SIR Model Monte Carlo Simulation of Pandemic Flu Spread in a Classroom Setting

Sam Brady

March 11, 2021

Abstract

For this project I will simulate a pandemic flu spreading in a classroom setting, as per the description below, and then answer a few questions pertaining to the number of infected persons and the spread of the distribution on various days. The situation has been simplified as to remove multiple other factors and dependent probabilities, however I will show the complexities of the model and start from it's basic probabilistic foundation and theory behind it's inner mechanisms. This project will consist of Monte Carlo simulations of discrete probability distributions, which I will then use to invoke the law of large numbers in order to estimate and generalize certain conclusions. I will use the SIR model to build a simulation which is a fairly common approach to modeling any kind of flu or disease spread. There are many variations of this model but for this simulation I will stick with the basic SIR components, explained in detail later on. As a former classroom teacher I was curious to see if my own experience could help me draw any conclusions in this simulation, however it only provided me with more curiosity regarding the complexities of the situation and possible next steps to be taken which I will discuss later as well. Aside from the revelations regarding the current simulation, this project did open my eyes to how long weekends, scattered school schedules, along with the recovery rate of a virus can call impact the timeline of a pandemic and should absolutely be taken into account by administrators and planning officials.

Background and Description

This will be a simulation of a pandemic flu spreading in a classroom setting. The classroom will consist of 21 children, one of whom is sick. The flu will be a new strain and thus all children are to be considered healthy and susceptible to the flu on the first day. The sick student will be infectious for a total of three days, and come to school everyday whether he is sick or not. The probability that the infected student infects any other individual susceptible child on any of the three days is $p = 0.02$, and thus all days and children will be independent. If a child becomes infected, he or she will also be infections for three days starting on the next school day.

To build the simulation I chose to work in python and perform my work in a jupyter notebook. There are many advanced statistical packages available, however I chose to work with a few of the more standard packages - mostly numpy.

Additionally I will aim to answer the questions below:

(a) What is the distribution of the number of kids that become infected on Day 1?

(b) What is the expected number of kids that become infected on Day 1?

(c) What is the expected number of kids that are infected by Day 2?

(d) Simulate the number of kids that are infected on Days 1,2,. . . . Do this many times. What are the (estimated) expected numbers of kids that are infected by Day i, i = 1, 2, . . .? Produce a histogram detailing how long the "epidemic" will last.

Independence and Non-Mutual Exclusivity

Although it is clear from the problem that the given probability for a student becoming infected is independent of all other students becoming infected as well as independent from the day, it is less clear on how this independence translates when there is more than one student who is infected. To keep things simple, but also interesting, I will make the assumption that the given probability that the first infected student infects any susceptible student on any day is really the probability of the infected student making infectious contact (contact that would certainly cause a student to become infectious) with a susceptible student. On the first day there really isn't a difference since this is a new flu strain and all students are susceptible. Any infectious contact would then result in infection. However, when there are more than one infected students this distinction is important and allows for the events of two distinct infected students to be regarded as independent. This is a valid assumption to make, in fact many of the infectious disease models I researched used some type of contact rate to model the spread of diseases. The events will then be considered to be non-mutually exclusive as they can both happen (it is very possible for two infected students to make infectious contact with a susceptible student on the same day). In this case it will only result in 1 new infection. Simply put, two infected students cannot get the same susceptible student sick at the same time, but they can both make infectious contact and their probabilities of making infectious contact remain independent. This fact will be important later on when I start building the simulation code and start adding events [1][2].

One Bernoulli Trial

Next I will explore the individual probability that a student infects another student on any given day. This probability will be a crucial building block to the entire simulation. According to Wikipedia [3], a Bernoulli distribution is the discrete probability distribution of a random variable which takes the value of 1 with probability p and 0 with the probability 1-p. In our case the infectious student either infects another student ("1" with probability = 0.02) or doesn't ("0" with a probability = 0.98). The Bernoulli distribution has many unique characteristics which make it valuable to statisticians especially as the building block for other distributions.

$$f(k;p) = \begin{cases} p & k = 1 \\ 1-p & k = 0 \end{cases}$$

$$f(k;0.02) = \begin{cases} 0.02 & k = 1 \\ 0.98 & k = 0 \end{cases}$$

A Special Binomial

Now so far I have only discussed the probability of one student becoming infected, but there are multiple chances for a student to become infected as there are 20 healthy students to start out. At the end of each day we would like to add up all of the successes to determine the number of newly infected students on that day. And luckily for us the probability of any one of the students becoming infected happens to be independent, identically distributed or i.i.d. As it turns out, if we have multiple n Bernoulli trials that are i.i.d. then their sum is distributed according to a Binomial distribution $Bin(n, p)$. Or a better way to think about it is that $Bin(n, p)$ is the number of successes from n Bernoulli trials [4][5]. So in this simulation I will be using the Binomial distribution to model the probability of multiple Bernoulli trials. And if there are multiple days then we can sum up the number of trials and the distribution is still Binomial. This amazing fact will still even hold true for the events of multiple infected individuals as we have preserved independence with the assumption of the probability.

Theorem: If $X_1, ..., X_n \overset{iid}{\sim} Bern(p)$ then $Y \equiv \sum_{i=1}^{n} X_i \sim Bin(n, p)$

Theorem: If $Y_i, ..., Y_n$ are *independent* and $Y_i \sim Bin(n_i, p)$ then $\sum_{i=1}^{k} Y_i \sim Bin\left(\sum_{i=1}^{k} n_i, p\right)$

Poisson Approximation and the Law of Rare Events

Since the individual probability is so low, it is worth pointing out that this simulation qualifies for the conditions of Poisson Approximation. While the events would be the most precisely modeled by the Binomial distribution, the events can be approximated by a Poisson distribution [6]. This approximation works when n (the number of events) is sufficiently large and p (the probability of each event) is sufficiently small. The general rule of thumb is that the Poisson distribution is a good approximation when $n \geq 20$ and $p \leq 0.05$ and an excellent approximation when $n \geq 100$ and $np \leq 10$. In our case it is clear that these values of n and p most certainly qualify for a good approximation. Using the theorem stated above, we can see that n will increase each day and the value for np will stay relatively small. In other words the limit of n is going toward infinity. Don't worry, it doesn't quite get there, but its enough to say that the approximation is getting better each day. This is sometimes called the Law of Rare Events, as the approximation can only be used when the probability of each event is quite small.

Monte Carlo

With a Monte Carlo type simulation, draws of pseudo random numbers can be used to simulate a probability distribution [7]. In our case here, the individual probability that a student infects another student on any given day is p = 0.02. Then for a Monte Carlo simulation I can choose pseudo random numbers on from the Uniform distribution $Uni(0, 1)$ and if that number $U < 0.02$ then we can say the student became infected. For this I will use the python package numpy which I will use to do all of the heavy lifting of generating random numbers. With Monte Carlo simulation I can do this process several times and then invoke the law of large numbers to reveal the estimated distribution and expected values. I will take the averages of several runs of the Monte Carlo simulation to get the average infected children on each day. Some simulations will have more infections than others, some may have none at all, but the law of large numbers will let me estimate on average how many infected students there are each day - wonderful!

$$r(U) = \begin{cases} 1 & U < 0.02 \\ 0 & U >= 0.02 \end{cases}$$

Simulating Bernoulli Trials with numpy

There are many statistical packages in python but none I trust more than numpy. It's simple, but highly effective. Specifically I will be using the *numpy.random.Binomial()* function to generate random samples from a Binomial distribution where n, the number of trials, is set to 1 thus making it a single Bernoulli trial. The output here represents a "0" - or the susceptible student did not become infected.

```
# 1 bernoulli trial for 1 infected person

random.binomial(n=1, p=0.02, size=1)
```

```
array([0])
```

The real reason I chose to work with this function over more advanced statistical tools is the way in which I can generate the numbers, by shape. There is an optional shape parameter which can display each individual Bernoulli trial by three variables m, n and k. For this project I will simply generate a row of values corresponding to the number of susceptible students on each given day, after running the trials some of these values will be 1 but most will be 0 due to the low individual and independent probabilities of 0.02. Here is what it would look like on the first day with 1 infected student and 20 susceptible students. You can see the unlikely outcome that two students have become infected.

```
# 20 bernoulli trials for 1 infected person and 20 susceptible students

random.binomial(n=1, p=0.02, size=20)
```

```
array([0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

While it is beyond the scope of this project the real interesting use case for the shape parameter would be to simulate shapes of seating arrangements in say, a classroom setting with assigned seats or even somewhere like an airplane, and use that to help model the spread.

```
# bernoulli trials for 30 classroom seats at tables

random.binomial(n=1, p=0.02, size=(5, 2, 3))
```

```
array([[[0, 0, 0],
        [0, 0, 0]],

       [[0, 0, 0],
        [0, 0, 0]],

       [[0, 0, 0],
        [0, 1, 0]],

       [[0, 0, 0],
        [0, 0, 0]],

       [[0, 0, 0],
        [0, 0, 0]]])
```

```
# bernoulli trials for 120 airplane seats (just coach)

random.binomial(n=1, p=0.02, size=(2, 3, 20))
```

```
array([[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]],

       [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]])
```

Visualizing the Probability Mass Function with scipy

The probability mass function, or pmf, is the function that will give the probability of a discrete random variable taking a certain value [8]. These probabilities can be very insightful into the distribution of our events. Since the events in this project are modeled by the Binomial distribution, we can use the Binomial distributions pmf to quickly calculate the probability that 0 students become infected, 1 student becomes infected, and at least one student becomes infected, etc.

$$p_X(x_i) = P(X = x_i) = \binom{n}{x} p^x q^{n-x}$$

```
# pmf P(X=1)
binom.pmf(k=1, n=20, p=0.02)
```

```
0.2724930496959568
```

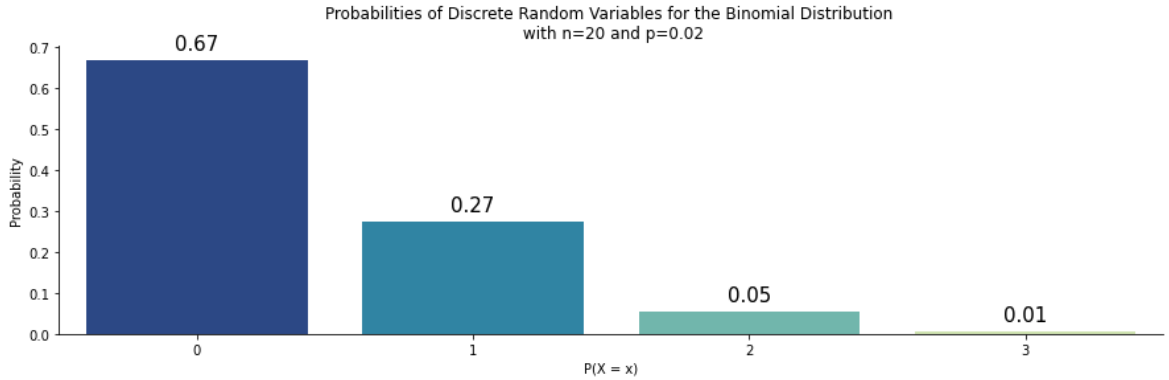
```
# pmf P(X=0)

binom.pmf(k=0, n=20, p=0.02)
```

```
0.6676079717550945
```

```
# pmf P(X!=0)

1 - binom.pmf(k=0, n=20, p=0.02)
```

```
0.33239202824490555
```

The scipy package has a function *binom.pmf()* that will allow us to do exactly that. We can use this function to estimate the probabilities for the first day's events, with 20 susceptible students and one infected student who has an independent probability of infecting each student of 0.02. Here are the probabilities that one student becomes infected, no students become infected, and more than one student becomes infected. This function also makes it easy to visualize the pmf.



Estimating Expected Value with numpy

Although each Bernoulli trial models the event that a susceptible student becomes infected or not, what I am really simulating here is the number of students who become infected each day. The link between the individual probabilities and the number of students who become sick is the expected value of the probability distribution. The expected value can be thought of as the mean of a random variable [9]. So for this project it will be the average number of students who become sick each day. One thing to note is that the values may not be discrete anymore as we are taking averages. To verify my model (a good practice when working with

software someone else wrote) I will simulate multiple trials and calculate the average number of new infections in order to then check this against the formula for the expected value for the Bernoulli and Binomial distributions.

If Random Variable $X \sim Bern(p)$ then $E[X] = p$

Theorem: $Y \sim Bin(n, p)$ implies $E[Y] = E\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} E[X_i] = np$

```
# expected value for a bernoulli trial E[x] = p = 0.02

count = 0.0
n = 1000000

for t in range(0, n):

    curr = np.sum(random.binomial(n=1, p=0.02, size=1))
    count += curr

count/n
```

```
0.020211
```

```
# expected value for a binomial dist or 20 bernoulli trials E[X] = np = 20 * 0.02 = 0.4

count = 0.0
n = 1000000

for t in range(0, n):

    curr = np.sum(random.binomial(n=1, p=0.02, size=20))
    count += curr

count/n
```

```
0.399408
```

SIR Model

The actual model I will be using is a fairly common model for disease and illness simulations. The SIR model is a popular compartmental model in epidemiology and has many variations. The basic model is simple and compartmentalizes a fixed number of people into three categories of being either susceptible (S), infectious (I), or recovered (R), then models their progress through time [10]. Since this project assumes a fixed number of people and discrete probabilities I thought it would be a good choice.

I chose to code my own simulation using python but drew heavy inspiration from the ModSimPy package, which is wonderful but overkill for this simulation. It is primarily set up to handle continuous events. Rather than a robust program I simply wrote a set of pythonic functions which can be easily understood. I will walk through those steps now.

Initializing Parameters

This first mini function simply instantiates the dataframe that will be used in the simulation and populates each column with the initial values on day 0. Since this is a pandemic flu and we are assuming no one has immunity to the disease yet there will be 20 susceptible, 1 infectious, and 0 recovered students as we will see later.

```
def init(S0, I0, R0):
    """
    Initializes SIR dataframe with initial values.

    S0 = number of susceptible people day 0
    I0 = number of infected people day 0
    R0 = number of recovered people day 0

    returns: Initialized dataframe
    """
    intitialize = np.array([[S0, I0, R0]])

    new_df = pd.DataFrame(intitialize , columns=['S', 'I', 'R'])

    return new_df
```

Simulating it One Day at a Time

Next I wrote a function that would simulate one day's worth of discrete time to model the number of new infections that were spread that day. This is really the heart of the project and the information I am out to gather and analyze. This function takes in the number of susceptible people and the number of infectious people on the start of the day and returns the number of new infections at the end of the day based off of the Binomial distribution described earlier in this report. If there are 7 susceptible people and 3 infectious people on a given day then this generates 3 rows of 7 i.i.d. Bernoulli trials each with the same probability. One thing to note is that a susceptible student can only become infected once. For instance, two infectious people cannot get the same person sick. This is all accounted for in the code by simple subtraction, without changing the independence of the events.

```
def new_infections(S, I, p):
    """
    Calculates the number of new infectious contacts per day.
    For mutually inclusive events.

    S = number of susceptible people
    I = number of Infected people
    p = probability of infectious contact

    returns: Total number of new infections
    """
    new_contacts = random.binomial(n=1, p=0.02, size=(I, 1, S))
    total_contacts = int(np.sum(new_contacts))

    count_mult = np.zeros((1, S))
    for i in range(I):
        count_mult += new_contacts[i]

    num_mult = np.sum(np.greater(count_mult, 1))

    new_total_infections = total_contacts - num_mult

    return new_total_infections
```

Updating Values

The next function is the update function which takes this new number of infected people and updates the dataframe of SIR values. This function takes in the number of days to recover from the infectious period, 3 days. The variable t is for time in days, however I may wish to use this differently in the future. As it is now this project assumes the students go to school every day whether it's a weekday or weekend. This works well for the low probability of 0.02 since this simulated pandemic would be over on the first three day weekend. I could have hardcoded the t values into the function, however I am leaving this t variable in there for future simulations where I would like to be able use a school calendar to determine if it is a school day. When the function has done it's job calculating the new SIR values it returns an updated dataframe of values.

```
def update(t, rec, S0, I0, R0, df):
    """
    Updates the dataframe of SIR values, for every t.

    rec = number of days to recover from being infectious
    S0 = number of susceptible people day 0
    I0 = number of infected people day 0
    R0 = number of recovered people day 0
    df = dataframe of SIR values already initialized or started

    returns: Same dataframe with updated SIR values for new t
    """
    s, i, r = df['S'].iloc[t-1], df['I'].iloc[t-1], df['R'].iloc[t-1]

    infected = new_infections(S=s, I=i, p=0.02)

    if t==rec:
        recovered = 1
    elif t > rec:
        recovered = df['S'].iloc[t-(rec+1)] - df['S'].iloc[t-rec]
    else:
        recovered = 0

    s -= infected
    i += infected - recovered
    r += recovered

    if i < 0:
        df.loc[len(df.index)] = [s, 0, r]
    elif s == S0 and i == I0 and r == R0 and t >= rec:
        df.loc[len(df.index)] = [S0, 0, I0 + R0]
    else:
        df.loc[len(df.index)] = [s, i, r]

    return df
```

Putting it All Together

My last function is where it all comes together. This is the function call that would run a simulation or multiple runs of a simulation so I am able to calculate averages and invoke the law of large numbers to generate some estimated expected values. It takes in all the variables from previous sub-functions and has optional metric parameters if you are interested in seeing some statistics for the simulations. The function calls all of the sub-functions and appends them to a dataframe for easy viewing and graphing.

```
def simulation(S0, I0, R0, rec, days, sims, verbose=False, summary=False):
    """
    Runs a SIR simulation and returns a dataframe of SIR values for each day.
    Optional indvidual and summary metrics can be printed.

    S0 = number of susceptible people day 0
    I0 = number of infected people day 0
    R0 = number of recovered people day 0
    rec = number of days to recover from being infectious
    days = number of days to run simulation for
    sims = number of simulations to run
    verbose = if True will print metrics for each simulation
    summary = if True will print metrics for average of all simulations

    returns: Dataframe of SIR values for sims number of simulations of length days
    """
    df = pd.DataFrame()

    length_list = []
    num_sick_list = []

    for s in range(0, sims):

        run = init(S0, I0, R0)

        for t in range(1, days):
            update(t, rec, S0, I0, R0, run)

        df = df.append(run.T, ignore_index=False)

        length = run['I'].idxmin([1, True])
        length_list.append(length)

        n_sick = max(run['S']) - min(run['S'])
        num_sick_list.append(n_sick)

        if verbose == True:
            print("The pandemic flu sim #{} lasted for {} days.".format(s+1, length))

    if summary == True:
        print('\n')
        print('**************************************************')
        print("The average length of a pandemic was {} days.".format(round(sum(length_list)/sims, 2)))
        print("The average number of people who became infected was {}.".format(round(sum(num_sick_list)/sims, 2)))
        print('**************************************************')

    return df
```

Running Many Simulations

Here is the function call for 1000 simulations of 30 day length starting with 20 susceptible students and 1 infected student, where each student has a 0.02 chance of being infected each day and takes 3 days to recover from being infectious. Here is the sample output cut short.

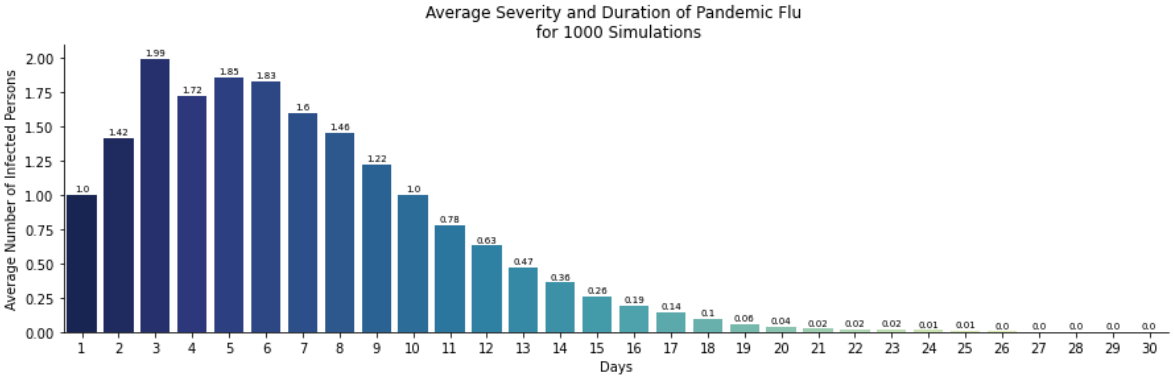
```
# run a simulation(S0, I0, R0, rec, days, sims, verbose=False, summary=False)

sim = simulation(20, 1, 0, 3, 30, 1000, False, True)
```

```
**************************************************************
The average length of a pandemic was 8.32 days.
The average number of people who became infected was 5.03.
**************************************************************
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
S	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
I	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Plotting the Histogram

Here you can see a histogram of 1000 simulations and the estimated average number of infected students on each day.



Findings and Conclusions

First, some things to note would be the dip after day 3 when the first student finally recovers from the pandemic flu. You can also see that no pandemic simulated goes beyond 25 days. One more thing to note would be that the estimated expected value is less than 1 after day 10. Aside from the dip after day 3, I would say that the distribution of the number of infected students appears to be Poisson, which makes sense as this is modeling the number of newly infected students each day, but I believe that the mean rate of arrivals, λ, needs to be constant for this to hold true. One possibility is that it could closely resemble the distribution due to the Poisson Approximation or the Law of Rare Events which was talked about earlier. As the number of Bernoulli trials, n, gets bigger by accumulating each day the Poisson distribution becomes a better approximation and $X \sim Pois(np)$. All of this aside, the biggest take away from this project would be that under no circumstances should any student be required to go to school every day, including weekends with no breaks. Re-running this simulation with weekends breaks would show the virus spread stopping at the first three day weekend. Maybe this is the justification for a four day work week that we've been waiting for!

With the simulation run, and the averages of the number infected people on each day I am finally ready to address the questions proposed at the beginning of this report.

a) What is the distribution of the number of kids that become infected on Day 1?

The distribution is the distribution of 20 independent and identically distributed Bernoulli trials $Bern(p)$ which turns out is a Binomial distribution $Bin(n, p)$.

b) What is the expected number of kids that become infected on Day 1?

The expected number is the expected value of the Binomial Distribution. $E[X] = np = (20)(0.02) = 0.4$

c) What is the expected number of kids that are infected by Day 2?

This is a slightly more challenging question. I will first make the assumption that "by" Day 2 means by the end of Day 2, which would translate to the number of infected students on the start of day 3 (as shown in the dataframe). One interpretation would be that since the expected value of infected students on day 2 is only 0.4 which is less than one person, that we could say there is still only one person infected on that day, thus all of the 20 Bernoulli trials on day 1 and 2 are still independent from one another. Because of this fact we could then invoke the following Theorem to simply sum up the number of Bernoulli trials into an updated Binomial distribution.

Theorem: If $Y_i, ..., Y_n$ are *independent* and $Y_i \sim Bin(n_i, p)$ then $\sum_{i=1}^{k} Y_i \sim Bin\left(\sum_{i=1}^{k} n_i, p\right)$
The expected value on this day then becomes $E[Y_2] = \left(\sum_{i=1}^{k} n_i\right)(p) = (20 + 20)(0.02) = 0.8$

Or another approach would be to use a Theorem which let's us sum expected values whether the random variables are independent or not.

Theorem: $E[X + Y] = E[X] + E[Y]$

Which in this situation would be $E[Y_1 + Y_2] = E[Y_1] + E[Y_2] = (n_1)(p) + (n_2)(p) = (20)(0.02) + (20)(0.02) = 0.4 + 0.4 = 0.8$

And another would be to use the LOTUS method to show that $E[2X] = \sum_x 2f(x) = (2)(np) = 2(20)(0.02) = 0.8$

Yet another approach would be to look at the estimated expected values from the simulation in the histogram. Which turns out to be 0.99, or 1.99 (1 + 0.99) if you count the first infected student. This number being higher would be an indicator that I need to run more than 1000 simulations.

d) Simulate the number of kids that are infected on Days 1,2,. . . . Do this many times. What are the (estimated) expected numbers of kids that are infected by Day i, i = 1, 2, . . .? Produce a histogram detailing how long the "epidemic" will last.

You can see my histogram above detailing the estimated expected number of kids that are infected each day, the values are displayed above the bars.

Next Steps

As I've already mentioned I would love to incorporate the school calendar into the simulation and also rather than have a fixed infection rate possibly research a distribution of common flu infection rates to use. I could see a lot of opportunity to incorporate Bayesian priors to generate some interesting results as there most certainly exists plenty of data for flu spread. Additionally I would love to incorporate preventative measures such as hand washing and immunization to see the effect on the simulation.

References

[1] enwiki:1000601863, author = "Wikipedia contributors", title = "Bernoulli distribution — Wikipedia, The Free Encyclopedia", year = "2021", howpublished = "https://en.wikipedia.org/w/index.php?title=Bernoulli_distribution&oldid=1000601863", note = "[Online; accessed 10-March-2021]"

[2] enwiki:1008034863, author = "Wikipedia contributors", title = "Independence (probability theory) — Wikipedia, The Free Encyclopedia", year = "2021", howpublished = "https://en.wikipedia.org/w/index.php?title=Independence_(probability_theory)&oldid=1008034863", note = "[Online; accessed 10-March-2021]"

[3] enwiki:1005387532, author = "Wikipedia contributors", title = "Mutual exclusivity — Wikipedia, The Free Encyclopedia", year = "2021", howpublished = "https://en.wikipedia.org/w/index.php?title=Mutual_exclusivity&oldid=1005387532", note = "[Online; accessed 10-March-2021]"

[4] D.Goldsman, P. Goldsman, *A First Course in Probability and Statistics*, David Goldsman and Paul Goldsman, December 18, 2020

[5] enwiki:1009678985, author = "Wikipedia contributors", title = "Binomial distribution — Wikipedia, The Free Encyclopedia", year = "2021", howpublished = "https://en.wikipedia.org/w/index.php?title=Binomial_distribution&oldid=1009678985", note = "[Online; accessed 10-March-2021]"

[6] enwiki:1010813792, author = "Wikipedia contributors", title = "Poisson distribution — Wikipedia, The Free Encyclopedia", year = "2021", howpublished = "https://en.wikipedia.org/w/index.php?title=Poisson_distribution&oldid=1010813792", note = "[Online; accessed 10-March-2021]"

[7] enwiki:1006343539, author = "Wikipedia contributors", title = "Monte Carlo method — Wikipedia, The Free Encyclopedia", year = "2021", howpublished = "https://en.wikipedia.org/w/index.php?title=Monte_Carlo_method&oldid=1006343539", note = "[Online; accessed 10-March-2021]"

[8] enwiki:993368279, author = "Wikipedia contributors", title = "Probability mass function — Wikipedia, The Free Encyclopedia", year = "2020", howpublished = "https://en.wikipedia.org/w/index.php?title=Probability_mass_function&oldid=993368279", note = "[Online; accessed 10-March-2021]"

[9] enwiki:1009309347, author = "Wikipedia contributors", title = "Expected value — Wikipedia, The Free Encyclopedia", year = "2021", howpublished = "https://en.wikipedia.org/w/index.php?title=Expected_value&oldid=1009309347", note = "[Online; accessed 10-March-2021]"

[10] enwiki:1009414539, author = "Wikipedia contributors", title = "Compartmental models in epidemiology — Wikipedia, The Free Encyclopedia", year = "2021", howpublished = "https://en.wikipedia.org/w/index.php?title=Compartmental_models_in_epidemiology&oldid=1009414539", note = "[Online; accessed 10-March-2021]"