~~~~~~

Impact of Human personality

on the

Spread of COVID19

~~~~~~

(Agent-Based and probability code using graphing libraries, coordinate math, and random integer generation)

New Mexico

SuperComputing Challenge

Final Report

April 10, 2021

Team 23

Los Alamos Middle School (LAMS)

Team member:

Hyunoo (Hayden) Kim

Project Mentor:

S. Jun Kim

Table of Contents

Executive Summary

|~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~|

Based on current events, we can all agree that the CoronaVirus was a devastating virus that has taken many lives and has caused us to drastically adjust our day-to-day schedule. I believe that we could have prevented most of this. My code aims to help pin-point which preventing measures are the most effective against the spread of the virus and also has many basic and sophisticated features to make the running of the code and the results be realistic. The spread starts with one infected that stays infected for 14 virtual days and also cannot be isolated or hospitalized. The first infected agent starts in the middle of the environment. My project also has the ability to take into account human personality. Because the CoronaVirus has been confirmed to be airborne, a big part of the spread is of where people are and where they move to. To maintain realism, I implemented a feature that gave agents unique movement patterns, calling it Human Personalities. As of now, there are five different personalities that an agent can have and they are distributed evenly among the agents unless manipulated otherwise. These five are **Stubborn**, **Obedient**, **Loud**, **Quiet**, and **Teen**. Each has their own style of movement that either contributes or stops the spread of the virus. (The personality names do not necessarily point at age groups or anything like that. They simply are words that relate to the movement patterns that each has.) Most of the code uses probability and relies on random number generation. This is what makes my code different from other COVID19 models. Many always depend on math or any concrete data about the virus, but my code can go purely based off of round number guesses. This causes each output to be unique in some way. Each day is shown as a scatterplot of the 600 by 600 large environment. (See Appx. A). The plot is color coded to either show personalities or status in infection. This project aims to help educate people. Data has shown that if 100% of the population wore masks and socially isolated themselves when positive with COVID19, then the number of infected people would lessen by over 650% (~7.5 times). My code can show data on individual groups. This is so that I can compare a stubborn population to an obedient population spread. So, are you helping the spread? Are you a contributor to the spread, or are you a person that helps stop the spread?

Project Introduction

|~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~|

As many of us are aware, the CoronaVirus outbreak happened in 2019 and the first COVID19 case in the United States of America was found during early 2020. The virus entered New Mexico on March 11, 2020 and caused most schools in New Mexico to shut down 2 to 3 months before the end of the school year. Because of this virus, many markets crashed, but some skyrocketed. Most people will agree that this outbreak was destructive to the economy and many other things. The virus is still here. We are currently living in the middle of the pandemic. As of now, COVID19 has taken approximately 2.9 million people. There have been 135 million reported and confirmed cases of the virus. In the United States alone, there have been 31.2 million cases and 561 thousand deaths. This means that 1 in every 56 people were infected with the CoronaVirus worldwide and 1 in every 10 people were infected with the virus in the United States. Of course, all of the cases were not present at a single point in time, but this is just to show how damaging the virus is. The peak of new cases was 300 thousand cases per day. This was just three months ago. If you were married, had 2 parents, 1 grand parent, 2 cousins, 2 aunts, 2 siblings with 2 kids of their own, and 2 kids, then you would have a 10% chance that one of your family members have been infected. This is also only your side of the family. The chance would be doubled if your spouse's family counted. This is if only 2 million cases were present in one day. The total is 31 million cases. The data shows that this virus destroyed families, and caused lots of harm to many people. My project aims to help that. It has the capability to generate significant data that can change how people view the virus. Some of us may think that we just need to wear masks and social distance to stop the spread, others stay confined to their homes. Some even think that this is all a myth. Most people are good. They want to stop the spread. My project can educate many people about the virus. Help stop the spread.

Description

|~~~~~~~~~~~~~~~~~~~~~~~~~|

For my project, I used agent-based modeling to simulate an environment of 250 people. (Doing this on a global scale would require very accurate measurements and ratios that I do not have yet. I would also need a better computer to run it on.) The platform I used was Python. My code works by defining all necessary functions and modules at the beginning and then calling them in order at the end to create a realistic day. Then the day repeats, but with different variable inputs from the previous day.

My code can be split into 5 different parts:

- Human definitions
- Main functions
- Extra functions
- Plot first day
- Main loop

Human Definitions:

This part of my code defines the parameters for the agents. I used a “Class” function to do this. It just acts like the blueprint for the creation of the agents at the beginning of the code. This first segment is of the variables needed to create an agent.

```
#humand fuction definition :
class human:
    def __init__(self, status, color, location_x, location_y, daysin, inc, mask, homexcoor, homeycoor, asym
        self.location_x = location_x
        self.location_y = location_y
        self.status = status
        self.color = color
        self.daysin = 0
        self.inc = inc
        self.mask = mask
        self.homexcoor = homexcoor
        self.homeycoor = homeycoor
        self.asymm = asym
        self.doctor = doctor
        self.mdis = mdis
        self.personality = personality
        self.movepat = movepat
        self.party = party
```

Each agent requires many variables which help the code run properly. Some major ones are the position variables, status variable, and the personality variables. This is

just the blueprint. The next segment will contain the randomization and definition of the agents.

```
#ramdomize the people location using human function
x = 0
color = 'green'
while x <= n:
    x1 = random.randint(-300, 300)
    y1 = random.randint(-300, 300)
    homex = random.randint(-250, 250)
    homey = random.randint(-250, 250)
    asym = 0
    doctor = random.randint(0,25)
    if doctor == 25:
        doctorcon = 'y'
    else:
        doctorcon = 'n'
    #Chart:
    #1: stuborn
    #2: obedient
    #3: loud, doesn't like mask
    #4: quiet, but still doesn't wear mask very often
    #5: teen, likes parties
    #1 = move lots, likes to go to supermarket
    #2 = stays at home and avoids people
    #3 = goes out a lot
    #4 = avoids people but move lots. doesn't wear masks
    #5 = gathers with other teens and like to go the the supermarket in groups
    rper = random.randint(1, 5)
    if rper == 1:
        personality = "s"
        movep = "1"
    if rper == 2:
        personality = "o"
        movep = "2"
    if rper == 3:
        personality = "l"
        movep = "3"
    if rper == 4:
        personality = "q"
        movep = "4"
    if rper == 5:
        personality = "t"
        movep = "5"
    if personality == 's':
        color = 'red'
    if personality == 'o':
        color = 'blue'
    if personality == 'l':
        color = 'yellow'
    if personality == 'q':
        color = 'purple'
    if personality == 't':
        color = 'pink'

    people.append(human('s', color , x1, y1, 0, 0, 'n', homex, homey, 0, doctorcon, 0, personality, movep, 'n'))
    if people[x].status == 's':
        color = 'green'
    elif people[x].status == 'i':
        color = 'red'
    else:
        color = 'gray'
    x = x + 1
```

This segment of the code uses random numbers and some parameters to define all of the variables needed to finish an agent.

Main Functions:

Three main functions run inside of the main loop.

1. Spreading module
2. Movement module
3. Boundary module

1.) Spreading module:

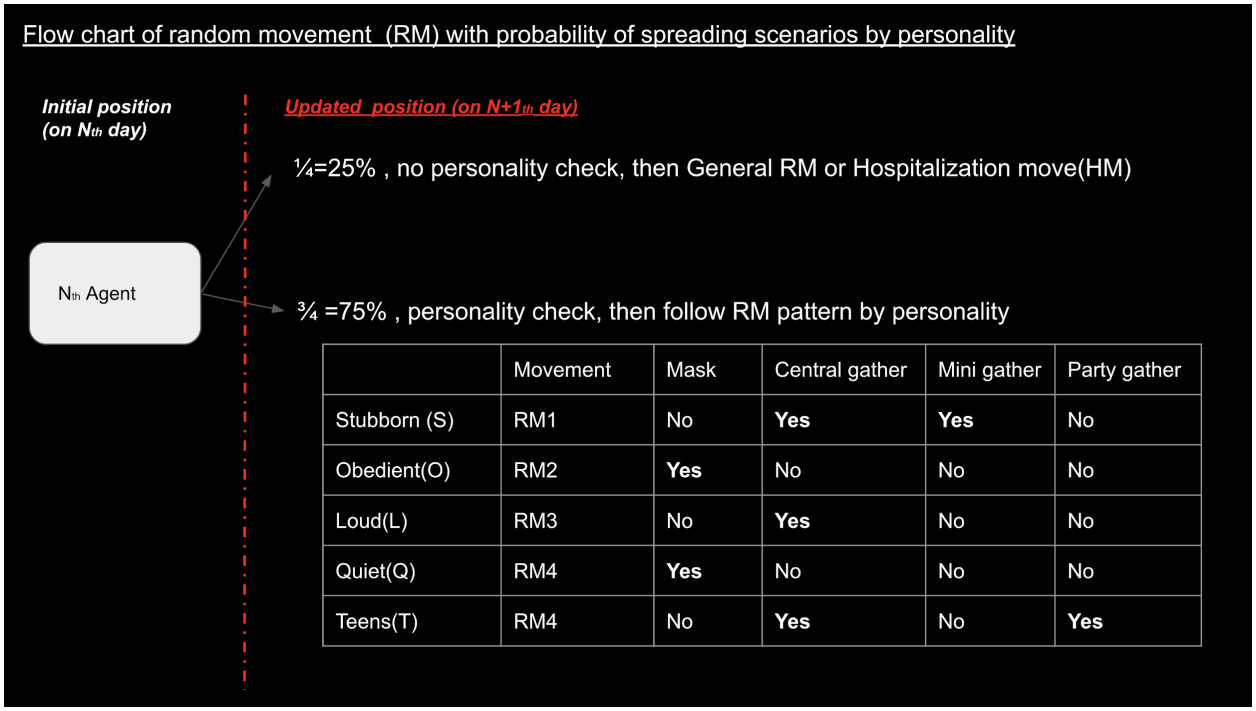
The spreading module contains multiple loops that use coordinate math to calculate spreads.

```
#checks the spread of the disease and spreads it
def check_s():
    global n
    x = 0
    y = 0
    r = 50
    while x <= n + 1:
        global inftotal
        if people[x].status == 'i':
            y = 0
            while y <= n + 1:
                #print("error")
                if people[x].inc > 0:
                    #print("broke from", x, "had", people[x].inc, "inc")
                    break
                if people[y].status == 'i' or people[y].status == 'r' or people[y].asymm ==1:
                    #print("infected... passing")
                    y = y + 1
                    continue
                else:
                    if people[y].location_x < r + people[x].location_x and people[y].location_x > -r + people[x]
                        if people[y].movepat == '2':
                            ic = random.randint(0,25)
                        elif people[y].mask == 'n':
                            ic = random.randint(0,5)
                        elif people[y].mask == 'y':
                            if people[y].doctor == 'y':
                                ic = random.randint(0,50)
                            else:
                                ic = random.randint(0,25)
                        #print(ic)
                        #ic = 3
                        if ic == 3:
                            people[y].status = 'i'
                            inftotal = inftotal + 1
                            people[y].inc = 5
                            asymc = random.randint(0, 10)
                            if asymc == 5:
                                people[y].asymm = 1
                            #print("checked", y, "infected from", x)
                            y = y + 1

                        else:
                            #print("checked", y, "not infected from", x)
                            y = y + 1

                    else:
                        #print("passed", y, "from", x)
                        y = y + 1


        else:
            pass
        x = x + 1
        #comment line below out if not want icolation
        icolation()
        #print(x, y)
```

This function takes input variables from agents and uses position to calculate what agents have the chance to become infected. Some of the variables it uses are position variables and status variables. As seen at the bottom of the screen shot, a function called “icolation” is called. This is an extra feature that will be explained later. (It is also spelled wrong in the code. This was done on purpose.)

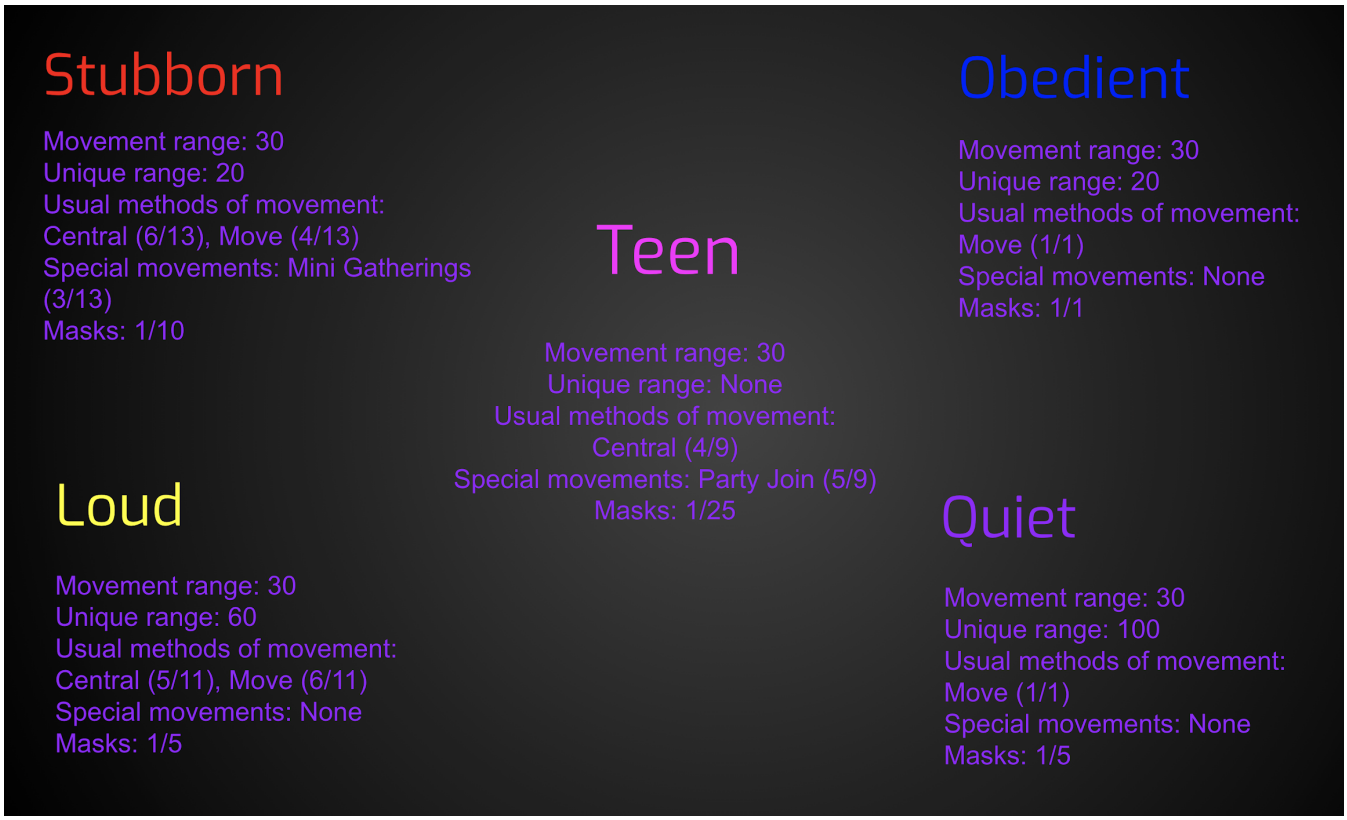
2.) Movement module:

The movement module is the part of my code that has been updated the most. This is because of constant ideas for realism. Currently, it is called “move_5” because it is the fifth version of move functions. The movement module at this current time allows many different things. Each agent with a different human personality can move differently. Below is a flow chart of the possible movements an agent can do. (This is from my slideshow)



	Movement	Mask	Central gather	Mini gather	Party gather
Stubborn (S)	RM1	No	**Yes**	**Yes**	No
Obedient(O)	RM2	**Yes**	No	No	No
Loud(L)	RM3	No	**Yes**	No	No
Quiet(Q)	RM4	**Yes**	No	No	No
Teens(T)	RM4	No	**Yes**	No	**Yes**

This is also from my slideshow. It shows, in more detail, how agents move with personality. (These names (Stubborn, Obedient, Loud, Quiet, Teen) do not signify any sort of relation to anything other than just being personality names that I used.)



3.) Boundary Module:

This is just a simple check that the code performs to keep all of the agents inside of the set boundaries.

```
#this funtion checks the people bondaries
def check_b():
    global n
    x = 0
    while x <= n + 1:
        if people[x].location_x == -400 or people[x].location_x == 400:
            x = x + 1
            continue
        if people[x].location_x <= -301:
            people[x].location_x = 250
        if people[x].location_x >= 301:
            people[x].location_x = -250
        if people[x].location_y <= -301:
            people[x].location_y = 250
        if people[x].location_y >= 301:
            people[x].location_y = -250
        x = x + 1
```

Extra Functions:

These functions and modules were added to create more factors that helped bring by simulation closer to being almost identical to the real world. There are currently 8 extra functions.

1. Masks
2. Isolation
3. Vaccine
4. Personality control
5. Color modification
6. Graphical output
7. Scatterplot days
8. Hospitalization

1.) Masks:

This function gave controllable mask distribution. The Spreading module takes masks into account before giving chances of infection.

```
#adds maskes to people
def mask():
    global n
    x = 0
    #the lower the number, the higher chance. do not go below 1
    while x <= n + 1:
        #print("error")
        if people[x].doctor == 'y':
            people[x].mask = 'y'
        elif people[x].personality == 's':
            rsc = random.randint(0,10)
            if rsc == 4:
                people[x].mask = 'y'
            else:
                people[x].mask = 'n'

        elif people[x].personality == 'o':
            people[x].mask = 'y'

        elif people[x].personality == 'l':
            rsc = random.randint(0,5)
            if rsc == 4:
                people[x].mask = 'y'
            else:
                people[x].mask = 'n'

        elif people[x].personality == 'q':
            rsc = random.randint(0,5)
            if rsc == 4:
                people[x].mask = 'y'
            else:
                people[x].mask = 'n'

        elif people[x].personality == 't':
            rsc = random.randint(0,25)
            if rsc == 4:
                people[x].mask = 'y'
            else:
                people[x].mask = 'n'


        x = x + 1
```

2.) Isolation:

This function runs at the end of each spread check. It gives infected agents a chance to become isolated.

```
def icolation():
    global n
    x = 0
    while x <= n:
        if people[x].status == 'i'and people[x].inc <= 0 and people[x].asymm < 1 and people[x].location_x
            randci = random.randint(0, 125)
            if randci == 3:
                people[x].location_x = 400
                people[x].location_y = 400
                people[x].mdis = 1
        x = x + 1
```

3.) Vaccine:

This function is not yet complete and will be implemented in the future. It will give a randomized release date of the vaccine and a random effectiveness percentage. Then agents will start getting vaccinated and it will act like an upgrade to the mask.

4.) Personality Control:

This concept is not yet finished but is close to being ready to implement into the code. It allows the code to selectively give agents personalities and allows the developer to control what the starting infected agent is.

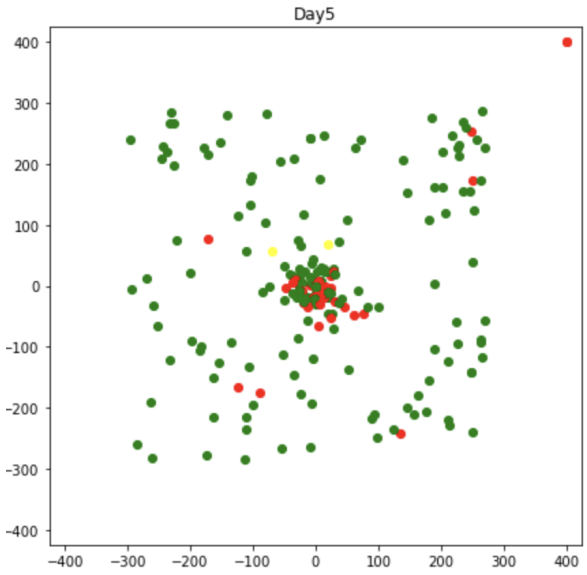
```
x1 = random.randint(-300, 300)
y1 = random.randint(-300, 300)
personinput = input("What personality would you like the first infected agent to have?")
if personinput == 's':
    people.append(human('i', 'red', 0, 0, 0, 0, 'n', homex, homey, 0, 'n', 0, 's', movep, 'n'))
elif personinput == 'o':
    people.append(human('i', 'red', 0, 0, 0, 0, 'n', homex, homey, 0, 'n', 0, 'o', movep, 'n'))
elif personinput == 'l':
    people.append(human('i', 'red', 0, 0, 0, 0, 'n', homex, homey, 0, 'n', 0, 'l', movep, 'n'))
elif personinput == 'q':
    people.append(human('i', 'red', 0, 0, 0, 0, 'n', homex, homey, 0, 'n', 0, 'q', movep, 'n'))
elif personinput == 't':
    people.append(human('i', 'red', 0, 0, 0, 0, 'n', homex, homey, 0, 'n', 0, 't', movep, 'n'))
```

5.) Color Modification:

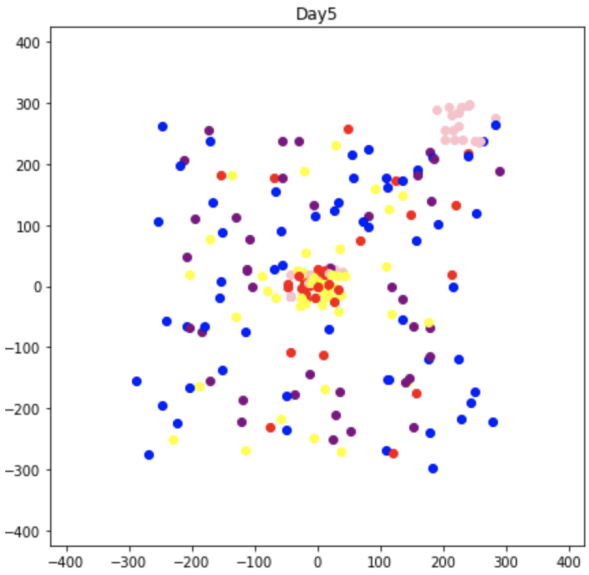
When I implemented human personality to the code, there needed to be a way to see the personalities affecting the movement patterns. So, as a result, instead of using the regular colors that correlated to the status of infection a different option was added

to allow the colors to relate to the personalities. Red for Stubborn, Blue for Obedient, Yellow for Loud, Purple for Quiet, and Pink for Teen.

Without Color Change

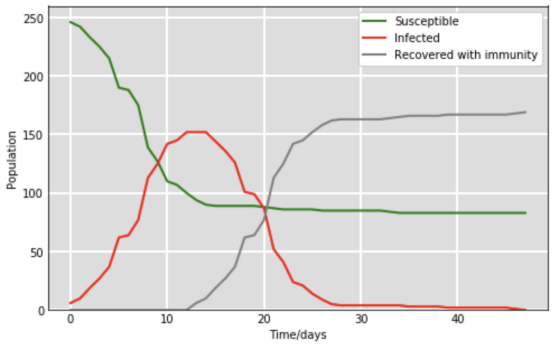


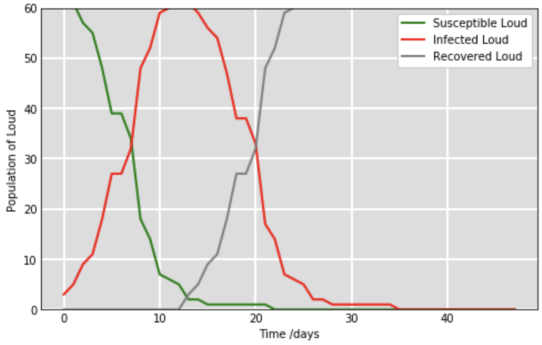
Color Change

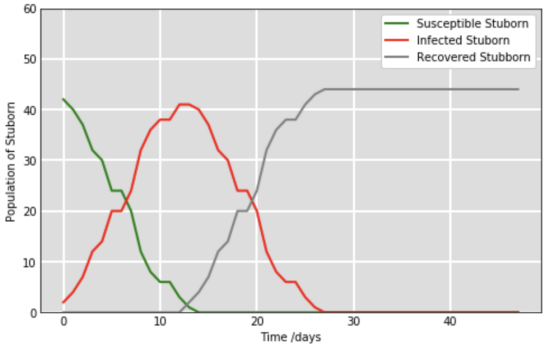


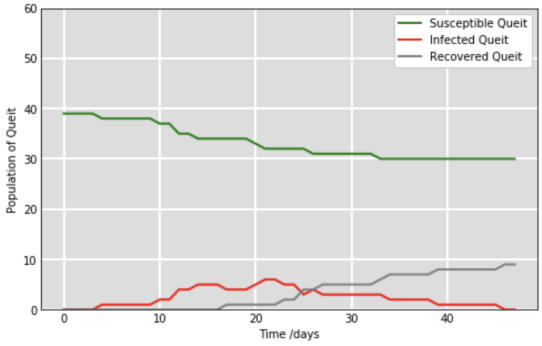
6.) Graphical Output:

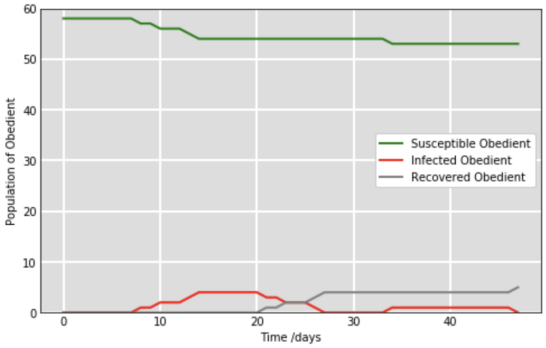
This feature was essential for clear results. Currently, the simulation gives 7 graphs as a result to one run. These are: The Total population, Stubborn, Obedient, Loud, Quiet, Teen, and Total population with isolated and hospitalized.

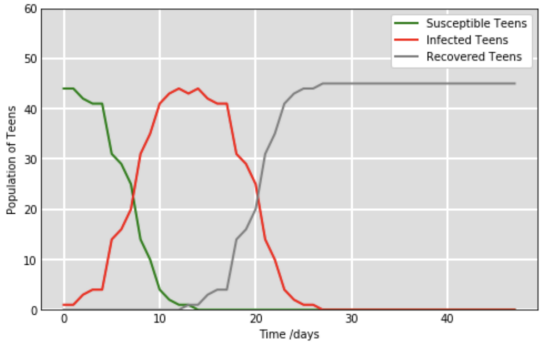

Susceptible
Infected
Recovered with immunity
Population
Time/days

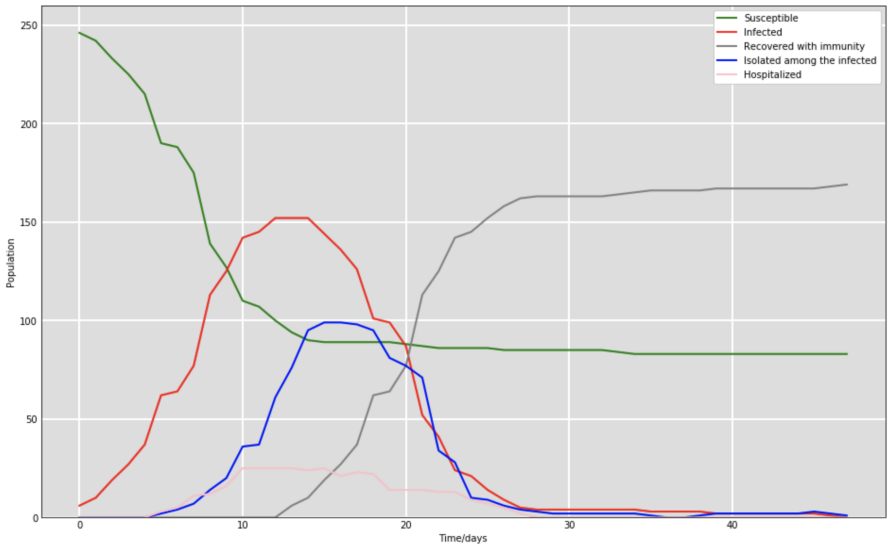

Susceptible Loud
Infected Loud
Recovered Loud
Population of Loud
Time /days


Susceptible Stuborn
Infected Stuborn
Recovered Stubborn
Population of Stuborn
Time /days


Susceptible Queit
Infected Queit
Recovered Queit
Population of Queit
Time /days

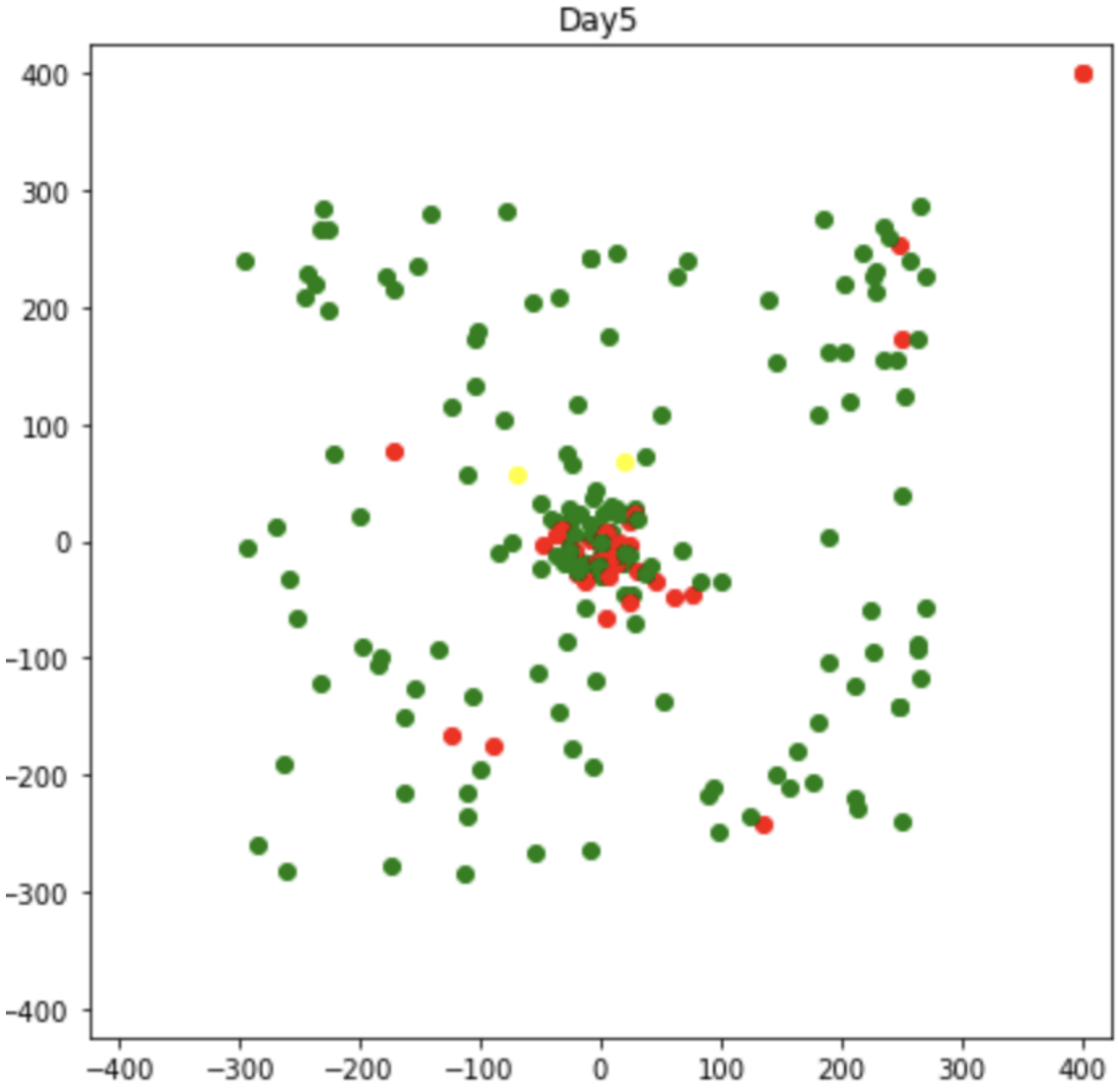

Susceptible Obedient
Infected Obedient
Recovered Obedient
Population of Obedient
Time /days


Susceptible Teens
Infected Teens
Recovered Teens
Population of Teens
Time /days


Susceptible
Infected
Recovered with immunity
Isolated among the infected
Hospitalized
Population
Time/days

7.) Scatterplot days:

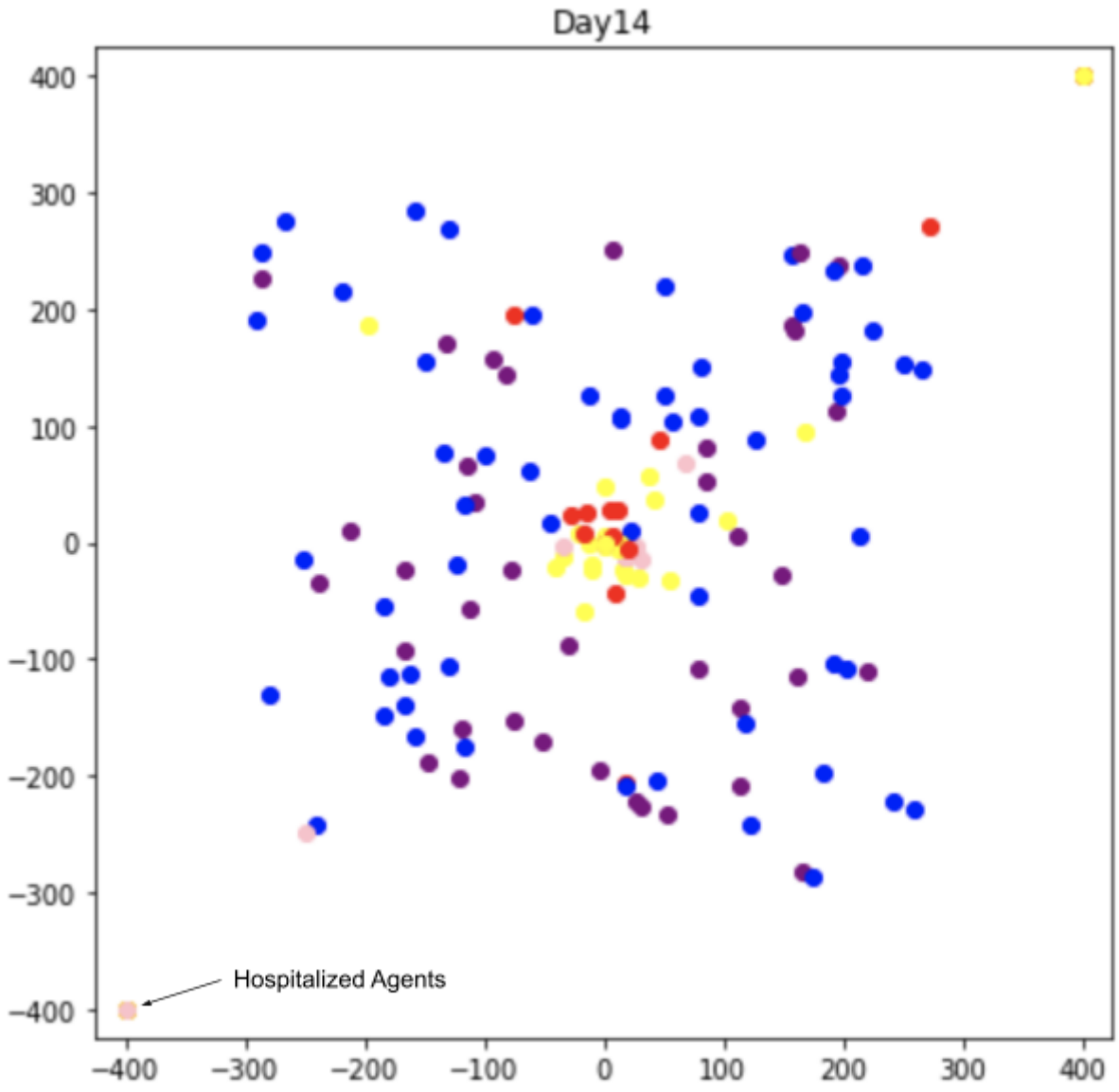
Since the start of my project, one of the main struggles I faced was how to give a visual of how the environment worked and what it looked like. I started by looking into TurtlePy, a simple graph library that I was very familiar with. Then I realized that the module was very weak. I needed something more powerful to show visuals. Eventually, after taking about a month break from visuals, I started to use MatPlotLib, another graphics library, but for plotting. Using this, I managed to take location variables and plot them on the grid and create a nice canvas that could be used as a visual.



8.) Hospitalization:

An important part of pandemics is what we do to stop them. Hospitals have been a big help in slowing the spread of COVID19. They isolate the patients from society and also have better recovery rates. Because the goal of my project is not to show death rates and how we can change them, hospitalization in my code is just another factor that can isolate people. (Isolated (Top right) and Hospitalized (Bottom Left) agents overlap. The results of counting all of the types of agents are displayed at the top of each day.)

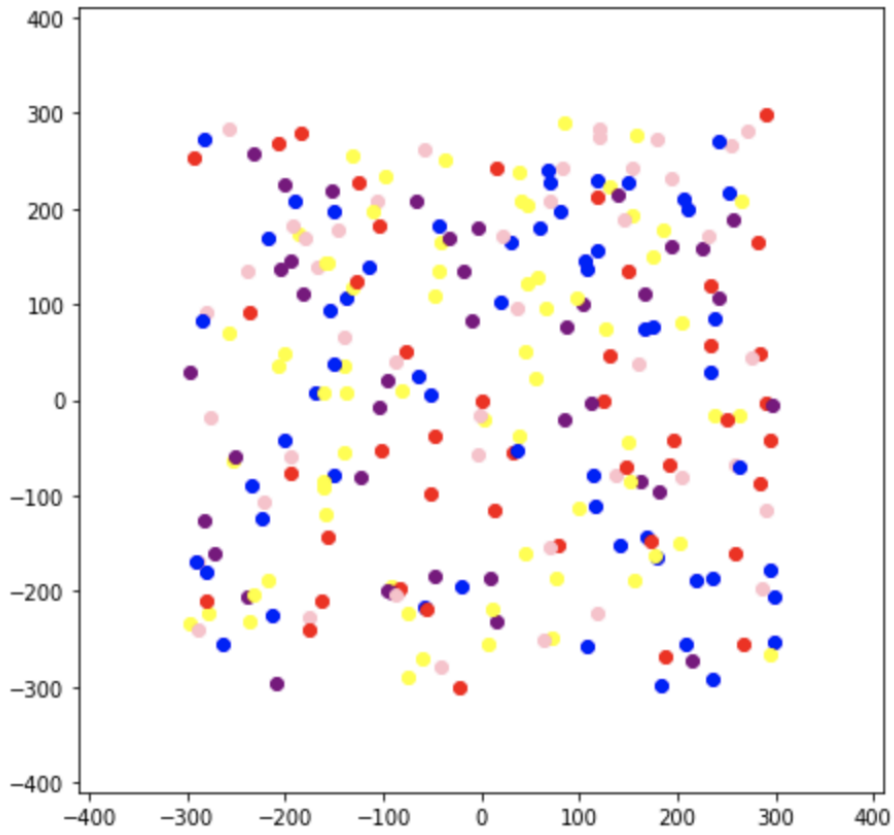
```
Day 14 >>> Susceptible: 94 | Infected: 152 | Removed: 6 | isolated: 76 | Hospitalized: 25
```



Plot First day:

The first day is plotted separately and just shows the untouched version of the environment. The color modification is always active on the first day and the one dot in the middle is always the infected agent. There will always be an even spread of agents because the movement module has not yet affected the agent's position.

```
Day-zero
```



Main Loop:

The Main loop is an infinite loop that runs until the count of infected agents hits 0. All the functions mentioned above get called in a specific order and a day is plotted at the end. Then, when the Main Loop breaks, all of the graphs are plotted. (See Appx. 3)

Human Personality

|~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~|

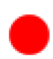
Human personality is an important concept that my program uses. It allows agents to gain their own style of everyday movement and mask wearing patterns. There are five personalities.

1. Stubborn
2. Obedient
3. Loud
4. Quiet
5. Teen

Each has different ways that they prefer to move.

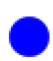
Stubborn:

Agents that are stubborn will appear on the scatterplots as a red dot. They like to be in the middle of the canvas, similar to a person going to a central gathering place such as downtown or a supermarket/mall. They can gather in small groups and only have a 10% chance of wearing a mask.



Obedient:

Agents that are obedient will appear on the scatterplots as a blue dot. They stay away from the central gathering place and normally just stay at home or wander the "town" on their own. They have a 100% chance of wearing a mask.



Loud:

Agents that are loud will appear on the scatterplots as a yellow dot. They like to be in the middle of the canvas, similar to a person going to a central gathering place such as downtown or a supermarket/mall. They have a 20% chance of wearing a mask.

Quiet:

Agents that are quiet will appear on the scatterplots as a purple dot. They avoid the central gathering place but don't stay in their homes very often they like to wander the "town". They have 20% chance of wearing a mask.

Teen:

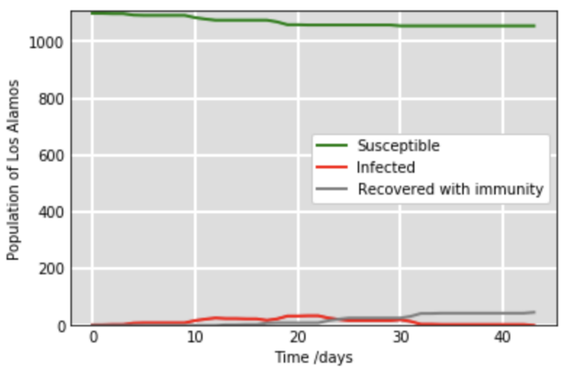
Agents that are teens will appear on the scatterplots as a pink dot. They like to be in the middle of the canvas, similar to a person going to a central gathering place such as downtown or a supermarket/mall. They also can do something called "party-gathering" which means that one teen is selected as the party host and most teens come to the party. They only have 4% chance of wearing a mask.

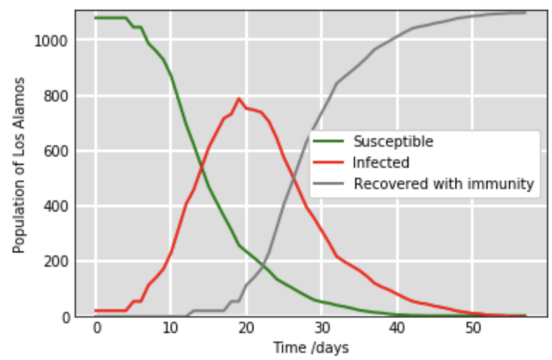
Results & Conclusion

|~~~~~~|

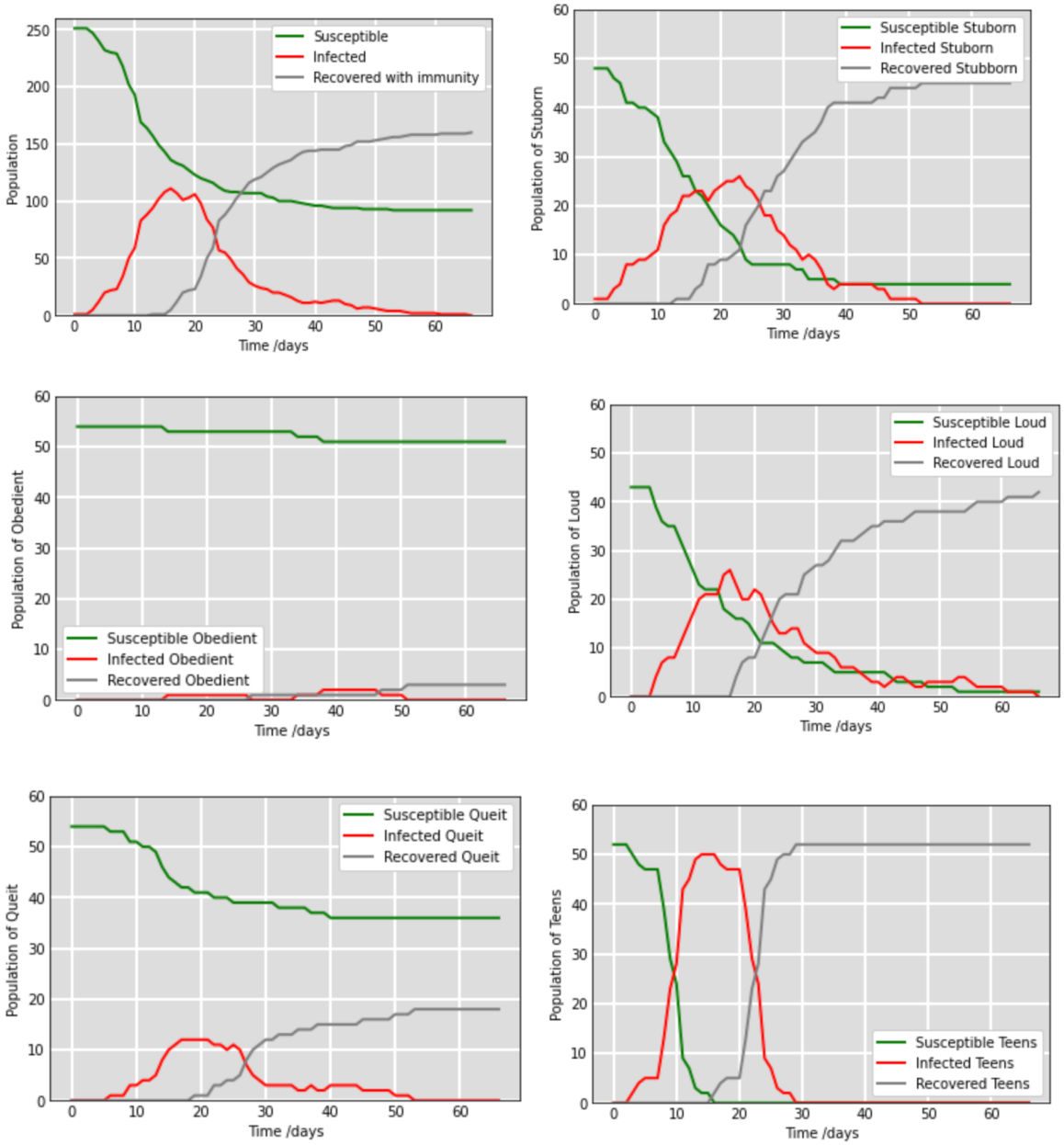
In conclusion, my code can show and educate people about the spread of this virus. Many people remain uneducated about how this virus spreads and how deadly it is. My project can help with this. When the government wants to stop a group of people from becoming too strong, one of the first things they will do is cut education. My project can help educate people to help them remain strong against the virus.

~~~~~~

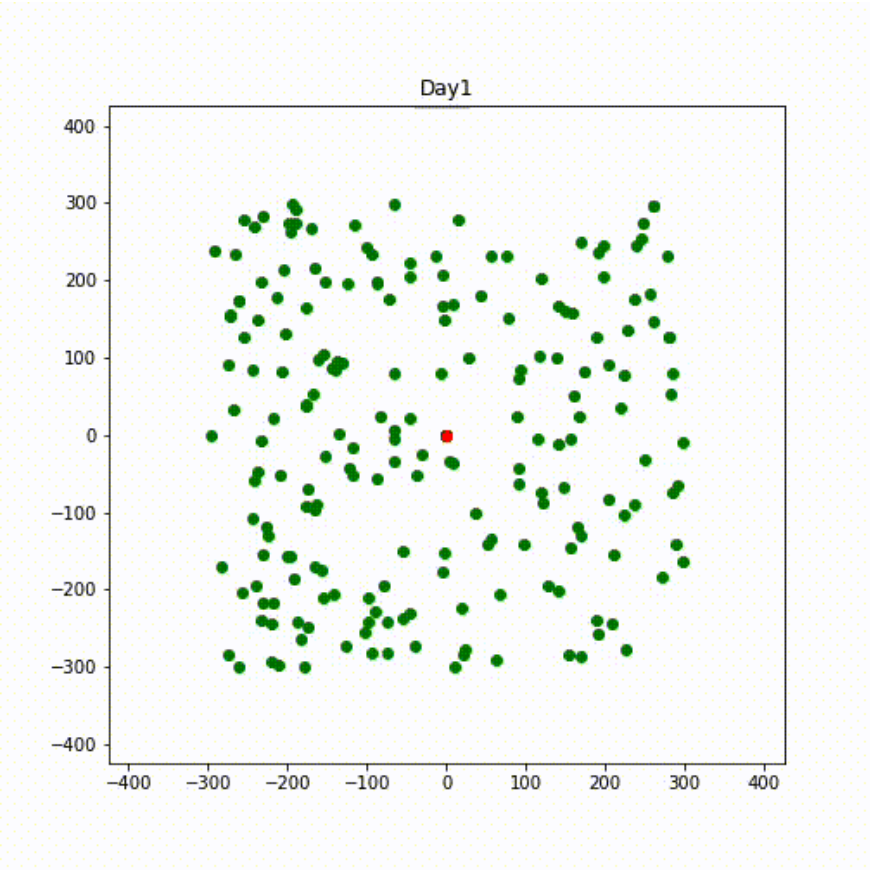




The graph on the left shows the total population graph when all people wore masks and socially distanced. The one on the right shows the graph for if people just lived like they used to. People can change. Change is necessary for a time like this. We have adapted. But many people are afraid of change. They like the way things are. But they're only looking at the small picture. When we look at the big picture, we can see that we will get back to normal quicker if we live differently for a short amount of time. If we try to go back to normal too quickly, then the amount of infected people shoots up. If we didn't try to keep reopening and waited for an extra 2 to 3 weeks, we would be able to get back to normal completely, rather than wearing masks in school because they opened too early. People can adapt to change. For me, transitioning from online school to in-person school was terrifying. I only lived in online school for about 7 to 8 months. Then changing my schedule was scary. But after only a week, I can't imagine being back to online school. People will adapt to change fast. It's part of our DNA. Changing will get us back to the norm faster and with less lives lost.
~~~~~~



These are 6 graphs that the simulation gave as a result to 1 run of the code. Some look scary, some look impossible. The 6th graph is the graph of the teens. The 3rd graph is of the obedient. There is a significant difference in shape. The only difference between the Loud personality and the Quiet personality is that Loud goes to the central gathering place, and Quiet don't. They have the same rates of wearing masks. The graphs are very different. Loud has a peak of infection more than 2 times of the Quiet peak of infection.

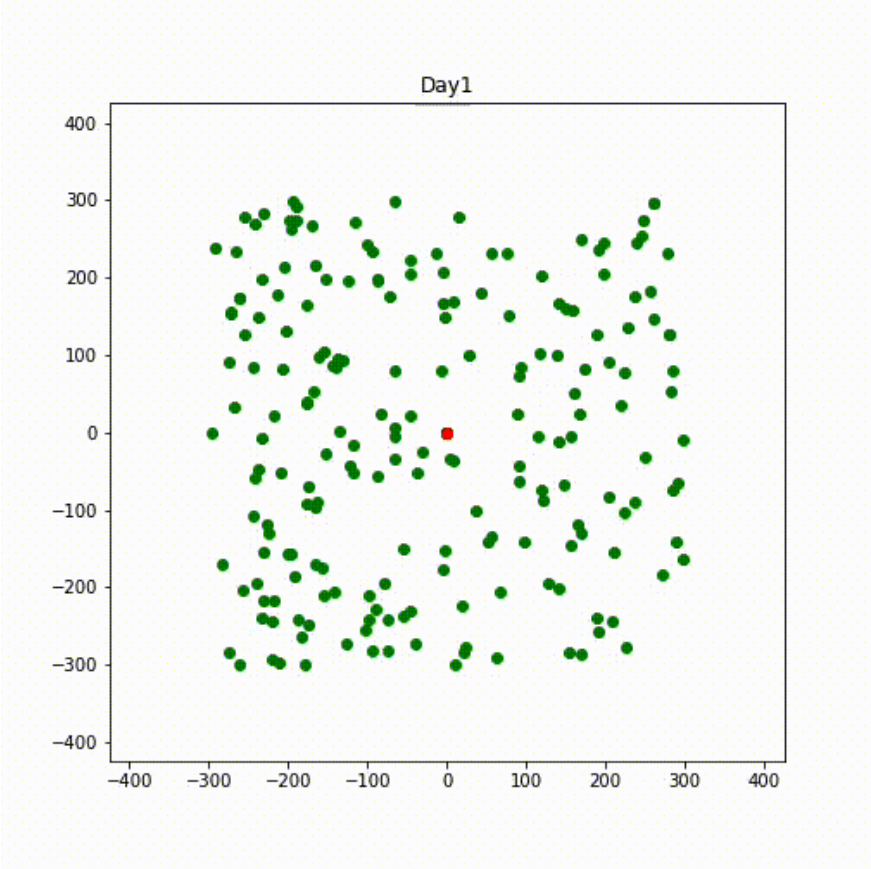


This is an animated collection of all of the scatterplots from one simulation.

Appendix

|~~~~~~~~~~~~~~~~~~~~|

Appendix (A): An animation of a collection of day image outputs.



Appendix (B):Movement Module.

```
def move5():
    global n, closure
    if closure == 'null':
        pass
    else:
        check_closure()
    hosbedlim = 25
    x = 0
    x5 = 0
    while x5 <= n + 1:
        people[x5].party = 'n'
```

```
    x5 = x5 + 1
x5 = 0
while x5 <= n + 1:
    if people[x5].personality == 't':
        pchance = random.randint(0,3)
        if pchance == 2:
            people[x5].party = 'y'
            print("Party to", x5)
            x5 = n + 10
        else:
            x5 = x5 + 1
    else:
        x5 = x5 + 1
x5 = 0

while x <= n + 1:
    if people[x].mdis == 1:
        x = x + 1
        continue
    skip = random.randint(0, 3)
    if skip == 2 or skip == 3:
        #print("DIDN'T SKIP")
        if people[x].movepat == "1":
            #print("moving 1")
            mc1 = random.randint(0,12)
            if mc1 == 0 or mc1 == 1 or mc1 == 2:
                #print("mini gathering")
                mp1 = random.randint(0, n + 1)
                people[x].location_x = people[mp1].location_x
                people[x].location_y = people[mp1].location_y
            elif mc1 == 3 or mc1 == 4 or mc1 == 5 or mc1 == 6:
                mx1 = random.randint(-10,10)
                my1 = random.randint(-10,10)
                people[x].location_x = people[x].location_x + mx1
                people[x].location_y = people[x].location_y + my1
                #print("moved to", people[x].location_x, people[x].location_y)

            else:
                people[x].location_x = 0
                people[x].location_y = 0
                #print("gone")
```

```
elif people[x].movepat == "2":
    #print("moving 2")
    mx2 = random.randint(-10,10)
    my2 = random.randint(-10,10)
    people[x].location_x = people[x].location_x + mx2
    people[x].location_x = people[x].location_x + my2
    #print("moved to", people[x].location_x, people[x].location_y)


elif people[x].movepat == '3':
    #print("moving 3")
    mc3 = random.randint(0,10)
    if mc3 == 1 or mc3 == 0 or mc3 == 2 or mc3 == 3 or mc3 == 4:
        people[x].location_x = 0
        people[x].location_y = 0
        #print("gone")

    else:
        mx3 = random.randint(-30,30)
        my3 = random.randint(-30,30)
        people[x].location_x = people[x].location_x + mx3
        people[x].location_y = people[x].location_y + my3

elif people[x].movepat == '4':
    #print("moving 4")
    mc4 = random.randint(0,1)
    if mc4 == 0:
        people[x].location_x = people[x].homexcoor
        people[x].location_y = people[x].homeycoor
    else:
        mx4 = random.randint(-50,50)
        my4 = random.randint(-50,50)
        people[x].location_x = people[x].location_x + mx4
        people[x].location_y = people[x].location_y + my4

elif people[x].movepat == '5':
    #print("moving 5")
    mc5 = random.randint(0,8)
    if mc5 == 1 or mc5 == 2 or mc5 == 3 or mc5 == 4 or mc5 == 5:
        x4 = 0
        while x4 <= n + 1:
            if people[x4].party == 'y':
```

```
                    people[x].location_x = people[x4].location_x
                    people[x].location_y = people[x4].location_y
                    #print("moved to party", people[x].location_x, people[x].location_y)
                    x4 = n + 10
                else:
                    x4 = x4 + 1
            else:
                people[x].location_x = 0
                people[x].location_y = 0
                #print("gone")


        else:
            #print("SKIPPED")
            if people[x].status == 'i' and people[x].asymm == 0 and people[x].inc <= 0 and x != 251 and
people[x].location_x != 400:
                hospc = random.randint(0,10)
                x1 = 0
                global hos
                hos = 0
                while x1 < n+1:
                    if people[x1].location_y == -400:
                        hos = hos + 1
                    x1 = x1 + 1
                hospc = 3
                if hospc == 3:
                    if hos >= hosbedlim:
                        x = x + 1
                        continue
                    else:
                        #print("hospital")
                        people[x].location_x = -400
                        people[x].location_y = -400
                        people[x].mdis = 1
                c10 = random.randint(0,7)
                if c10 == 2:
                    people[x].location_x = 0
                    people[x].location_y = 0
                    #print("moved to", people[x].location_x, people[x].location_y)
            else:
                mx10 = random.randint(-30,30)
                my10 = random.randint(-30,30)
```

```
            people[x].location_x = people[x].location_x + mx10
            people[x].location_y = people[x].location_y + my10


        check_b()
#           if vaccine == "true":
#               randomcure = random.randint(0, 3)
#               if randomcure == 2:
#                   people[x].status = "r"


    x = x + 1
```

Appendix (C): Main Loop.

```
global hos
num = 1000000 #maximum days simulation can progress to
num2 = 0 #current number of days simulated
day = 1
#scatter plotting
while num2 < num:
    #global vaccine
    x = 0
    while x <= n + 1:
        if people[x].status == 'r':
            people[x].mdis = 0
        x = x + 1
    global hos
    x = 0
    #incubation period calculation
    while x <= n + 1:
        people[x].inc = people[x].inc -1
        x = x + 1
#move here
    move5()
    check_s()
    x = 0
    #calculate the days infected
    while x <= n + 1:
        if people[x].status == 'i':
            people[x].daysin = people[x].daysin + 1
        else:
            pass
        x = x + 1
    x = 0
    #removing infected people
    while x <= n + 1:
        if people[x].daysin == 14:
            people[x].status = 'r'
        else:
```

```
        pass
    x = x + 1
#color change
x = 0
while x <= n + 1:
    if people[x].status == 's':
        people[x].color = 'green'
    elif people[x].status == 'i' and people[x].asymm == 1:
        #print("error")
        people[x].color = 'yellow'
    elif people[x].status == 'i':
        people[x].color = "red"
    else:
        people[x].color = 'gray'
    x = x + 1
#color change mod
x = 0
while x <= n + 1:
    if people[x].personality == 's':
        people[x].color = 'red'
    if people[x].personality == 'o':
        people[x].color = 'blue'
    if people[x].personality == 'l':
        people[x].color = 'yellow'
    if people[x].personality == 'q':
        people[x].color = 'purple'
    if people[x].personality == 't':
        people[x].color = 'pink'
    x = x + 1
#count
x = 0
sus = 0
inf = 0
rem = 0
Ssus = 0
Sinf = 0
Srem = 0
Osus = 0
Oinf = 0
Orem = 0
Lsus = 0
Linf = 0
Lrem = 0
Qsus = 0
Qinf = 0
Qrem = 0
Tsus = 0
Tinf = 0
Trem = 0
while x <= n+1:
    if people[x].status == 's':
        sus = sus + 1
    elif people[x].status == 'i':
        inf = inf + 1
```

```
    else:
        rem = rem + 1
    x = x + 1
x = 0
#count Stubborn
while x <= n + 1:
    if people[x].personality == 's':
        if people[x].status == 's':
            Ssus = Ssus + 1
        elif people[x].status == 'i':
            Sinf = Sinf + 1
        else:
            Srem = Srem + 1
        x = x + 1
    else:
        x = x + 1
x = 0
#count Obeidient
while x <= n + 1:
    if people[x].personality == 'o':
        if people[x].status == 's':
            Osus = Osus + 1
        elif people[x].status == 'i':
            Oinf = Oinf + 1
        else:
            Orem = Orem + 1
        x = x + 1
    else:
        x = x + 1

x = 0
#count Loud
while x <= n + 1:
    if people[x].personality == 'l':
        if people[x].status == 's':
            Lsus = Lsus + 1
        elif people[x].status == 'i':
            Linf = Linf + 1
        else:
            Lrem = Lrem + 1
        x = x + 1
    else:
        x = x + 1

x = 0
#count Quiet
while x <= n + 1:
    if people[x].personality == 'q':
        if people[x].status == 's':
            Qsus = Qsus + 1
        elif people[x].status == 'i':
            Qinf = Qinf + 1
        else:
            Qrem = Qrem + 1
```

```
            x = x + 1
        else:
            x = x + 1

    x = 0
    #count Teen
    while x <= n + 1:
        if people[x].personality == 't':
            if people[x].status == 's':
                Tsus = Tsus + 1
            elif people[x].status == 'i':
                Tinf = Tinf + 1
            else:
                Trem = Trem + 1
            x = x + 1
        else:
            x = x + 1
    x = 0
    ico = 0
    while x < n+1:
        if people[x].location_x == 400:
            ico = ico + 1
        x = x + 1
    x = 0
    hoscount = 0
    while x < n + 1:
        #print(people[x].location_y)
        if people[x].location_y == -400:
            hoscount = hoscount + 1
        x = x + 1
    print("Day", day, f">>> {bcolors.OKGREEN}Susceptible{bcolors.ENDC}:", sus, f"| {bcolors.WARNING}Infected{bcolors.ENDC}:", inf, f"|
{bcolors.GRAY}Removed{bcolors.ENDC}:", rem, f"| {bcolors.YELLOW}isolated{bcolors.ENDC}:", ico, f"|
{bcolors.OKCYAN}Hospitalized{bcolors.ENDC}:", hoscount)
    S.append(sus)
    I.append(inf)
    R.append(rem)
    Icolated.append(ico)
    Hospitalizeda.append(hoscount)
    Ss.append(Ssus)
    Is.append(Sinf)
    Rs.append(Srem)
    So.append(Osus)
    Io.append(Oinf)
    Ro.append(Orem)
    Sl.append(Lsus)
    Il.append(Linf)
    Rl.append(Lrem)
    Sq.append(Qsus)
    Iq.append(Qinf)
    Rq.append(Qrem)
    St.append(Tsus)
    It.append(Tinf)
    Rt.append(Trem)
    #print("check 2", people[x7].color)
```

```
#     print(S)
#     print(I)0-0
#     print(R)
#     print(Ss)
#     print(Is)
#     print(Rs)
#     print(So)
#     print(Io)
#     print(Ro)
#     print(Sl)
#     print(Il)
#     print(Rl)
#     print(Sq)
#     print(Iq)
#     print(Rq)
#     print(St)
#     print(It)
#     print(Rt)
    x = 0
    plt.figure(figsize=(7,7))
    while x <= n + 1:
        plt.title('Day' + str(day))
        plt.scatter(people[x].location_x, people[x].location_y, color = people[x].color, label = day)
        plt.xlim(-425,425)
        plt.ylim(-425,425)
        x = x + 1
    x = 0
    plt.savefig('Day-'+ str(day) + '.png')
    plt.show()
    day = day + 1
    num2 = num2 + 1
    if inf == 0:
        break
    if vaccine == "false":
        if inftotal >= n/2:
            print(f"{bcolors.OKBLUE}Vaccine being used{bcolors.ENDC}")
            vaccine = "true"
```

Appendix (D): The Code. The code consists of 800+ lines.

Because the code is too big and would give unnecessary amounts of pages to the pdf, I have uploaded my code to github and listed it as public viewing.
https://github.com/skim277/Covid19/blob/main/COVID19%20RSF%20(0316).ipynb

Bibliography

|~~~~~~~~~~~~~~~~~~~~~~~~|

1. https://intermountainhealthcare.org/blogs/topics/live-well/2020/04/whats-the-difference-between-a-pandemic-an-epidemic-endemic-and-an-outbreak/

2. https://ourworldindata.org/covid-vaccinations

3. https://en.wikipedia.org/wiki/COVID-19_pandemic

4. https://covid.cdc.gov/covid-data-tracker/#datatracker-home

5. https://onlinelibrary.wiley.com/doi/10.1002/adts.202000277

6. https://www.nature.com/articles/s41598-021-84055-6