relationship was produced under log-log for the probabilities of epidemic sizes. A generalization for the sub- and super-critical cases was then created with the mean epidemic size plotted. The SIR model was then investigated for the same probability and compared to the Neutral Birth-Death model. We found the SIR model to not share this linear relationship found in the Birth-Death model. Finally, we investigate the theoretical derivation of the probability of epidemic sizes in the Birth-Death model.

1. Introduction

Records of epidemics causing widespread chaos both medically and socially, have been around as early as 430BC. In this time a plague devastated the city of Athens killing an estimated 25% of the population and according to the Athenian General Thucydides,

> *"The catastrophe was so overwhelming that men, not knowing what would happen next to them, became indifferent to every rule of religion or law."*

Today epidemics are becoming less common and when they occur less deadly. A big part of this is due to new medical advances, although this is not the only reason. Another major reason comes from our ability to model epidemics, allowing us to intelligently react in such a way as to minimize the effects it has on society. One of the key metrics we want to understand before reacting is what is the probability of a certain number of people contracting the virus. We seek here to explore this probability in various models both theoretically and via simulation.

2. The Critical Birth-Death Process

An important question in epidemic modelling is around what is the probability, $\mathcal{A}_n$, that a given epidemic will infect n unique bodies. To determine this we first have to choose a method of modelling the epidemic. We begin by observing one of the simplest branching processes, the birth-death process. The process works by an infected body possessing the ability to become cured at a rate α and to infect a new body at rate β. We begin by looking at the critical case of the model, this means that α and β are the same rate. In section 2.3 of the source notes [1], we find the theoretical form of $\mathcal{A}_n$ for the neutral model as $\mathcal{A}_n = \frac{(2n-3)!!}{2^n n!}$. We also are able to use Stirling's approximation to find a slow asymptotic decay form as $A_n \sim n^{-3/2}/\sqrt{4\pi}$. Here we look to verify these results via simulating the

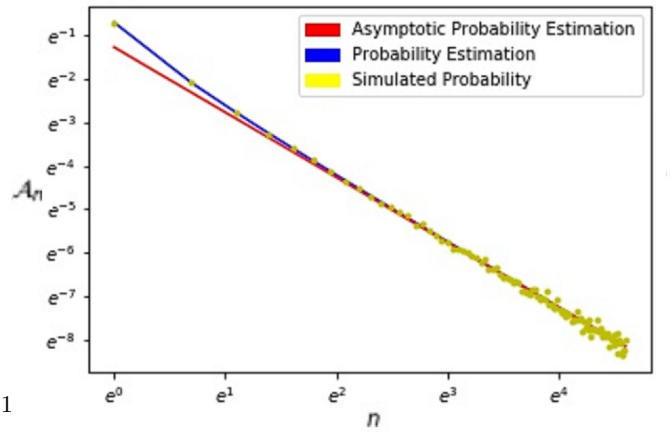

Figure 1. Log-Log Plot of $\mathcal{A}_n$

first 100 values of $\mathcal{A}_n$ using Python. To do this
15,000 critical birth-death processes were created for each n from 1 to 100 and the proportion of these runs that resulted in n unique infected bodies was returned. Along with this the theoretical probability estimation and the asymptotic probability estimation was calculated for each n. In Figure 1 we can see the results of these simulations displayed in a log-log plot. Clearly, the simulated probability follows the theoretical estimation as we'd expect. There may be initial alarm at the noise displayed towards the end of the plot however this is just the variance due to the small probabilities and batch sizes.

3. Super-Critical and Sub-Critical

We now consider the cases where the rate of become cured and the rate of infecting another are not equal. We refer to these cases as super-critical when the rate of infection surpasses that of becoming cured, and sub-critical if the rate of becoming cured surpasses that of becoming infected. To find the value of $\mathcal{A}_n$ theoretically we follow the method found in section 2.3 of [1], and start with the recursion:

$$\mathcal{A}_n = \frac{\alpha}{(\alpha+\beta)} \sum_{i+j=n} \mathcal{A}_i \mathcal{A}_j + \frac{\beta}{(\alpha+\beta)} \delta_{n,1} \tag{1}$$

Introducing $F(s)$ in the same way as section 2.3 gives us,

$$F(s) = \frac{\alpha}{(\alpha+\beta)} F(s)^2 + \frac{\beta}{(\alpha+\beta)} s \tag{2}$$

$$\implies F(s) = \frac{(\alpha+\beta) - \sqrt{(\alpha+\beta)^2 - 4\alpha\beta s}}{2\alpha}. \tag{3}$$

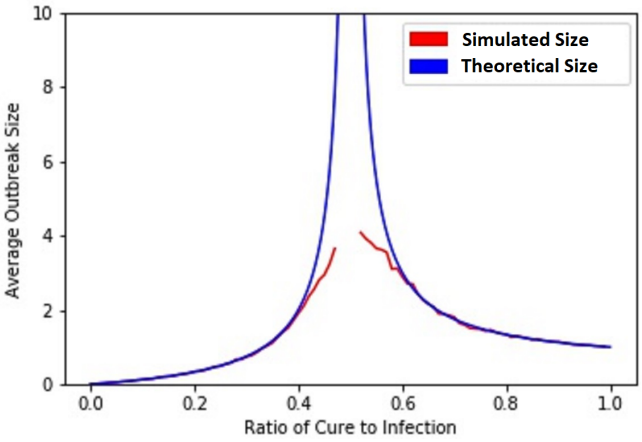

Figure 2. Theoretical and Simulated $\mathbb{E}(N)$

From this we can derive $\mathcal{A}_n$ as,

$$\partial_s^n F(s) = \frac{\beta^n (4\alpha)^{n-1} \prod_{k=1}^{n-1} \frac{2k-1}{2}}{((\alpha+\beta)^2 - 4\alpha\beta s)^{\frac{(2n-1)}{2}}} \implies \mathcal{A}_n = \frac{1}{n!} \partial_s^n F(s)\Big|_{s=0} = \frac{\beta^n (4\alpha)^{n-1} \prod_{k=1}^{n-1} \frac{2k-1}{2}}{n!(\alpha+\beta)^{(2n-1)}}. \tag{4}$$

Another interesting property that we can observe is the expected number of people to be infected. This value denoted $\mathbb{E}(N)$ can be found to be the value of $\partial_s^1 F(s)|_{s=1}$ which we can find using (3) as,

$$\mathbb{E}(N) = \partial_s^1 F(s)\big|_{s=1} = \frac{\beta}{\sqrt{(\alpha+\beta)^2 - 4\alpha\beta}} = \frac{\beta}{\sqrt{(\alpha-\beta)^2}} = \frac{\beta}{|\alpha-\beta|}. \tag{5}$$

We can compare this closed form function to simulations of different ratios of $\alpha : \beta$. This can be seen in Figure 2.

4. The SIR Model

The SIR Model stands for suseptible, infected, and recovered. This acts similarly to the model we previously had investigated with two key differences. The first is that we now look at a finite population of individuals rather than one that is infinite. Secondly, we have three classes of individuals now; susceptible, infected and recovered. To begin, all but one individual who is infected starts as recovered. The infected individual then has the ability to infect the susceptible population at a rate $1/S$, where S denotes the number of susceptible individuals. While this

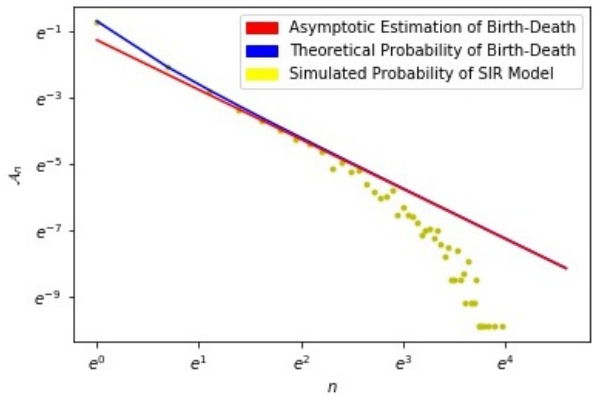

FIGURE 3. Log-Log for $\mathcal{A}_n$ in SIR and neutral BD Model

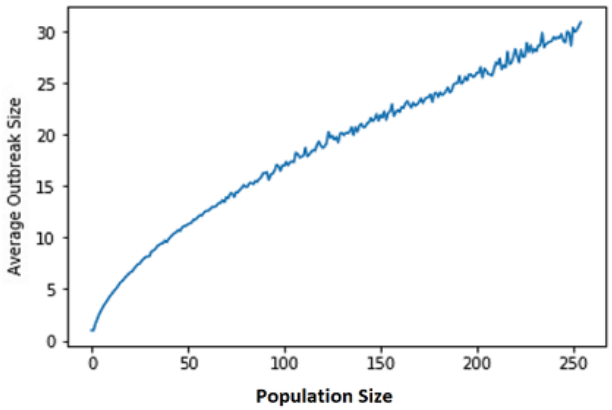

FIGURE 4. $\mathbb{E}(N)$ for SIR Model vs Population Size

happens the infected individuals have the ability to recover at a rate of 1, and once recovered they cannot become infected again. In Figure 3, we simulate $\mathcal{A}_n$ for the SIR model and compare it directly to the previous results of the neutral birth-death model. We notice that the SIR model sees a much stricter drop off in probability of high outbreak situations when compared to the neutral birth-death. In Figure 4, we then simulate the values of $\mathbb{E}(N)$ for varying levels of S. Interestingly, here we see that as the population size increases the mean outbreak size begins to trend almost logarithmically.

5. Deriving $\mathcal{A}_n$ for the Critical Birth-Death Process

We have from [1] in equation (2.12) that $F(s) = 1 - \sqrt{1-s}$. Immediately after we obtain the following result via differentiation:

$$\mathcal{A}_n = \frac{1}{n!}\partial_s^n F(s)\Big|_{s=0} = \frac{(2n-3)!!}{2^n n!} \tag{6}$$

How this is obtained however is not immediately obvious, and in a more personal gripe the notation for double factorials are easily misinterpreted. In this section we aim to derive $\mathcal{A}_n$ while avoiding the double factorial notation. We begin by tackling the meaning behind $F(s)$. This is a generating function for the values of $\mathcal{A}_n$. This means that the coefficents of the Maclaurin Series of $F(s)$ are the values of $\mathcal{A}_n$. We recall that the Maclaurin Series for a given function is of the form $f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(0)}{n!} x^n$, applying this to $F(s)$ and focusing on the coefficients we quickly see,

$$\mathcal{A}_n = \frac{F^{(n)}(0)}{n!} = \frac{1}{n!}\partial_s^n F(s)\Big|_{s=0} \tag{7}$$

To give a closed form of this expansion we begin by calculating a few early differentiations of $F(s)$, this can be seen in Table 1. From here we can recognise that the numerator is the sequence of double factorials of odd parity and the denominator is the sequence of powers of two, which is the form found in the notes. Though to avoid the double factorial, we instead use the fact that $\partial_s^n (s^a) = (a)_n \cdot s^{a-n}$ where $(a)_n$ denotes the falling factorial. We can apply this to $F(s)$, using the

n	1	2	3	4	5	6
$F^{(n)}(s)$	$\frac{1}{2(1-s)^{1/2}}$	$\frac{1}{4(1-s)^{3/2}}$	$\frac{3}{8(1-s)^{5/2}}$	$\frac{15}{16(1-s)^{7/2}}$	$\frac{105}{32(1-s)^{9/2}}$	$\frac{945}{64(1-s)^{11/2}}$
$F^{(n)}(0)$	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{3}{8}$	$\frac{15}{16}$	$\frac{105}{32}$	$\frac{945}{64}$

TABLE 1. Early Differentiation's of $F(s)$

real value extension of the falling factorial, where $(a)_n = \frac{\Gamma(a+1)}{\Gamma(a-n+1)}$, this gives:

$$\partial_s^n F(s) = \partial_s^n \left(1 - \sqrt{1-s}\right) = (-1)^{n+1} \left(\frac{1}{2}\right)_n (1-s)^{(\frac{1}{2}-n)} = (-1)^{n+1} \frac{\Gamma(\frac{3}{2})}{\Gamma(\frac{3}{2}-n)} (1-s)^{(\frac{1}{2}-n)} \tag{8}$$

This allows us to write $\mathcal{A}_n$ as the following:

$$\mathcal{A}_n = \frac{(-1)^{n+1}\Gamma(\frac{3}{2})}{n!\Gamma(\frac{3}{2}-n)} = \frac{(-1)^{n+1}\sqrt{\pi}}{2(n!)\Gamma(\frac{3}{2}-n)} = \frac{(-1)^{n+1}\sqrt{\pi}}{2(n!)\Gamma(\frac{1}{2}-n)(\frac{1}{2}-n)} \tag{9}$$

From here we implement the following relationship,

$$\frac{(-1)^{n-1}\pi}{(\frac{1}{2}-n)} = \Gamma(n-\frac{1}{2})\Gamma(\frac{1}{2}-n). \tag{10}$$

Which when applied gives us the final form of our closed form expression of $\mathcal{A}_n$,

$$\mathcal{A}_n = \frac{(-1)^{n+1}\sqrt{\pi}}{2n!\frac{(-1)^{n-1}\pi}{(\frac{1}{2}-n)\Gamma(n-\frac{1}{2})}(\frac{1}{2}-n)} = \frac{\Gamma(n-\frac{1}{2})}{2n!\sqrt{\pi}}. \tag{11}$$

To verify that this is equivalent to what was found in [1], we convert this to the familiar form. We begin by using,

$$\Gamma\left(\frac{1}{2}+n\right) = \frac{(2n-1)!!\sqrt{\pi}}{2^n}. \tag{12}$$

Applying this to our closed form gives,

$$\mathcal{A}_n = \frac{\Gamma(n+\frac{1}{2})}{2n!\sqrt{\pi}(n-\frac{1}{2})} = \frac{(2n-1)!!}{2^n n!(2n-1)} = \frac{(2n-3)!!}{2^n n!}. \tag{13}$$

6. Conclusion

The strength of the branching models we have seen is their ability for quick implementation to give fast insight to a situation that may be quickly developing. Further, with the ability to gather strict theoretical results from them they become more versatile and accurate. However that is not to say they cannot be improved. Examples of valid extensions to these models could include:

(1) Can we use graphs to represent human to human contact to better represent spread?
(2) Does segmentation of populations affect these models? And if so how does the ratio of segmentation affect them?

References

[1] Lecture notes
[2] *NIST Digital Library of Mathematical Functions*: F. W. J. Olver, A. B. Olde Daalhuis, D. W. Lozier, B. I. Schneider, R. F. Boisvert, C. W. Clark, B. R. Miller, B. V. Saunders, H. S. Cohl, and M. A. McClain, eds.

Python Code

MiAA3

```
"""
Created on Fri Mar  6 15:46:20 2020

@author: jonny
"""

import math, random as rnd, numpy as np, matplotlib.pyplot as plt, ...
   matplotlib.patches as mpatches, matplotlib.ticker as mtick
import statistics
##Q1
def ticks(y, pos):
    return r'$e^{{{:.0f}}}$'.format(np.log(y))

def pandemic_model(probinf,testsize,runs,startsize):
    obs_count = 0

    for run in range(runs):
        numinf = startsize
        obs = startsize
        while numinf>0 and obs≤testsize:
            change=np.random.choice([0,1],p=[probinf,1-probinf])
            if change == 1:
                numinf += 1
                obs += 1
            else:
                numinf += -1
        if obs==testsize:
           obs_count = obs_count+1
    rate=obs_count/runs
    theo=math.gamma(testsize-0.5)/(math.gamma(testsize+1)*
            math.sqrt(4*math.pi))
    theo_est=(1/testsize**(3/2))/math.sqrt(4*math.pi)
    return rate,theo,theo_est
# ============================================================
# ratelist=[]
# timelist=[]
# theolist=[]
# theoestlist=[]
# for i in range(1,100):
#     t=time.time()
#     rate,theo,theo_est= pandemic_model(1/2,i,150000,1)
#     timelist.append(time.time()-t)
#     ratelist.append(rate)
#     theolist.append(theo)
#     theoestlist.append(theo_est)
```

MiAA3

```
# fig,ax1 = plt.subplots()
# ax1.loglog(range(1,100),theolist,'b-',
#            range(1,100),theoestlist,'r-',
#            range(1,100),ratelist,'y.',
#            basex=np.e, basey=np.e)
# ax1.xaxis.set_major_formatter(mtick.FuncFormatter(ticks))
# ax1.yaxis.set_major_formatter(mtick.FuncFormatter(ticks))
# ax1.set(xlabel='$n$', ylabel='$\mathcal{A}_n$', title='Log-Log Plot ...
    of $\mathcal{A}_n$')
# red_patch = mpatches.Patch(color='red', label='Asymptotic Probability ...
    Estimation')
# yellow_patch = mpatches.Patch(color='yellow', label='Simulated ...
    Probability')
# blue_patch = mpatches.Patch(color='blue', label='Probability Estimation')
# plt.legend(handles=[red_patch,blue_patch,yellow_patch])
# plt.show(fig)
# #fig.savefig('Q1.jpg')
# ================================================================
   ##Q2
# ================================================================
# ratelistmain=[]
# theorylistmain=[]
# step=0.001
# step1=0.01
# casestheory=list(np.arange(0,1+step,step))
# cases=list(np.arange(0,0.48,step1))
# cases.extend(list(np.arange(0.52,1,step1)))
# for j in cases:
#     ratelist=[]
#     for i in range(1,50):
#         rate,theo,theo_est= pandemic_model(j,i,5000,1)
#         ratelist.append(rate)
#     index=[a*b for a,b in zip(ratelist,list(range(1,50)))]
#     index=sum(index)
#     ratelistmain.append(index)
# for j in casestheory:
#     theo=(j)/abs(2*j-1)
#     theorylistmain.append(theo)
# fig,ax1 = plt.subplots()
# ax1.plot(cases[0:int(len(cases)/2)], ...
    ratelistmain[0:int(len(cases)/2)],'r',
#          cases[int(len(cases)/2):], ratelistmain[int(len(cases)/2):],'r',
#          casestheory, theorylistmain,'b')
# ax1.set(xlabel='Ratio of Cure to Infection', ylabel='Average Outbreak ...
    Size')
# ax1.set_ylim([0,10])
# plt.show(fig)
# fig.savefig('Q2.jpg')
#
```

MiAA3

```
# ============================================================
    ##Q3
# ============================================================
def SIR_model(testsize,runs,startsize,totalpop):
    obs_count = 0
    for run in range(runs):
        numinf = startsize
        numcure = 0
        numsus= totalpop-startsize
        obs = startsize
        while numinf>0 and obs<=testsize:
            probinf= ...
                (numsus*numinf/totalpop)/(numinf+(numsus*numinf/totalpop))
            change=np.random.choice([0,1],p=[probinf,1-probinf])
            if change == 1:
                numinf += 1
                obs += 1
                numsus += -1
            else:
                numinf += -1
                numcure += 1
        if obs==testsize:
           obs_count = obs_count+1
    rate=obs_count/runs
    theo=math.gamma(testsize-0.5)/(math.gamma(testsize+1)*
                math.sqrt(4*math.pi))
    theo_est=(1/testsize**(3/2))/math.sqrt(4*math.pi)
    print("{}% Done".format(testsize))
    return rate,theo,theo_est

# ================================================================
ratelist=[]
theolist=[]
theoestlist=[]
for i in range(1,100):
     rate,theo,theo_est= SIR_model(i,20000,1,100)
     ratelist.append(rate)
     theolist.append(theo)
     theoestlist.append(theo_est)
```

MiAA3

```
fig,ax1 = plt.subplots()
ax1.loglog(range(1,100),ratelist,'y.',
           range(1,100),theolist,'b-',
           range(1,100),theoestlist,'r-',
           basex=np.e, basey=np.e)
ax1.xaxis.set_major_formatter(mtick.FuncFormatter(ticks))
ax1.yaxis.set_major_formatter(mtick.FuncFormatter(ticks))
ax1.set(xlabel='$n$', ylabel='$\mathcal{A}_n$')
red_patch = mpatches.Patch(color='red', label='Asymptotic Estimation of ...
    Birth-Death')
yellow_patch = mpatches.Patch(color='yellow', label='Simulated ...
    Probability of SIR Model')
blue_patch = mpatches.Patch(color='blue', label='Theoretical ...
    Probability of Birth-Death')
plt.legend(handles=[red_patch,blue_patch,yellow_patch])
plt.show(fig)
# ================================================================
fig.savefig('Q3.jpg')

def SIR_model_size(runs,startsize,totalpop):
    CurrentRuns=[]
    for run in range(runs):
        numinf = startsize
        numcure = 0
        numsus= totalpop-startsize
        obs = startsize
        while numinf>0 and numsus>0:
            probinf= ...
                (numsus*numinf/totalpop)/(numinf+(numsus*numinf/totalpop))
            change=np.random.choice([0,1],p=[probinf,1-probinf])
            if change == 1:
                numinf += 1
                obs += 1
                numsus += -1
            else:
                numinf += -1
                numcure += 1
        CurrentRuns.append(obs)
    print("{}% Done".format(totalpop))
    return CurrentRuns

lis=[]
for i in range(10):
    size=statistics.mean(SIR_model_size(200,1,i))
    lis.append(size)
plt.plot(range(10),lis,'r', xlabel='$Total Population Size$', ...
    ylabel='$\mathbb{E}(N)$')
```