Pandemic Flu Spread (Q4)

Team BMK: Kevin Cooper, Brent Radcliffe, Matteo Zullo

Abstract

This project looks at the spread of the flu within a congregate setting given a single initial infection vector and a probability, *p=0.02*, of infection. We also know that every infected individual is infectious for three days and that post-infection, the child is immune and no longer a potential vector for infection. Appendix A shows that the probability of at least one infection after day 1 was very high (67%) with an expected value of 1.4. By day 2, the expected value of newly infected increases to 1.9. Lastly, based on a simulation of 10,000 runs, the pandemic is very likely (>30%) to last three days due to the low probability of infection from the initial vector. There is a small spike at day 6 (~7%) that happens when only one kid is infected. Outcomes between 7 and 13 days are all similarly likely with a probability of ~5%. The likelihood of an outbreak that lasts longer than that is very unlikely as can be seen in Appendix B.

Background and Problem Description

Infectious disease spread is obviously an area of intense interest right now. This problem looks at a hypothetical scenario in which a child enters a classroom with 20 other children while infected with the flu. The initial vector is infectious for 3 days and has a probability of *p=0.02* of infecting another child. Additionally, we know that all kids and days are independent so that we have independent and identically distributed (i.i.d) Bernoulli(p=0.02) trials. Any child that is infected will also be infectious for 3 days and will no longer be a potential vector for infection upon recovery.

Our report will focus on three results:

- The distribution of the number of kids infected on day 1 along with the expected value of kids infected on day 1.
- The expected number of kids infected on day 2.
- The number of kids infected on subsequent days and the expected duration of the epidemic.

We wrote up a Jupyter notebook available here defining several functions in basic Python. They allow for flexible replication of the epidemic with a default of 10,000-run simulations. The most important function is *update_kids()* which takes a dictionary and an integer as parameters holding the “classroom” definition at any points and the number of infected kids at the beginning of the day. The “state of the classroom” at any point is defined by 21 kids each assigned with a binary health status (0 when healthy and 1 when sick) and a counter for the days elapsed since the initial infection. This counter is iterated upon until three days post-infection -- i.e. when “counter” is equal to 4. The function loops through the healthy

kids and for each of those kids, realizes as many Bernoulli(p=0.02) trials as there are infectious kids (i.e., kids who have been infected and are within three days of the initial infection). The code uses the Python's module *random* to generate PRNs for the Bernoulli trials.

Main Findings

We found that the probability of one infection on day 1 was very high (67%) and an approximately one in four chance of two infections on day 1. The empirical distribution of infected kids on day 1 for 10,000 simulations is shown in Appendix A while a detailed breakdown is reported in the code snippet below. The functional form for the empirical PDF can be obtained by calculating the probability of each outcome or number of occurrences divided by 10,000. This would be a categorical variable or generalized Bernoulli where the outcome x = 0 has probability 0 and outcomes [1, …, x_i, ... , 21] have probability p_i. The cap is set at 21 because that is the maximum number of infected students in the (arguably far-fetched) scenario where the initial infection vector succeeds at infecting all other kids on day 1. While this outcome has a non-zero probability analytically, it did return a zero probability in our simulation. As the number of runs grows larger, the outcomes start to populate.

$$X = \begin{cases} 0, & \text{if } x = 0 \\ p_1, & \text{if } x = 1 \\ \cdots, & \cdots \\ p_i, & \text{if } x = x_i \\ \cdots, & \cdots \\ p_{21}, & x = 21 \\ 0, & otherwise \end{cases}$$

```
Here's the event set with associated probabilities:
p = 0, if x = 0
p = 1, if x = 0.6697
p = 2, if x = 0.2695
p = 3, if x = 0.053
p = 4, if x = 0.0074
p = 5, if x = 0.0004
p = 0, otherwise
```

Formally, the expected value is given by the weighted occurrence of each outcome in the event set with weights for the probability of the outcome happening.

$$E[X] = \sum_i x_i p_i$$

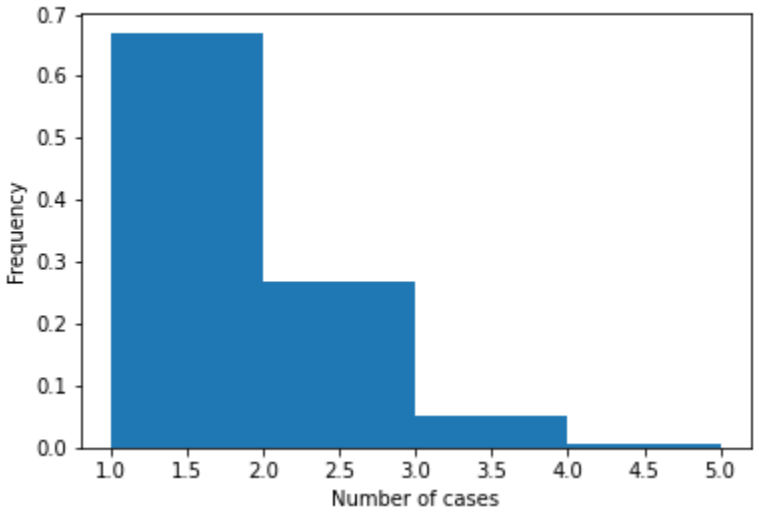
The expected value of the number of kids that Tommy, the initial infection vector, infects on day 1 is 1.4. By day 2, the expected number of infected kids is 1.9. Note that on day 2 no infected child will have fully recovered given that any vector will be infectious for 3 days. The expected number of days that the pandemic will last is the number of days until there are no infected kids. Appendix B is a histogram that shows the timespan distribution of the expected number of cases by day from 10,000 simulated runs. The pandemic is very likely (>30%) to last 3 days because the probability of the initial vector infecting others is very low (*p=0.02*). Another peak is seen at 6 days, which happens when only one kid is infected. Outcomes between 7 and 13 days are all similarly likely (~5%), and epidemics that extend beyond 13 days are far less likely as can be seen from the rapidly declining probabilities beyond day 13.

Conclusion

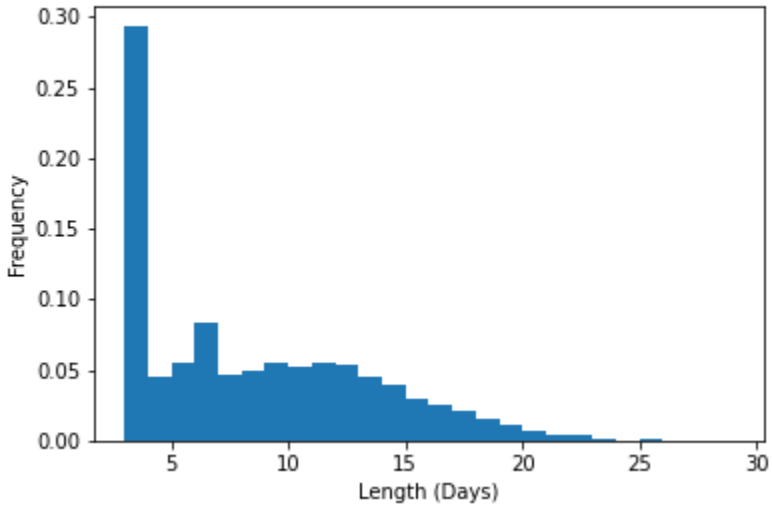
Our project shows that even from a relatively low baseline probability of initial infection (*p=0.02*) the likelihood of at least one other infection is quite high and by far the most likely outcome. However, because of the low probability of infection, the pandemic is also very likely to come to a close fairly quickly. An interesting area for further research would be to look into how the simulated outcomes change with more virulent diseases with higher probabilities of infection. Also importantly, the simplifying assumptions for this example do not contemplate reinfections nor do they parametrize the Bernoulli trials with some measure of proximity. Arguably, kids who spend more time with vectors of infection or sit closer to them might have higher chances of getting infected. Lastly, parents might pull their kids out of school with some probability when symptoms are severe, which could also be studied and modelled.

Appendix

Appendix A



Appendix B



Appendix C

Main

```
import random
import matplotlib.pyplot as plt
import numpy as np
```

```
from collections import defaultdict
from google.colab import files


def initialize_kids():
   """ The function initializes the classroom with kid_0 as infected."""
   kids = ['kid_0'] + ['kid_' + str(i) for i in range(1, 21)]  # kid tag
   sick = [1] + [0 for i in range(1, 21)]  # infected yes/no
   counter = [1] + [0 for i in range(1, 21)]  # length of infection
   return {'kids': kids, 'sick': sick, 'counter': counter}  # output
dictionary


def count_infected(probs: list):
   """ The function returns the number of infected kids"""
   return sum(probs)


def update_kids(kids: dict, n_infected: int):
   """ The function updates the health status of kids."""

   # Begin of day: Update status of healthy kids.
   for i in range(21):  # loop through kids

     # update sick kids
     if kids['counter'][i] > 0:
       kids['counter'][i] += 1

     # update healthy kids
     if kids['counter'][i] == 0:
       count = 0
       while count < n_infected:  # n independent Bern(0.02)
         p = random.random()  # generate random Unif(0,1)
         if (p <= 0.02): # probability of infection
           kids['sick'][i] = 1  # update status to sick
           kids['counter'][i] += 1  # add 1 to counter
         count += 1

   # End of day: Update status of recovered kids.
   for i in range(21):
```

```
      if kids['counter'][i] >= 4:
        kids['sick'][i] = 0
    return kids


def simulate_day1(no_runs = 10000):
 """ The function simulates day-1 spread."""
 output = []
 run = 0

 while run < no_runs:
   # initialize classroom
   kids_0 = initialize_kids()
   n_0 = count_infected(kids_0['sick'])
   # day 1
   kids_1 = update_kids(kids_0, n_0)
   no_1 = count_infected(kids_1['sick'])
   output.append(no_1)
   run += 1
 return output


def simulate_day2(no_runs = 10000):
 """ The function simulates day-2 spread."""
 output = []
 run = 0

 while run < no_runs:
   # initialize classroom
   kids_0 = initialize_kids()
   n_0 = count_infected(kids_0['sick'])
   # day 1
   kids_1 = update_kids(kids_0, n_0)
   no_1 = count_infected(kids_1['sick'])
   # day 2
   kids_2 = update_kids(kids_1, no_1)
   no_2 = count_infected(kids_2['sick'])
   output.append(no_2)
   run += 1
 return output
```

```
def simulate_epidemic(no_runs = 10000):
 """ The function simulates an epidemic."""

 length = []
 days = defaultdict(list)
 run = 0

 while run < no_runs:
    # initialize classroom
   kids__ = initialize_kids()
   no__ = count_infected(kids__['sick'])
   no_day = 0
   # stop epidemic when everyone's healthy again
   while no__ > 0:
     kids__ = update_kids(kids__, no__)
     no__ = count_infected(kids__['sick'])
     no_day += 1  # add to counter
     days[str(no_day)].append(no__)

   # save output
   length.append(no_day)
   run += 1
  # fill in 0s for missing days (epidemic ends at different times but
arrays must have equal length!)
 for day, values in days.items():
   days[day] = values + [0 for i in range(no_runs - len(values))]

 return length, days


def expected_value(cases: list):
 return sum([cases.count(x_i)/len(cases)*x_i for x_i in set(cases)])


def plot_cases(cases: list, xlab :str, file_name: str):
 """ The function returns the pdf of number of cases."""
 plt.hist(cases, density=True, bins = list(set(cases)))
 plt.ylabel('Frequency')
```

```
 plt.xlabel(xlab)
 plt.savefig(file_name)
 files.download(file_name)
 return plt.show()
```

Part A:

```
day_1 = simulate_day1()  # day 1 simulation
plot_cases(day_1, "Number of cases", "day_1_hist.png")  # plot PDF

print("\n")
print("Here's the event set with associated probabilities:")
print("p = 0, if x = 0")
for x_i in set(day_1):
 p_i = day_1.count(x_i)/len(day_1)
 print("p = {}, if x = {}".format(x_i, p_i))
print("p = 0, otherwise")
```

Part B:

```
E_X = expected_value(day_1)
print("The expected value is {}.".format(E_X))
```

Part C:

```
day_2 = simulate_day2()  # day 1 simulation
E_X = expected_value(day_2)
print("The expected value is {}.".format(E_X))
```

Part D:

```
length, days = simulate_epidemic()
plot_cases(length, "Length (Days)", "pandemic_len.png")  # plot PDF

print("Expected values for each day:")
for day, values in days.items():
 E_X = expected_value(values)
 print("E[x={}] = {}".format(day, E_X))
```