Team Los Angeles - Epidemic Report

Divyanshi Kamra, Naman Agrawal, Prasann Viswanathan, Tushar Nandy

April 11, 2020

Abstract

So, we're making a big leap here by using LaTeX to prepare the report on our study. We're keeping it as scientific and realistic as our inexperienced selves can; plus we're making a point to mention 'every' assumption made and source referred for our data. To say that this was a difficult project would be to put it lightly. I can speak on behalf of our team when I say we've abused Sanju for this xD. Notwithstanding, it was an immense learning experience and we can proudly call ourselves COVID-19 experts by now. I hope you enjoy our work :)

1 Breaking it Into Parts

As far as we could see it, the assignment had a few major problems that we needed to handle. Hence we decided to, you guessed it; "break it into parts"

1. We have to scale down the population somehow because no-one can work with a 1.34 billion sized array.

2. We have have to correspondingly scale down the size of the nation and express it as a grid layout to enable proper 10m range condition testing without having to make assumptions

3. We have to distribute the grid into state-wise territories and take care to distribute the population in ratio of the state shares

4. Finally, we must run the two conditions (a) and (b) and find a fool-proof way to extract data of the top 5 most affected states and check whether the proposed travel rate limits the spread within the hospital limit.

Following this will be the detailed description of how each issue was solved, as detailed as we could possible make it; along with specific mentions of cites used and assumptions made.

2 SIR model and Scaling Population down

As we scratched our heads over the dilemma of our nation's insane population, hour of research provided a clear solution. The SIR epidemic model. (1) We made the assumption that at such a vast population scale of 1.34 billion, we could convert the discrete distribution to roughly a continuous distribution hence making the issue of plotting extremely simple through python. (2) As a result of this source and with a little tweaking we created a code to model a new india with the number of infected scaled down by a factor of 10; thus implying 1 infected per state and UT instead of 10; Correspondingly we found the new required population to be 100,000 instead of 1.34 billion.

```
from scipy.integrate import odeint
import matplotlib.pyplot as plt
import numpy as np

# Total population, N.
N = 100000
# Initial number of infected and recovered individuals, I0 and R0.
I0, R0 = 38, 0
# Everyone else, S0, is susceptible to infection initially.
S0 = N - I0 - R0
# Contact rate, beta, and mean recovery rate, gamma, (in 1/days).
beta, gamma = 0.220558, 1./7
# A grid of time points (in days)
t = np.linspace(0, 1000, 1000)

# The SIR model differential equations.
def deriv(y, t, N, beta, gamma):
```

```
    S, I, R = y
    dSdt = -beta * S * I / N
    dIdt = beta * S * I / N - gamma * I
    dRdt = gamma * I
    return dSdt, dIdt, dRdt

# Initial conditions vector
y0 = S0, I0, R0
# Integrate the SIR equations over the time grid, t.
ret = odeint(deriv, y0, t, args=(N, beta, gamma))
S, I, R = ret.T

# Plot the data on three separate curves for S(t), I(t) and R(t)
fig = plt.figure(facecolor='w')
ax = fig.add_subplot(111)
ax.plot(t, S, 'b', alpha=0.5, lw=2, label='Susceptible')
ax.plot(t, I, 'r', alpha=0.5, lw=2, label='Infected')
ax.plot(t, R, 'g', alpha=0.5, lw=2, label='Recovered with immunity')
ax.set_xlabel('Time /days')
ax.set_ylabel('Number')
ax.yaxis.set_tick_params(length=0)
ax.xaxis.set_tick_params(length=0)
ax.grid(b=True, which='major', c='w', lw=2, ls='-')
legend = ax.legend()
legend.get_frame().set_alpha(0.5)
for spine in ('top', 'right', 'bottom', 'left'):
    ax.spines[spine].set_visible(False)
plt.show()

for i in range(600):
  if(I[i]<1):
    print(i)
    break;

#to find the days needed for infection to die out

print(R[i]/N, S[i]/N)
```

```
#to assess the ratio of recovered and uninfected people remaining

#On rerunning code with various values of N initial,
#in order to observe population scaling down characteristics
#we observe that ratio remains unaffected (astonishing!)
#to maintain the timelines integrity,
#we can reduce population to 100,000
#provided we scale the time axis by 2
```

3 Scaling the Areas and Compiling Data

Thanks to our handy SIR model gimmick; we had scaled India's population down; now to scale down the area. We have decided on a grid model hence properly involving the 10m radius of infection criterion without making nasty assumptions. In order to maintain the required population densities it was reasonable to assume that area would be scaled by the same proportion as population. Okay here goes nothing. Don't judge me for the references I'm gonna cite. At the least, try not to. (3), (4)
Next came travel rates between the states; (As I'm writing this i highly doubt whether it was successfully implemented nevertheless here goes nothing) We've made an assumption here; as it turns out the influx and out-flux of populace through states is not documented anywhere! I tried "i'm feeling lucky" although i wasn't feeling lucky and still found zilch. Hence we made the assumption that state to state travel should be in ratios of flight travels. Simple solution to a complicated problem. Will the results vary from actuality? of-course? Do we care? Well we don't seem to have any choice, do we? (5) That was pretty much it on the data compiling side; we've assumed the probabilities of leaving state A and going to state B to be independent; hence implying A to B is Po(A) x Pi(B). Hope it works, fingers crossed. You should be checking our data compiled on the XL sheet about now.

4 Testing Out for a Single State

Grappling with the confusion of how to tackle multiple states with our limited coding assets, we decided to code for our Beloved Goa and were astonished with the results; Turns out the scarce population density and 10m infection

range was enough to curtail the spread of virus abc tremendously. We placed humans randomly on the grid the size of goa and allowed them to do their magic but the infection did not spread? (scratches heads)
Here's the code of our success(/failure?)

```
from random import *

'''legend for health status:
    0: healthy
    1: recovered
    2: dead
    3: hospitalised
    4: infected but not showing symptoms
'''
def close_enough(s, t):
    sq = (s[0]-t[0])**2 + (s[1]-t[1])**2
    distance = sq**0.5
    if distance <= 10.0:
        return True
    else:
        return False


population = [] # a !D array of humans
x = 1250   # x * y = total area of the state
y = 2000    # it is a random distribution
pop = 5000 # total population

health_status = 0 # index of health status

for i in range(pop):
    a = [0, [randint(1, x), randint(1, y)], 0 , 0] # each element has 3\\ #attrib
    for m in range (1,13):  #predefine day of death
        c = random()
        if c <=0.05:
            break          # probability of dying
    if m<12:
        a[3]=m
```

```python
    elif c<=0.05:
        a[3]=12
    population.append(a[:])

I0 = sample(list(range(pop)), 10)       # randomly infecting 10 people
for i in range(10):
    random_guy = population[I0[i]]
    infected = 4
    random_guy[health_status] = infected

days = 5        # choose days for simulation

while True:


    all_done = True

    for i in range(pop):
        a = population[i]     # this where I am checking
                                # and updating health status
        if a[health_status] > 2: # if human is sick
            all_done = False
            a[2] += 1               # first, we increase the days
            if a[2]==a[3]:
                a[health_status] = 2
            elif a[2] == 5:         # if sick guy is on day5, hospitalise
                a[health_status] = 3
            elif a[2] == 12:         # if human survives till day7, recover
                a[health_status] = 1
    if all_done:
        break

    indices = sample(list(range(pop)), 20)   # choosing 20 random moves
    for i in range(20):
        a = population[indices[i]]              # movement is denoted as change
        a[1] = [randint(1, x), randint(1, y)]    # in x and y coordinates
```

```
    for i in range(pop-1):                          # now, everybody in 10m radius of
        if population[i][0] != 4:                   # sick people will get infected
            continue
        else:
            for j in range(1, pop):
                v = population[i]
                t = population[j]
                if t[health_status] == 0 and close_enough(v[1], t[1]):
                    t[health_status] = 4


    fine = 0
    deadcount = 0
    immune = 0
    sick = 0
    hospitalised = 0
    for i in range(pop):
        person = population[i]

        if person[health_status] == 2:
            deadcount += 1
        elif person[health_status] ==1:
            immune += 1
        elif person[health_status] == 0:
            fine += 1
        elif person[health_status] == 3:
            hospitalised +=1
        else:
            sick += 1

print(f"dead: {deadcount}")
print(f"immune: {immune}")
print(f"sick: {sick}")
print(f"fine: {fine}")
print(f"admitted: {hospitalised}")
print('')
```

5 Finale Code

(Fingers crossed, didn't get the time to run it but most probably it will work; Definitely running it tomorrow and making changes to this pdf xD)

```
import csv

f=open('AreaExpressedInXY(1).csv', 'r')
reader=csv.reader(f)
XYpop=[]

for row in reader:
  XYpop.append([int(row[2]), int(row[3]), int(row[4])])

from random import *

'''legend for health status:
    0: healthy
    1: recovered
    2: dead
    3: hospitalised
    4: infected but not showing symptoms
'''
def close_enough(s, t):
    sq = (s[0]-t[0])**2 + (s[1]-t[1])**2
    distance = sq**0.5
    if distance <= 10.0:
        return True
    else:
        return False



nation_info = XYpop[:]
nation = []

for i in nation_info:
    population = [] # a !D array of humans
```

```
    l = i[0]   # x * y = total area of the state
    b = i[1]    # it is a random distribution
    pop = i[2] # total population

    health_status = 0 # index of health status

    for j in range(pop):
        a = [0, [randint(1, l), randint(1, b)], 0 , 0] # each element has 4 attri
                                                        # days spent in suffering,
        for m in range (1,13):  #predefine day of death
            c = random()
            if c <=0.05:
                break          # probability of dying
        if m<12:
            a[3]=m
        elif c<=0.05:
            a[3]=12
        population.append(a[:])

    I0 = sample(list(range(pop)), 10)       # randomly infecting 10 people
    for i in range(10):
        random_guy = population[I0[i]]
        infected = 4
        random_guy[health_status] = infected


    nation.append(population[:])

days = 5        # we can choose days for simulation
                # By default, this code runs till all infected either die or rec

for s in len(nation_info):
    population = nation[s]
    x = nation_info[s][0]   # x * y = total area of the state
    y = nation_info[s][1]    # it is a random distribution
    pop = nation_info[s][2]

    while True:
```

```
all_done = True

for i in range(pop):
    a = population[i]      # this where I am checking
                            # and updating health status
    if a[health_status] > 2: # if human is sick
        all_done = False

        a[2] += 1                # first, we increase the days
        if a[2]==a[3]:
            a[health_status] = 2
        elif a[2] == 5:          # if sick guy is on day5, hospitalise
            a[health_status] = 3
        elif a[2] == 12:          # if he survives till day12, recover
            a[health_status] = 1

if all_done:
    break

indices = sample(list(range(pop)), 24)
# choosing 24 random people to move, last 4 people leave the state

for i in range(20):
    a = population[indices[i]]             # internal movement
    a[1] = [randint(1, x), randint(1, y)]

for k in range(20, 24):
    b = population.pop(indices[k])
    next_state = randint(1, 38)

    new_x = nation[next_state][10][1][0] + nation[next_state][90][1][0]
    new_x /= 2

    new_y = nation[next_state][14][1][0] + nation[next_state][70][1][1]
    new_y /= 2

    b[1] = [new_x, new_y]
    nation[next_state].append(b)
```

```python
    for i in range(pop-1):                          # now, everybody in 10m radiu
        if population[i][0] != 4:                   # sick people will get infect
            continue
        else:
            for j in range(1, pop):
                v = population[i]
                t = population[j]
                if t[health_status] == 0 and close_enough(v[1], t[1]):
                    t[health_status] = 4


fine = 0
deadcount = 0
immune = 0
sick = 0
hospitalised = 0

for i in range(pop):
    person = population[i]

    if person[health_status] == 2:
        deadcount += 1
    elif person[health_status] ==1:
        immune += 1
    elif person[health_status] == 0:
        fine += 1
    elif person[health_status] == 3:
        hospitalised +=1
    else:
        sick += 1

print(f"state no.{s+1}")
print(f"dead: {deadcount}")
print(f"immune: {immune}")
print(f"sick: {sick}")
print(f"fine: {fine}")
```

```python
    print(f"admitted: {hospitalised}")
    print('')
```

We've also uploaded the corresponding csv file used on our github profiles so be sure to check that out.

6 Conclusion

What can I say, we failed. And we're immensely sorry about that. WE had no idea we would run into so many technical issues along the way, We've learnt it's better to finish things early as superficially easy things are deceivingly tough. As far as Sanju's question goes, intuitively state to state travel shouldn't affect at all. Experimentally, we'll know tomorrow.

References

[1] *First SIR model site, it's complicated be warned*
`https://towardsdatascience.com/modelling-the-coronavirus-epidemic-spreading-in-a-city-with-python-babd14d82fa2`

[2] *SIR code source*
`https://scipython.com/book/chapter-8-scipy/additional-examples/the-sir-epidemic-model/`

[3] *Indian Population*
`https://en.wikipedia.org/wiki/`
`List`$_{o}f_{s}tates_{a}nd_{u}nion_{t}erritories_{o}f_{I}ndia_{b}y_{p}opulation$

[4] *Indian statewise area*
`https://en.wikipedia.org/wiki/`
`List`$_{o}f_{s}tates_{a}nd_{u}nion_{t}erritories_{o}f_{I}ndia_{b}y_{a}rea$

[5] *Flight details and frequencies*
`http://www.airindia.in/time-table.htm`