Maharshi Jinandra(NUID – 001546653)
Cibaca Pramod Khandelwal (NUID - 001052163)
INFO 6205 - Program Structures and Algorithms (Spring 2021)

COVID-19 Disease Simulation using SIR model

1. Introduction

COVID-19 has emerged as a global crisis that threatens to overwhelm public healthcare systems and disrupt the social and economic welfare of every country. The daily lives of billions of people have been impacted by the unprecedented social controls imposed by governments to inhibit the spread of the novel coronavirus.

COVID-19 has spread rapidly amongst a globally susceptible population, with rates of propagation and mortality greater than the averages typically associated with influenza. Initial projections of the global impact indicate that it will likely become the most severe pandemic event over a century, dating to the influenza epidemic of 1918.

2. Aim of the project

In this project we are aiming to simulate the spread of the viruses SARS-CoV-2 and validate below factors

- The R factor of the disease
- K Factor
- The usage and effectiveness of masks
- The prevalence of testing and contact tracing
- The availability and efficacy of the vaccine
- Barriers – Lockdown, Quarantine
- Contact Tracing
- Person Movement - Central Location, Inter Community Travel
- Symptoms

3. Project Details

We have used Python and its libraries to model the virus and it's spread in a population and build the simulation using pygame. For GUI we have used PySimpleGUI module and integrated it with the pygame window to showcase simulation with ensuring user friendly interface.

Configuration file is used to customize the parameters and functions are created for visualization and R/k factor calculations.

The complete code and the executable can be found at https://github.com/maharshi-neu/disease-simulation

- Features
    1. *Customization using Config file*
       Our config file is capable to customize each of the parameters used in the simulation which is useful when we need to test and modify the configuration based on different scenarios. Following points consist of features added in the simulation and its customizable parameters. Each of these parameters has default values added in the file.
    2. *Parameters for initial setup*
       These configurations include flexibility to add below parameter before the start of simulation
        - Number of Rows – Row count for creating Inter community travel grids
        - Number of columns – Column count for creating Inter Community travel grids
        - Number of Days for Simulation – Number of Days for which the simulation would run
        - Number of Person – Population Count with default configured to 300.
        - Initial Infected – Number of Person infected at the start of simulation
        - Transmission Probability – Probability with which the transmission of virus will take place in the range of 0 to 1.
        - Travel Frequency – Frequency/velocity with which People will travel in the simulation grid
    3. *Disease for Simulation*
       For this project we have research and simulated two diseases one of which been the recent pandemic caused due to the sars-cov-1 virus and other been the Influenza virus with less fatal consequences. Both diseases were simulated using SIR model as explained in the later section of the report.
       We have provided a toggle between both the diseases using Radio button as well as config file
    4. *Preventive measures*
       There were few recommended preventive measures suggested and implemented by the government for which we have provided configuration in this simulation. They are as follows
        - Mask (MASK) – Boolean flag is True if we need people to wear the mask
        - Mask Effectiveness (MASK_EFFECTIVENESS) – Value from 0 to 1 for setting how effective the mask should be when the flag is on
        - Ratio of Person (RATIO_OF_POP_WITH_MASKS) – Value in range of 0 to 1 to provide ratio of people wearing the mask
        - Vaccine – Boolean Flag to enable vaccine to provide population based on the config and availability of vaccine calculated.
    5. *Barriers*
       We have added quarantine and lockdown as a barrier replicating the real-world scenarios that were implemented across the world.

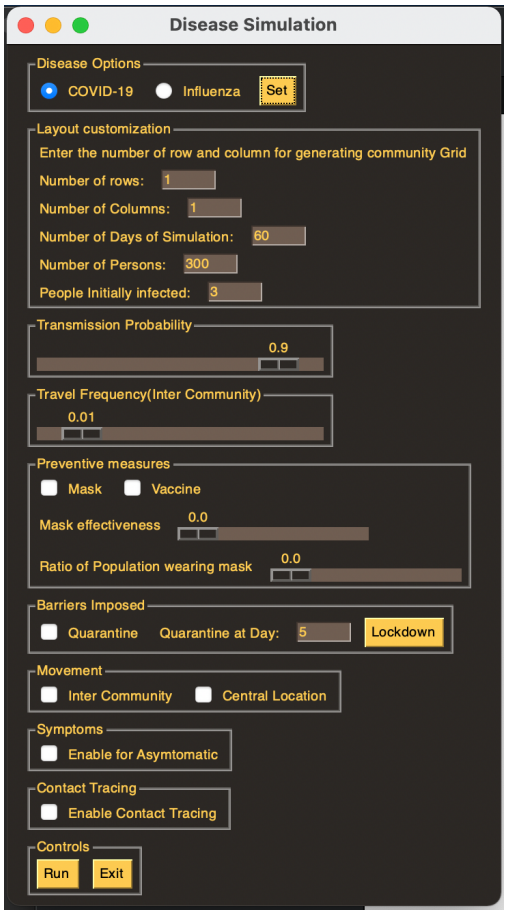
Quarantine provides a way to view the simulation when people are quarantine as well as they are not in quarantine. For this we have added two config parameters

- Set quarantine (QUARANTINE) – Boolean value to activate and disable quarantine.
- Quarantine Day (QUARANTINE_AT_DAY) – Day at which person will quarantine. Default value set to 5
- Lockdown – This button sets the default values for most of the components and provides a visual simulation

6. *Movement*
    - Inter-Community Travel -Inter Community Travel flags enable people travel within the grid and to other locations.
    - Central Location
      Central Location is a place where people will have high chance to contact the effect of virus. Enable Central Location Boolean flag to enable and disable the Central Location.
      Here we have added a condition if central location and Inter community travel is enabled at the same time then we have set the Travel Frequency is lower value to have better visual effect.
7. *Symptoms*

We have functions to showcase variability of scenarios for both kinds of patient whether they are symptomatic or Asymptomatic.

8. *Contact Tracing* - Contact Tracing enables a link between all members who have been in the contact with infected person and enables us to view the real-world virus transmission scenario. For this we have used queue for this purpose.
    - It is performed on daily basis and is constrained with resources; we assume that there exits only one team of contact tracers for the given population.
    - When an infected particle is quarantined, it is put into a queue for contact tracing. Queue.pop () gives the particle to trace. Every particle maintains a stack of particles whom it encountered. Stack.pop () is done on the trace particle if the pop particle is infected then quarantine when Stack empty (trace complete) remove particle from queue.



4. Mathematical Analysis

- SIR model

The Susceptible-Infected-Recovered (SIR) epidemic model is one of the first proposed for epidemiological simulation purposes and has been utilized widely due to its simplicity and effectiveness. In the SIR model, a society consists of three *compartments*.

1. susceptible (S) - contains individuals who are vulnerable and not yet infected
2. infected (I) - formed from susceptible individuals who become infected; in this state, they can shed the virus and spreading the disease.
3. recovered (R) - consists of previously infected individuals who have overcome the disease. The recovered individuals are presumed to have acquired some level of immunity to the disease, such that they have a lower probability of reinfection compared to susceptible individuals.

A simple mathematical description of the spread of a disease in a population is the so-called SIR model, which divides the (fixed) population of N individuals into three "compartments" which may vary as a function of time, t:

- S(t) are those susceptible but not yet infected with the disease.
- I(t) = the number of infectious individuals.
- R(t) are those individuals who have recovered from the disease and now have immunity to it

  Each of these parameters are calculated in our Simulator.py file using the formulas from the extracted from the research work.

- Clock and Day Sync
  Since we have utilized Pygame library in python for this simulation, we need to calculate Frames per second (FPS) to capture each state of simulation.
  Each day is equal to 60 FPS and 1 day in this simulation would be calculated by FPS * Run time in Days as stated in the config
- R0-factor
  R0 can be used to measure any contagious disease that may spread in a susceptible population. R-avg is calculated to gain the average r value of all particles.

- Particle
  Individual containers are created for SIR parameters and particles.
  A particle pair travels inter community per tick if there is more than one community. Max 2 tries are made to find unique particles suitable for travel
  Particles are chosen at random from all containers
  Filtering criteria:
  - Particles should not be in the same community
  - No particle should be quarantined
  - No particle should be in traveling phase
- K-factor

  The way the coronavirus is spread is not always the same. Not all positive cases infect the same number of people – instead, most contagions appear to be linked to specific events and super spreaders. Most infections are caused by a few people, and many positive cases never transmit the disease. This is both good and bad news.

  This highly skewed, imbalanced distribution means that an early run of bad luck with a few super-spreading events, or clusters, can produce dramatically different outcomes even for otherwise similar countries.

- Quarantine
  Moves infected particle to quarantine at recovery rate mentioned in config file. In case of contact tracing override flag is passed and even asymptomatic particles are quarantined
- Travel to central Location
  This function runs every 2nd day chosen particle travels to the central location right bottom room which has a smaller area than any other room in the simulation. The small area increases the probability of a particle of contracting the virus

if a particle with the virus is in the room already.

- Max: 5 particles are chosen at random from the whole container
- Min: 0 also can be chosen if the particles do not meet the following criteria.
  - Particle should not already be already in the room
  - Particle should not be quarantined

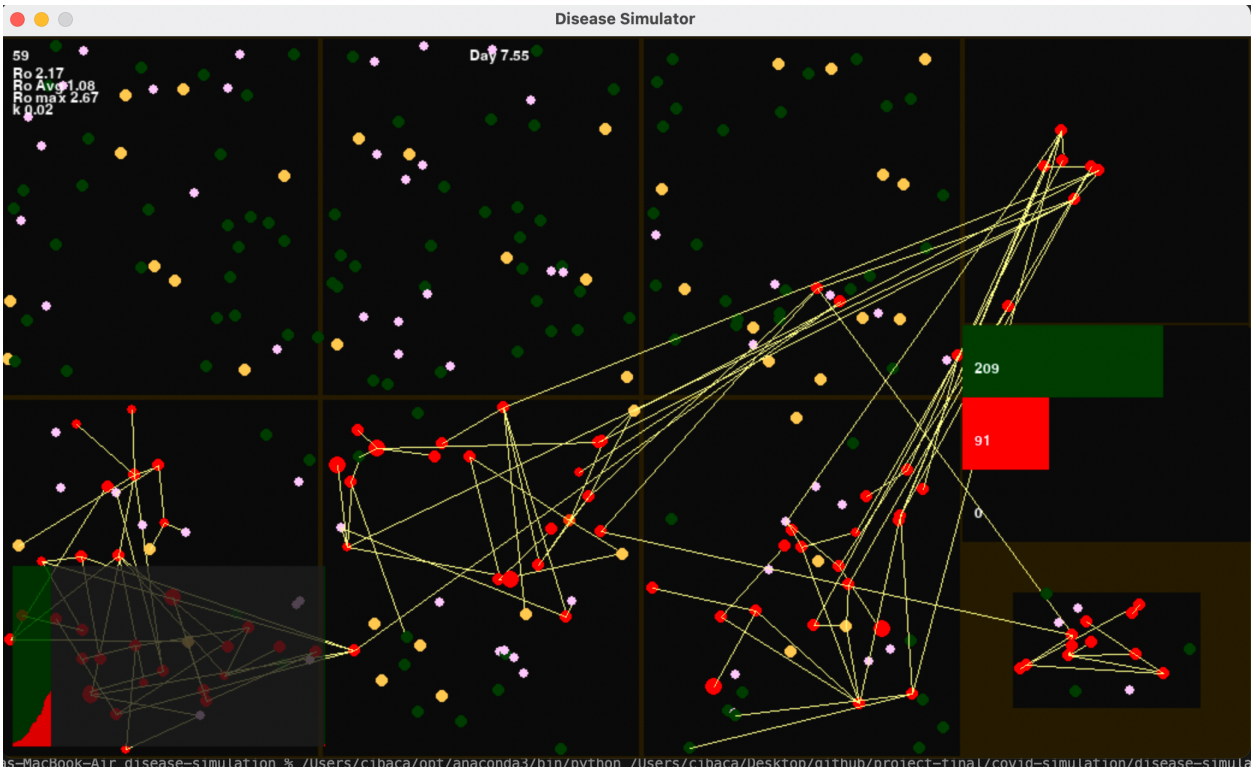
It runs every 10th day Particles in central location are transferred back to their original room only criteria filtering this is if the particle is quarantined.

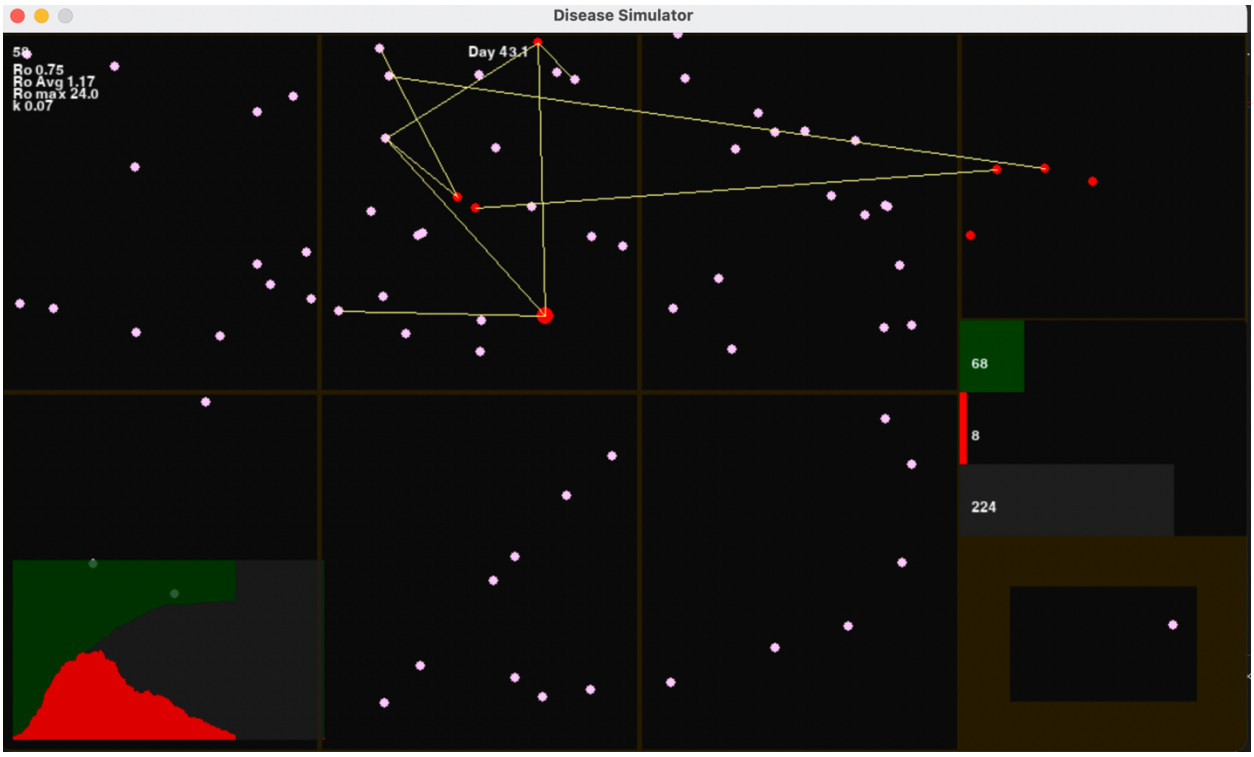
- Vaccinate

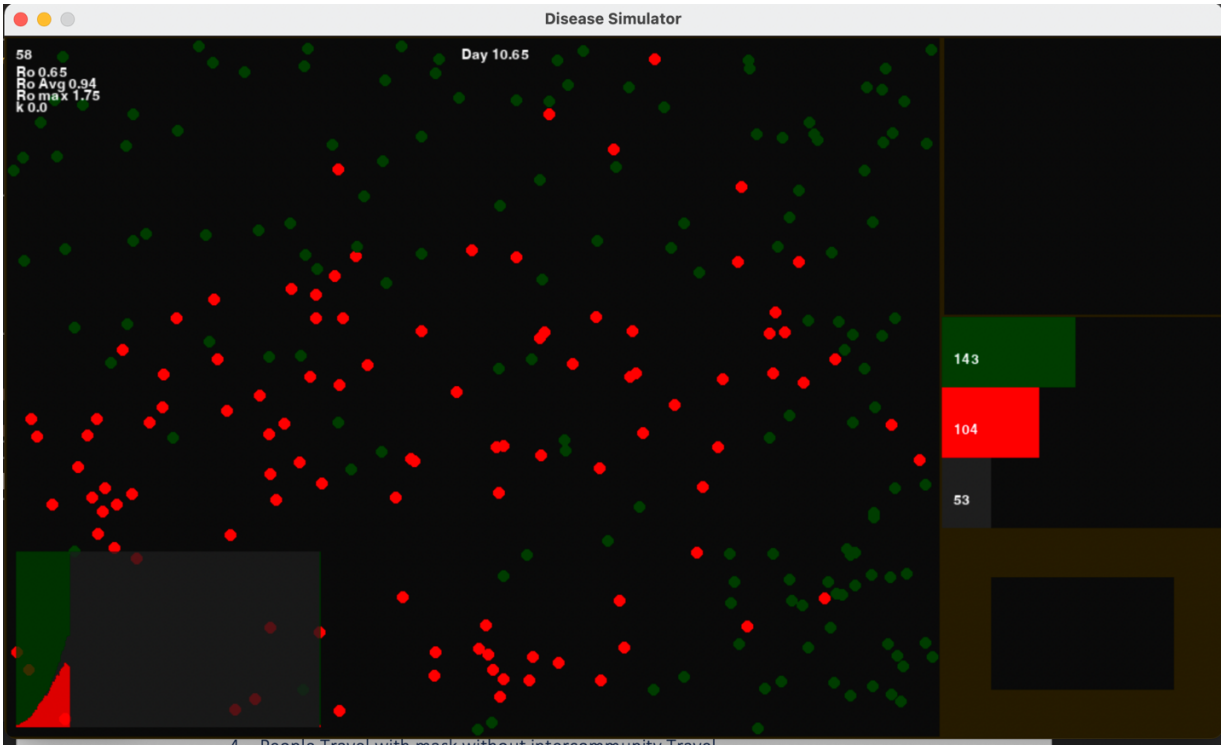
  Vaccine sessions are held twice per day if vaccine are available then it is provided to particle vaccine provides shield to particle shield brings down the probability drastically

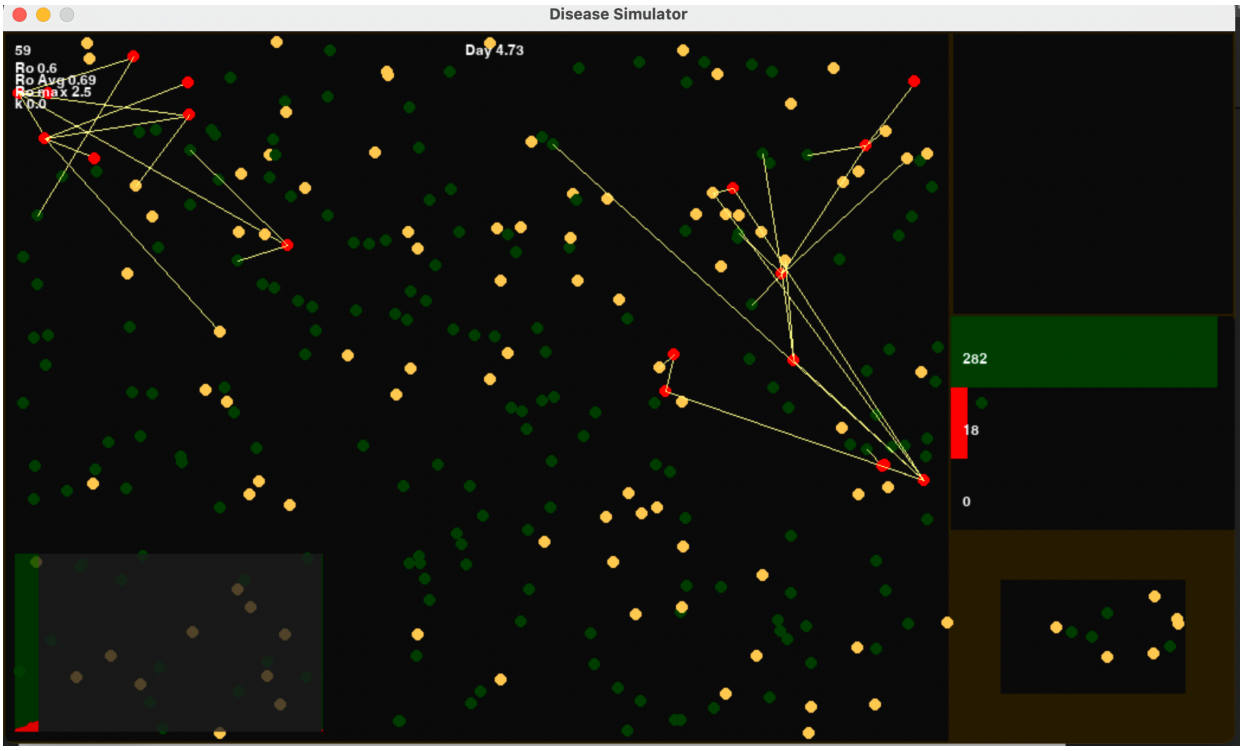
  2 doses can be given to a particle, 2nd dose gives complete immunity (depending on the shield value) and calculated by VACCINE_DISPERSION_RATE / suslen. When susceptible length is high than vaccine distribution is slow and when susceptible length drops distribution is higher than before

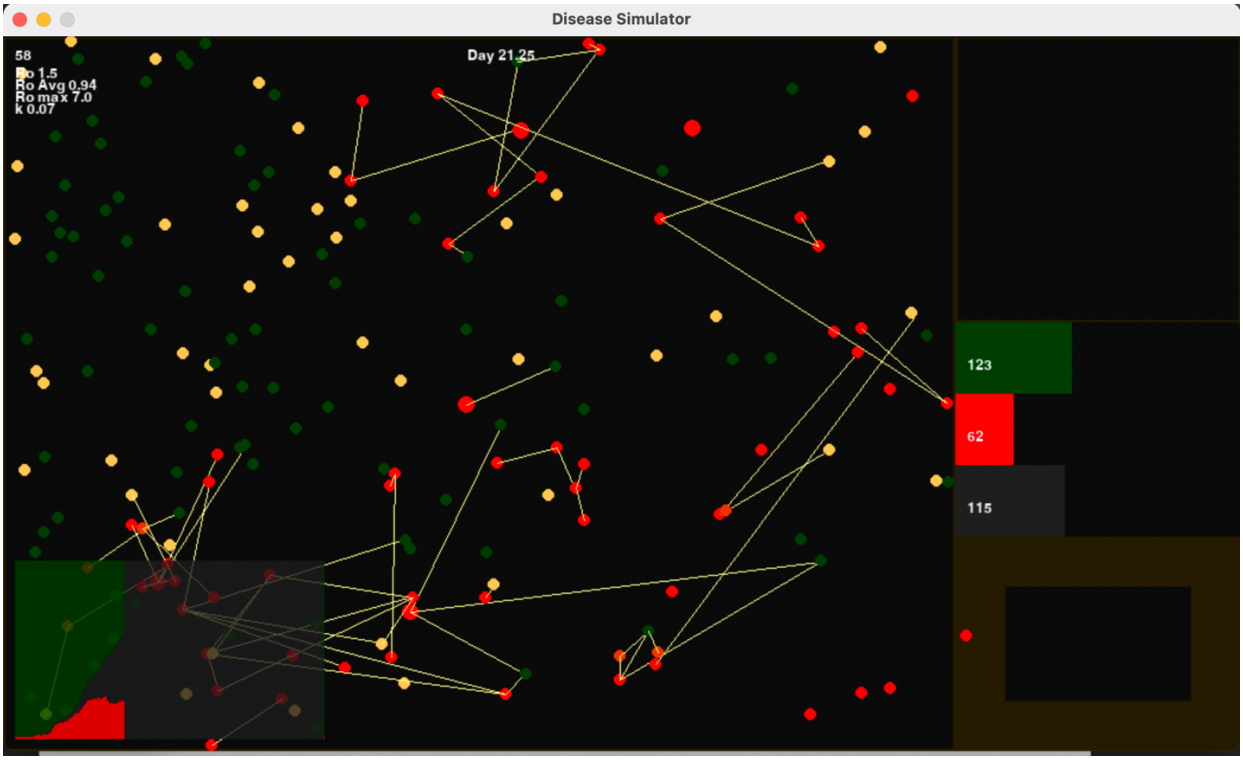
5. Simulation Scenarios

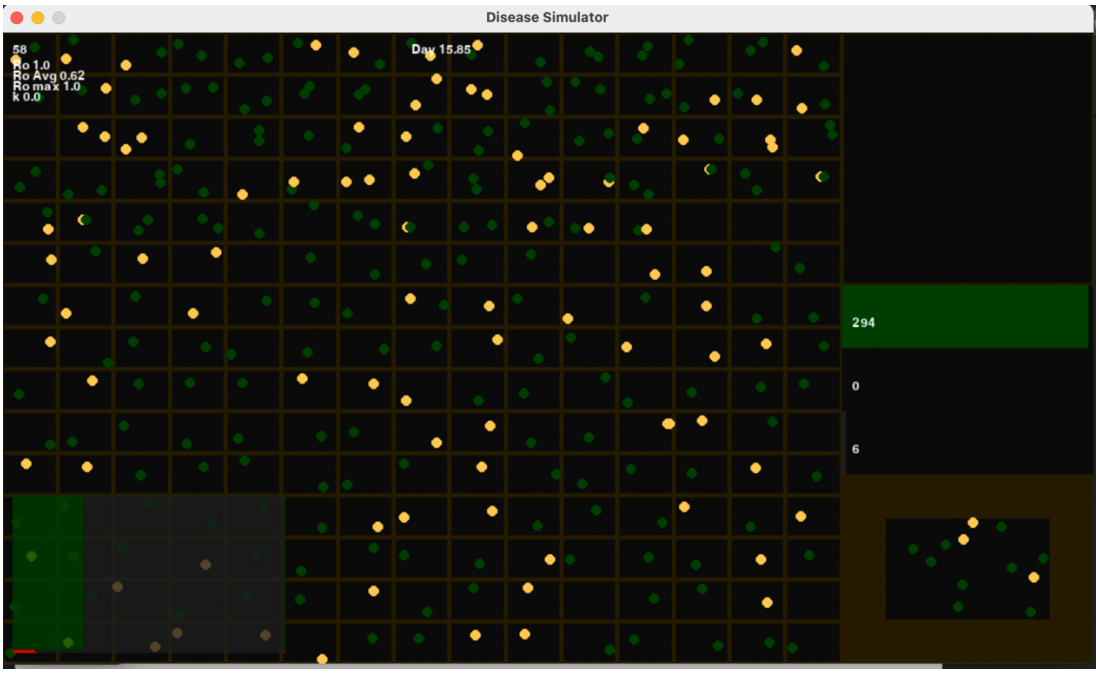



Disease Simulator
58
Ro 0.75
Ro Avg 1.17
Ro max 24.0
k 0.07
Day 43.1
68
8
224

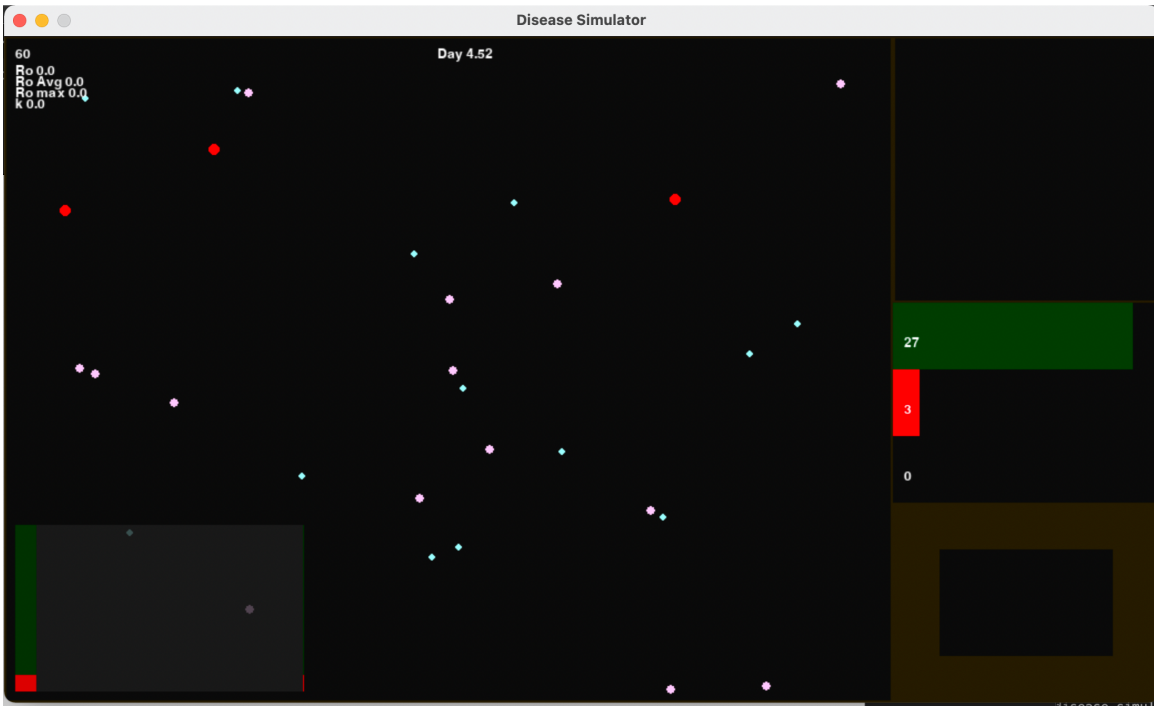

Disease Simulator
58
Ro 0.65
Ro Avg 0.94
Ro max 1.75
k 0.0
Day 10.65
143
104
53


Disease Simulator
59
Day 4.73
Ro 0.6
Ro Avg 0.69
Ro max 2.5
k 0.0
282
18
0


Disease Simulator
58
Day 21.25
Ro 1.5
Ro Avg 0.94
Ro max 7.0
k 0.07
123
62
115



Vaccinated for both the doses



Unit test run -

```
(base) cibaca@Cibacas-MacBook-Air disease-simulation % python src/unit_test.py
pygame 2.0.0 (SDL 2.0.12, python 3.8.5)
Hello from the pygame community. https://www.pygame.org/contribute.html
...............
----------------------------------------------------------------------
Ran 15 tests in 0.001s

OK
(base) cibaca@Cibacas-MacBook-Air disease-simulation %
```

6. Complexity Analysis and Optimizations

list

The Average Case assumes parameters generated uniformly at random.

Internally, a list is represented as an array; the largest costs come from growing beyond the current allocation size (because everything must move), or from inserting or deleting somewhere near the beginning (because everything after that must move). If you need to add/remove at both ends, consider using a collections.deque instead.

Operation	Average Case	Amortized Worst Case
Copy	O(n)	O(n)
Append[1]	O(1)	O(1)
Pop last	O(1)	O(1)
Pop intermediate[2]	O(n)	O(n)
Insert	O(n)	O(n)
Get Item	O(1)	O(1)
Set Item	O(1)	O(1)
Delete Item	O(n)	O(n)
Iteration	O(n)	O(n)
Get Slice	O(k)	O(k)
Del Slice	O(n)	O(n)
Set Slice	O(k+n)	O(k+n)
Extend[1]	O(k)	O(k)
Sort	O(n log n)	O(n log n)
Multiply	O(nk)	O(nk)
x in s	O(n)	
min(s), max(s)	O(n)	
Get Length	O(1)	O(1)

dict

The Average Case times listed for dict objects assume that the hash function for the objects is sufficiently robust to make collisions uncommon. The Average Case assumes the keys used in parameters are selected uniformly at random from the set of all keys.

Note that there is a fast-path for dicts that (in practice) only deal with str keys; this doesn't affect the algorithmic complexity, but it can significantly affect the constant factors: how quickly a typical program finishes.

Operation	Average Case	Amortized Worst Case
k in d	O(1)	O(n)
Copy[3]	O(n)	O(n)
Get Item	O(1)	O(n)
Set Item[1]	O(1)	O(n)
Delete Item	O(1)	O(n)
Iteration[3]	O(n)	O(n)

set

See dict -- the implementation is intentionally very similar.

Operation	Average case	Worst Case	notes
x in s	O(1)	O(n)	
Union s|t	O(len(s)+len(t))		
Intersection s&t	O(min(len(s), len(t))	O(len(s) * len(t))	replace "min" with "max" if t is not a set
Multiple intersection s1&s2&..&sn		(n-1)*O(l) where l is max(len(s1),..,len(sn))	
Difference s-t	O(len(s))		
s.difference_update(t)	O(len(t))		
Symmetric Difference s^t	O(len(s))	O(len(s) * len(t))	
s.symmetric_difference_update(t)	O(len(t))	O(len(t) * len(s))	

collections.deque

A deque (double-ended queue) is represented internally as a doubly linked list. (Well, a list of arrays rather than objects, for greater efficiency.) Both ends are accessible, but even looking at the middle is slow, and adding to or removing from the middle is slower still.

Operation	Average Case	Amortized Worst Case
Copy	O(n)	O(n)
append	O(1)	O(1)
appendleft	O(1)	O(1)
pop	O(1)	O(1)
popleft	O(1)	O(1)
extend	O(k)	O(k)
extendleft	O(k)	O(k)
rotate	O(k)	O(k)
remove	O(n)	O(n)

Union-grid spatial partition

- Worst Case => c(n^2)
- Average Case => c(n)
- Best Case => C(nlogn)

k-d tree (Tried and research but it had high collision misses)

- Worst Case => c(n^2)
- Average Case => c(n)
- Best Case => C(nlogn)

Trace_line – function to link particles for contact tracing

- Worst Case: O(n^2)
- Best case: O(n)

Travel to central location()

- Worst Case: O(n)
- Best Case: O(n)

Contact _tracing
- Worst Case: O(n^2)
- Best Case: O(n)

Vaccinate
- Worst Case: O(n)
- Best Case: O(n)

Update the grid
- Worst Case: O(n)
- Best Case: O(n)

Update_and_render
- Worst Case: O(n^2)
- Best Case: O(n^2)

Init Populate simulation
- Worst Case: O(N)
- Best Case: O(N)

Collision grid
- Worst Case: O(ROW*COL)
- Best Case: O(ROW*COL)

Inter community grid
- Worst Case – O(ROW*COL)
- Best Case _O(ROW*COL)

7. Results and Analysis

All results are stored in log/sim.py and graph are visible in the live simulation and can be paused and controls using the Config.py

8. Conclusion

Based on the results of the simulation by implementing multiple algorithms and scenarios to test covid-19 and influenza simulation we have arrived at below conclusions

- COVID-19 is more lethal as compared to Influenza
- R factor is itself not sufficient for predicting and analyzing the disease. Based on the infection, virus, and environment parameters we need to consider k factor and try to find other parameters.
- Particle collision simulation works better with Uniform grid spatial partition instead of sweep and prune method.
- Kd-Tree works but it misses more collisions as compared to uniform grid

9. References

a. Research
   i. https://github.com/coronafighter/coronaSEIR
b. Simulation
   i. https://arxiv.org/pdf/2001.02436.pdf
   ii. https://jckantor.github.io/CBE30338/03.09-COVID-19.html
   iii. How to Make a Coronavirus Simulator on Python using Pygame/Pymunk
   iv. https://dspace5.zcu.cz/bitstream/11025/10652/1/Li.pdf
   v. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.696.6994&rep=rep1&type=pdf
   vi. https://www.nature.com/articles/s41591-020-1092-0
   vii. https://jamanetwork.com/journals/jamanetworkopen/fullarticle/2774707
c. UI
   i. https://www.youtube.com/watch?v=D8-snVfekto
d. Visualizations
   i. https://www.youtube.com/watch?v=YaGQ2KS5VA0
e. Maths/Statistics
   i. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7388782/
   ii. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7237380/
   iii. Contact rate and removal rate in the SIR model of infectious disease
   iv. https://f1000researchdata.s3.amazonaws.com/manuscripts/29578/13eaf24b-f041-4af4-98e0-90e25d59eefe_26786_-_danielle_kurtin.pdf?doi=10.12688/f1000research.26786.1&numberOfBrowsableCollections=27&numberOfBrowsableInstitutionalCollections=4&numberOfBrowsableGateways=26