SIMULATING THE POTENTIAL SPREAD OF COVID-19 IN NEW YORK CITY

OPR 9730 | Simulation Modeling & Analysis
Spring 2020

Farwa Ismail
Yuqiang Lu
Alexander Sonneborn
Amy Xu

Table of Contents

Abstract

Since COVID-19 emerged in Wuhan, China at the end of 2019, it has been spreading rapidly across the globe. At this time, it has been identified in over 170 countries and territories, with more than 600,000 people confirmed infected and 30,000 dead. Meanwhile, the number of infected individuals continues to rise; after outbreaks in Wuhan and Europe, New York City became one of the hardest-hit areas. Additionally, only a small proportion of people have received COVID-19 testing due to the lack of test kits, which means there are unconfirmed infections. In order to study the outbreak and spread of COVID-19 in NYC, we designed a simulation model incorporating factors such as commuting distance, age distribution, estimation and number of contacts per day. To build the model, we used historical data of COVID-19 infection and death rates in Wuhan along with SARS data from the outbreak in 2003, and adjusted our parameters using the most recent New York City COVID-19 data. Our simulation model is concise in structure, providing an estimate for the outbreak that may help the public understand and react more effectively over the coming weeks and months.

Keywords

COVID-19, SARS-CoV-2, coronavirus, epidemic, pandemic, disease simulation model, compartment model, SEIR model, New York City

Introduction

On December 31, 2019, Chinese authorities informed the WHO of pneumonia cases in Wuhan, Hubei Province, China, which stemmed from an unknown disease. The warning did not attract the public`s attention until January 7, 2020 when a novel coronavirus was identified as the cause of the outbreak. At that time, officials began to worry about a possible pandemic situation, though a total of just 44 cases had at that point been confirmed.

In the days after China reported the first death linked to the new coronavirus (January 9, 2020), the outbreak began spreading rapidly across the globe. Asian countries, including Thailand, Japan, and South Korea confirmed cases soon after. The United States reported its first confirmed case in Washington state on January 21, 2020. France confirmed Europe's first case on January 25, 2020, and Australia confirmed their first case that same day as well.

As the number of cases grew and spread to more countries, the WHO's Director-General declared the outbreak a public health emergency of international concern (January 30, 2020). By that time, nearly 10,000 people had been infected, and 200 of them had died from the virus. Around this time, the number of new cases outside of China began to surpass those within the country. On February 11, 2020, the WHO assigned the novel coronavirus its official name: the virus was named SARS-CoV-2, while the disease was named COVID-19. Over the next few days, cases of COVID-19 would be confirmed on every continent except Antarctica.

The WHO declared the COVID-19 outbreak a pandemic after the total number of COVID-19 cases surpassed 100,000 globally (March 7, 2020), with over 100 countries reporting cases. This call may have come too late, however. While it had taken three months for the number of global cases to reach 100,000, that number reached 200,000 cases 12 days later, 300,000 cases 3 days after that, and 400,000 cases in just two more days. By March 29, 2020, the global death toll from COVID-19 surpassed 30,000, and 600,000 people had been infected.

During this explosive rise in COVID-19 cases, New York City became one of the largest global hotspots. The first confirmed case in the city was reported on March 1, 2020, and by March 16, the city reported 463 cases and 7 deaths. However, just ten days later, the number of confirmed cases reached 23,122 (an increase of 49x), and 365 deaths (an increase of 51x). By March 26, 2020, New York City accounted for fully 28% of the 81,782 confirmed cases in the United States. The numbers are alarming but not entirely surprising when considering relevant factors: a large, high-density population, heavy usage of public transportation, a large tourism industry, and a major center for international business – all of which provide ample conditions for COVID-19 to spread rapidly.

In order to understand the rapidly changing COVID-19 outbreak and to predict future trends, we constructed a series of disease simulations based on SEIR compartment models. We used historical data from COVID-19 and filled in any gaps by using research on previous coronavirus outbreaks. In particular, we used parameters from the 2003 SARS outbreak, which enabled us to estimate essential parameters (e.g. reproductive number, identification rate, incubation duration) that weren't yet available for COVID-19. These values were combined with estimates of New York City population behavioral parameters (e.g. average commuter distance, person-to-person contact rates).

Literature Review

Coronaviruses

First identified in the 1930s, coronaviruses are a group of related RNA viruses that infect both mammals and birds. Coronaviruses cause varying levels of disease, from mild respiratory symptoms to fatal infections. Some coronaviruses make up a small fraction of what we refer to as the common cold, while others are far more lethal - varieties which include SARS, MERS and SARS-CoV-2 (COVID-19).

Severe Acute Respiratory Syndrome (SARS) is a viral respiratory disease caused by coronavirus SARS–CoV-1 (About Severe Acute Respiratory Syndrome (SARS), 2017). Symptoms include cough, fever, and diarrhea in the first or second week of illness. SARS was first reported in Asia in 2003, but was contained that same year after spreading to North America, South America and Europe. A total of 8,439 people was infected, with fatality rate of 9.6%.

Middle East Respiratory Syndrome (MERS) is a coronavirus disease caused by the MERS-CoV virus, which was first reported in Saudi Arabia in September 2012. The virus is similar to European bat coronaviruses, in which most cases of infection pass from animals to humans. Transmission among people is rare, occurring mostly between family members or in healthcare settings. As of December 2019, 2,468 cases of MERS have been confirmed worldwide, with 851 ending in fatality (a mortality rate of approximately 34.5%). Symptoms of MERS are similar to those of COVID-19, including fever, cough, and shortness of breath (About Middle East Respiratory Syndrome (MERS), 2019).

SARS-CoV-2 is the virus that causes COVID-19, and closely resembles SARS-CoV-1. What is different about COVID-19 compared to SARS and MERS, however, is that COVID-19 has a lower fatality rate but a higher infectious rate. SARS-CoV-2 is primarily spread during close contact with other people, often via small droplets produced by coughing, sneezing and talking. Despite the lower fatality rate, the overall number of deaths from COVID-19 far outweighs that from both SARS and MERS.

COVID-19 Research

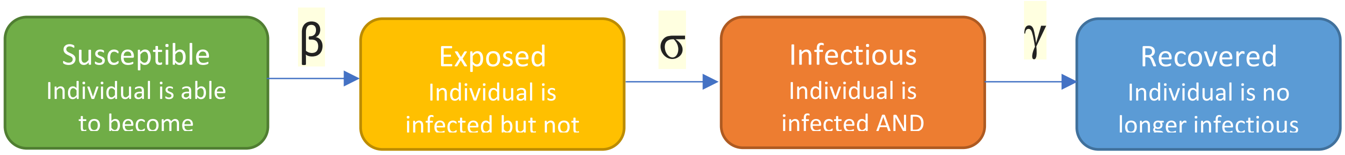
Based on early infection data from Wuhan and other Chinese cities, critical parameters such as basic reproductive number of infection (R_0) have been estimated for COVID-19. However, the values are different between studies due to the rapidly changing data and differing social parameters around the world. Another critical parameter, the dispersion parameter (k), has been estimated by quantifying the variability in the number of secondary cases, and can be interpreted as a measure of the impact of superspreading disease. However, this is not the preferred way to gather these measurements, and the accuracy remains in question. Finally, it is important to estimate the identification rate and the initial number of cases in Wuhan to make estimations, which would provide an index number to build models in New York City and elsewhere. Unfortunately, accurate counts may not be available for months or even years.

Compartment Models

Compartmental models are a class of models widely used for modeling the spread of infectious diseases. Compartment models treat each disease state (e.g. susceptible, infected, etc.) as a separate compartment, where compartments consist of a homogenous population of individuals. An individual can only be in one compartment at a given time, but can transition between states over time. At all times, the sum of each compartment will equal the total population, and all compartments together are modeled as a function of time.

The simplest of the compartment models is the *SIR* model, where *S* represents the number of susceptible individuals in a population, *I* represents the number of infected individuals in the population, and *R* represents the number of recovered individuals in the population. To simulate the spread of COVID-19 in New York City, an elaboration of the SIR model (called the SEIR model) will be used. For infectious diseases where there is an incubation period (the period where the patient is infected but is not yet infectious), it is more suitable to use this model. In SEIR, *S*, *I*, and *R* once again represent susceptible, infected, and recovered, and *E* represents the number of exposed (but not yet infectious) individuals.

The diagram below illustrates how individuals migrate between compartments in the SEIR model. For a set population, boxes represent the number of individuals in each compartment, and the arrows represent the possible movements. Movement between compartments are determined by three parameters: the infection rate (β), the incubation rate (σ), and the recovery rate (γ).


Susceptible
Individual is able to become
β
Exposed
Individual is infected but not
σ
Infectious
Individual is infected AND
γ
Recovered
Individual is no longer infectious

- **Infection rate (β)** represents the probability of infectious individuals transmitting the disease to susceptible individuals. This is estimated by finding the average number of human-to-human contacts a person in the study population has.
- **Incubation rate (σ)** represents the rate at which exposed individuals become infectious. It is estimated by determining the number of days between being infected and becoming infectious.
- **Recovery rate (γ)** represents the rate at which infectious individuals recover. The reciprocal, 1/ γ, is the average time an individual is infectious.

Over the course of a disease, the ratios between compartments change. At the initial outbreak, the susceptible population is large. As individuals are infected, however, the susceptible population decreases until the disease is unable to find new individuals to infect. The susceptible population can only grow through immigration or birth.

Agent-Based Models

For simple modeling of the spread of an infectious disease with a latent period (like COVID-19), common practice is to use the differential equation form of the SEIR model, as follows:

$$\frac{dS}{dt} = \mu(N - S) - \beta\frac{SI}{N} - \nu S$$

$$\frac{dE}{dt} = \beta\frac{SI}{N} - (\mu + \sigma)E$$

$$\frac{dI}{dt} = \sigma E - (\mu + \gamma)I$$

$$\frac{dR}{dt} = \gamma I - \mu R + \nu S$$

$$N = S + E + I + R$$

This method tracks expected proportions of the population in different states of the disease but doesn't have a mechanism for determining variances or higher-order moments of those proportions. One needs to be able to get a better picture of the variance in the predictions based on inherent noise in the model, and the differential form of the SEIR model doesn't lend itself to an easy stochastic generalization.

Agent-based models on the other hand, rely on repeated Monte-Carlo simulations, giving access to a large random sample of the sub-population to be able to do statistical analysis over. This helps with providing uncertainty estimates in our predictions and results.

Most importantly, a large disadvantage of the differential equation form of the SEIR model is that extensions of the model to account for real world population effects are non-trivial. For example, if we would like to implement social distancing, there is no way to do that other than to add ad-hoc terms to the model based off educated guesses of the phenomenon of social distancing. Often these guesses are obtained by nothing but mathematically analyzing agent-based models (where interpreting phenomenon and changing human behavior is trivial) and coming up with analytical expression for the expectation of aggregate variables. Hence, it felt much more natural to choose an agent-based model, allowing us to understand the mechanisms at play, directly analyze the effects of different policies and to evaluate the spreads of our predictions.

The Model

To build our initial model, we began with a 1x1 block (arbitrary units) to represent New York City. This block was then populated with N people (agents) that move randomly within the area. As an SEIR model, each agent can be in one of four disease states:

- **Susceptible**: agent has not yet been exposed
- **Exposed**: agent is exposed to the disease, but is not showing symptoms and is unable to spread the disease to others
  - exposed agents will become infected in $\widehat{t_\sigma}$ days, where $\widehat{t_\sigma} \sim$ Exponential(σ)
  - σ is the reciprocal of the average number of days it takes to develop symptoms
- **Infected**: the agent has developed symptoms and the ability to spread the disease
  - agents recover or die in $\widehat{t_\gamma}$ days, where $\widehat{t_\gamma} \sim$ Exponential(γ)
  - γ is the reciprocal of the average number of days it takes to recover or die
- **Removed**: the agent has either recovered from the virus or has died

The simulation progresses in time-steps of Δt days, which we set to be 0.25 earth-days (i.e. 6 hours). At each time-step, the agents move and their states are updated based on the information around them or based on random chance. At each time step several updates occur, each outlining different phenomena. We can break these into the “core phenomena” of the model, and the “policy/behavioral phenomena” which are adjusted to test various scenarios.

Core Mechanisms

- **Infection**: If a susceptible agent is within some radius ($r_{infection}$) of an infected person, a Bernoulli trial is done with probability $\beta \Delta$t. If it passes, the Susceptible agent is now Exposed
    - Note: β is the transmission rate
- **Gaining Symptoms**: At each time step an exposed agent has probability $\sigma \Delta$t of becoming infected
    - The transition time from Exposed to Infected is exponentially distributed, as Δt is small
- **Recovering/Dying**: At each time step an infected agent has probability $\gamma \Delta$t of dying or recovering from the disease (labelled as ‘Removed’)
- **Movement**: Agents experience a small acceleration in a random direction to make random movement possible.
- **MaxVelocity**: Agents’ speeds are capped at Vmax, even if an acceleration attempts to increase it
- **Walls**: Once agents hit the edge of the bounding box, they are reflected

Policy/Behavioral Mechanisms

Note: The control simulation did not incorporate any of the following features. Tables for the parameter values used for different models are available in the appendix.

Quarantine

When an agent population hits a threshold number of cases, $Q_{threshold}$ (defined as Infected + Removed population at time t), quarantining begins. At each time step after that, agents that have been infected for 2 days will be removed from the 1x1 box so they cannot infect others. Once recovered, they are returned to the box at their last recorded position.

At the beginning of a simulation, some proportion of the population, $Q_{proportion}$, can be labelled as “escapers”. These agents will not be affected by quarantine measures. This is a way to simulate ineffective efforts on the part of the government in being able to detect all people and quarantine them properly.

Social Distancing

Once an agent population reaches a threshold number of cases, $S_{threshold}$ (defined as Infected + Removed population at time t), social distancing begins. This is simulated by adding a repulsive acceleration factor between agents. For every person, P_j, within a social radius, r_{social}, of a Person Pi, there will be a repulsive force between them given by:

$$A_{ij} = \frac{-\alpha_{socialdistancing} \times \left(P_i(position) - P_j(position)\right)}{\left|P_i(position) - P_j(position)\right|^3}$$

This simulates a $1/r^2$ force between the agents. For the i^{th} person, this becomes:

$$A_i = -\sum_j \frac{\alpha_{socialdistancing} \times \left(P_i(position) - P_j(position)\right)}{\left|P_i(position) - P_j(position)\right|^3}$$

Here, $\alpha_{socialdistancing}$ is the parameter for the strength of social distancing and corresponds to how rigorously people avoid others.

At the start of a simulation, a proportion of the population, $S_{proportion}$, can be labeled as having decided to partake in social distancing practices. Those agents who do not participate will not feel the repulsive force. This emulates the real-world situation where some people do not follow social distancing guidelines.

Note: When implementing social distancing, the maximum velocity is reduced to 1/5th of its original limit due to the added acceleration causing a large amount of movement which would affect results unrealistically.

Central Market

At the beginning of a simulation, N_{market} squares with side length of 0.01 are positioned randomly within the 1x1 box. These are designated as “central markets”. Central markets represent locations where the public is more likely to be in close contact with one-another (e.g. concerts, grocery stores, classrooms, etc.). At each time step, a fraction of agents, $C_{proportion}$, decide to visit a central market. Those agents move to a random position inside the market for a period of n time-steps and then return to their previous position in the box.

Hyper Parameters

Hyper parameters are those that deal with simulation duration and granularity. These are not really parameters of the simulation per-se, so they are listed here separately. We chose to run the simulations for 250 days, and made sure that the time-step was short enough (0.25 days). The time step had to be relatively short because we simulated the time in recovery and latent periods as Bernoulli trials at each time step (with probability $p\Delta t$) until obtaining a positive result. This gave us a geometric distribution which as $\Delta t \rightarrow 0$, gives us a representative sample of an exponential distribution.

Parameter	Description	Vanilla Values
days	Number of Days the simulation is run	250
dt	Time Step of the simulation (in days)	0.25
N	Number of People simulated	1000

Meanwhile, choosing the number of people to simulate was a balancing act. We wanted sufficient interaction of people to give appreciable effects (on which we could test the effects of policies) while making sure each simulation could be run on a home laptop in a reasonable amount of time. We found N=1000 to be optimal for our situation.

Core Parameters

Core Parameters are the remaining parameters that the core functionality of the simulation depend on (e.g. epidemiological parameters like β, σ and γ and movement parameters).

Parameter	Description	Vanilla Values
beta	Transmission Probability (Probability that the virus is transmitted after a day of contact with an infected person)	0.2
sigma	Reciprocal of the expected number of days the virus is in its latent period	13
gamma	Reciprocal of the expected Recovery Time of the virus	114
radius	The radius (in units where the NYC is a 1x1 box) within which an infected person can transmit the disease.	0.03
maxvelocityset	Maximum velocity a person can attain when moving around	0.5
noise	Magnitude of the random kick (acceleration) per day on a person to simulate random motion.	0.05

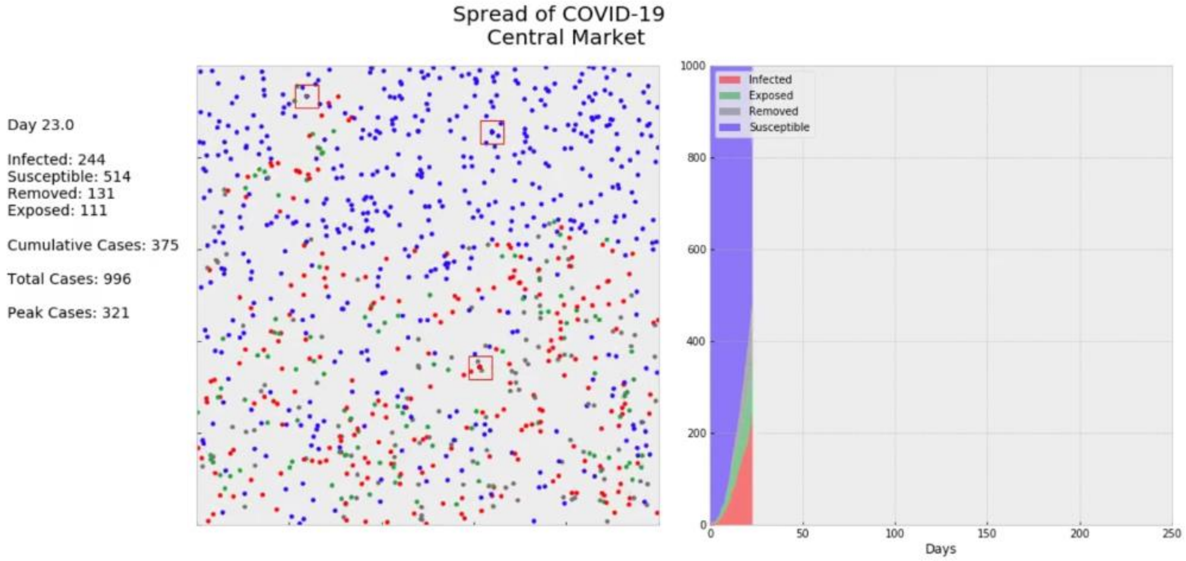
Normally, the value for β cannot be extracted by experiment without some modelled assumptions. For example, one could use the estimates for the R_0 value (the average number of people an infected person infects) for the epidemic in Wuhan, but the transmission probability value inferred from the data is extremely sensitive to the population density, mobility, and infrastructure specific to

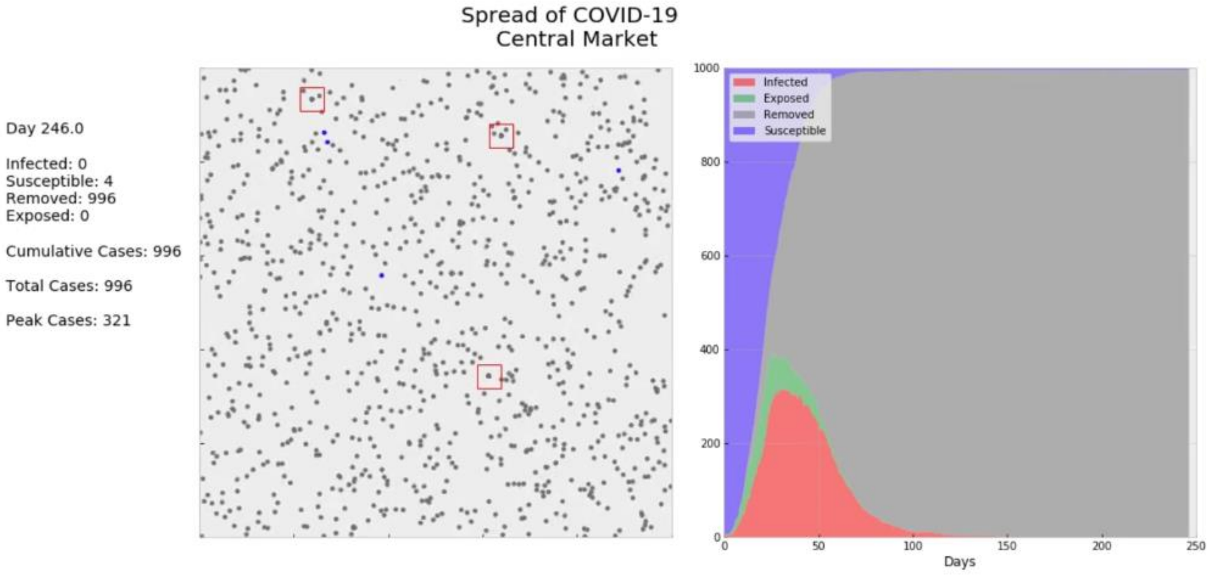
Wuhan. To account for this, we looked at the effects of different policies, which allowed us to compare the policies rather than determine the exact numbers that would be affected.

The values for σ and γ, on the other hand are only dependent on the disease itself. The time a person takes to recover or die from a disease and the time a person takes to gain symptoms is generalizable to most world populations. For our models, we set 3 days as the average time a person takes to gain symptoms and 14 days as the average time a person takes to recover or die from the disease.

Visualizations

We used *matplotlib* to visualize the results of our simulations. At the end of each run we saved the location and state of every person for every time step to generate animations. For each frame in an animation, we plotted a scatter plot and a stack plot. The scatter plot shows the position of each person at a given time, and colors them according to their state. Susceptible agents are in purple, Exposed are in green, Infected are in red, and Removed are grey. Central Markets are drawn as red squares. Agents may disappear for a short time in central market simulations, which indicates they are visiting the central market. Please see below for an example of an early (day 23) and late (day 246) frame from one animation.





Day 246.0

Infected: 0
Susceptible: 4
Removed: 996
Exposed: 0

Cumulative Cases: 996

Total Cases: 996

Peak Cases: 321

In the second frame (day 246), the simulation is near its end. In this case, most people are in the Removed state (represented as grey dots). On the right-hand side, we see the stacked area plot showing the proportion of people who are susceptible, exposed, infected and recovered against time.

Note that there is useful summary data show on the left side of the animation screen. The proportions of populations in different SEIR states are summarized, and these proportions change with time along with the cumulative cases. The total cases and peak cases over the course of the simulation are also displayed on the left.

Simulations

We ran several simulations with varying parameters (see Appendix B for details). Each simulation ran for 250 days with an agent population of 1,000. The number of agents within the simulation space was calculated to approximate the population density of New York City. Within each simulation, we recorded the total number infected, the number of cases at the disease peak and the duration of the disease outbreak.

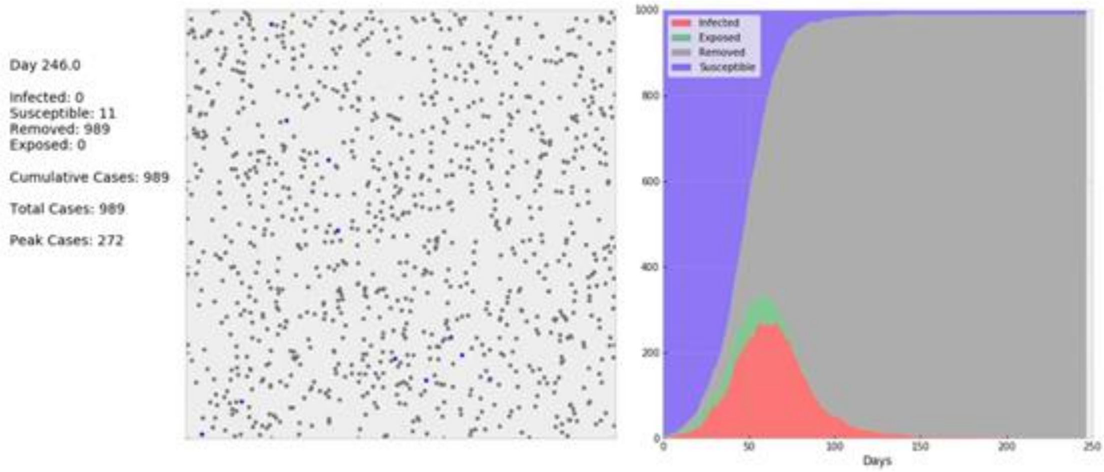
Given our finite medical resources and limited capabilities to adapt to rapid changes, the peak is a significant parameter for policy makers to ascertain whether they will be able to bear this burden. In general, the more the people that are sick together, the lower the chance that a given individual will be given proper medical attention and resources.

Another important thing to note is the length of the spread of the disease. A longer duration could imply greater loss to the economy. The peak and the duration of the disease are inversely related; therefore, we will be looking at ways that will allow us to reduce the size of this peak while not letting the disease spread for so long.

These models do not exactly determine how COVID-19 would spread under certain policies. Instead, they provide an indication of how, on average, the disease spread may be affected given different policy implementations.

Vanilla Run (Control)

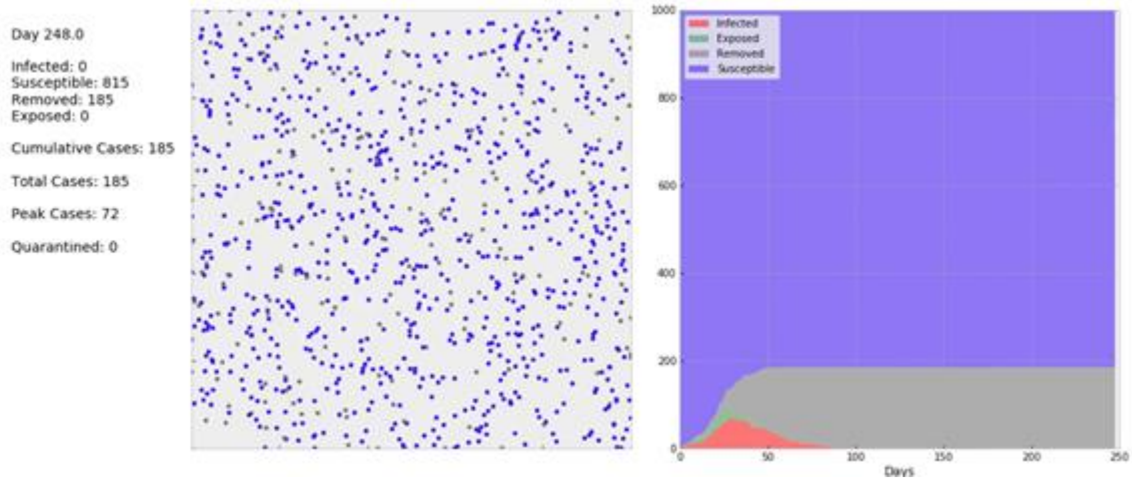
This model functions as a baseline for us to make sense of the effect of different policies, behaviors, and complexities. Under the control model, agents move randomly. According to this model (shown below), 98.9% of the agents end up getting the disease and 27.2% of the population is infected at the peak. The disease is almost gone by ~150 days.



Quarantine: 100%

This simulation models a policy in which everyone who gets the disease is quarantined. Quarantining begins after 100 people are infected and agents are quarantined 2 days after being diagnosed. This leads to a drastic change compared with the vanilla run. Only 18.5% of the people end up getting sick with only 7.2% of the people sick at the peak. The disease lasts for around 80 days.

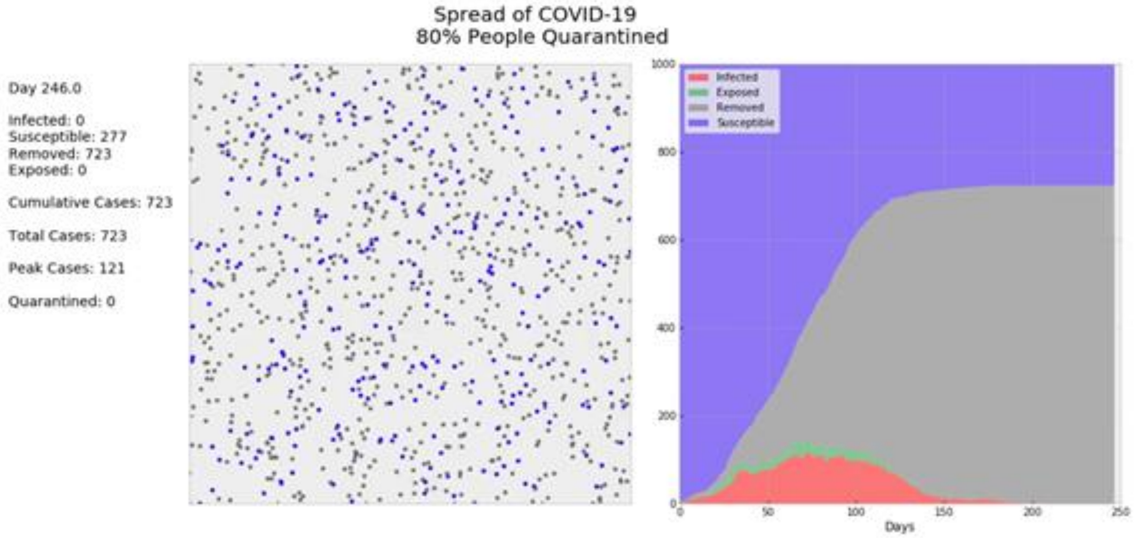
In reality, 100% quarantine is only possible if: 1. everybody infected shows symptoms, or 2. every person in the city is tested. Unfortunately, both of these conditions are unrealistic for COVID-19 (people with the disease often show symptoms after becoming infectious). Additionally, NYC's size and heterogeneity make it nearly impossible to test everyone. Therefore, we must investigate how varying degrees of quarantining affect the spread.

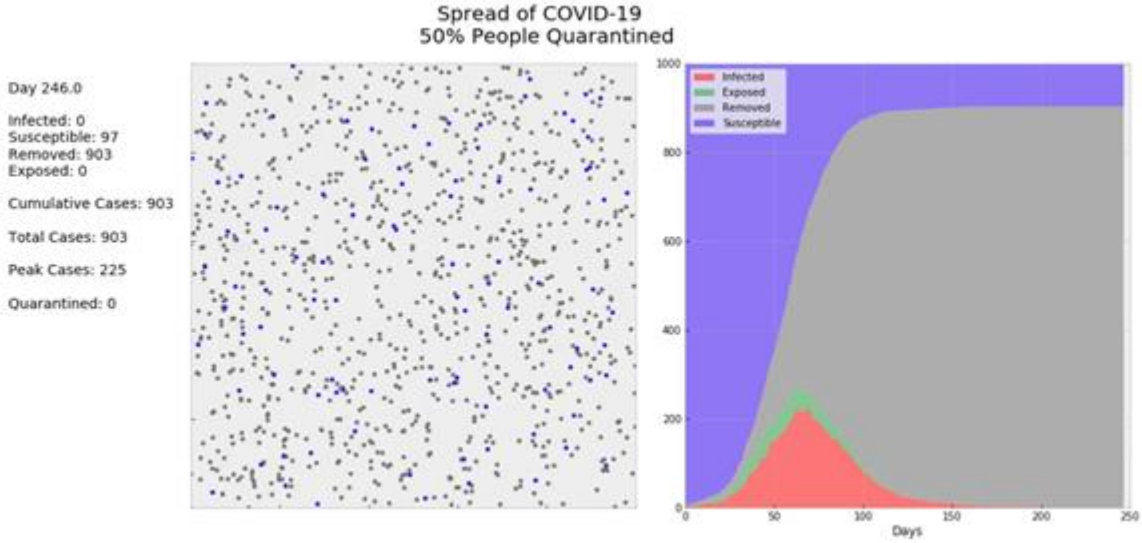


Quarantine: 80% and 50%

A more realistic representation of modeling COVID-19 quarantining in NYC would involve some people escaping the quarantining (i.e. people either did not show symptoms or were not tested). Thus, we developed simulations to represent how 80% (below, top) and 50% (below, bottom) quarantine levels might affect the spread.

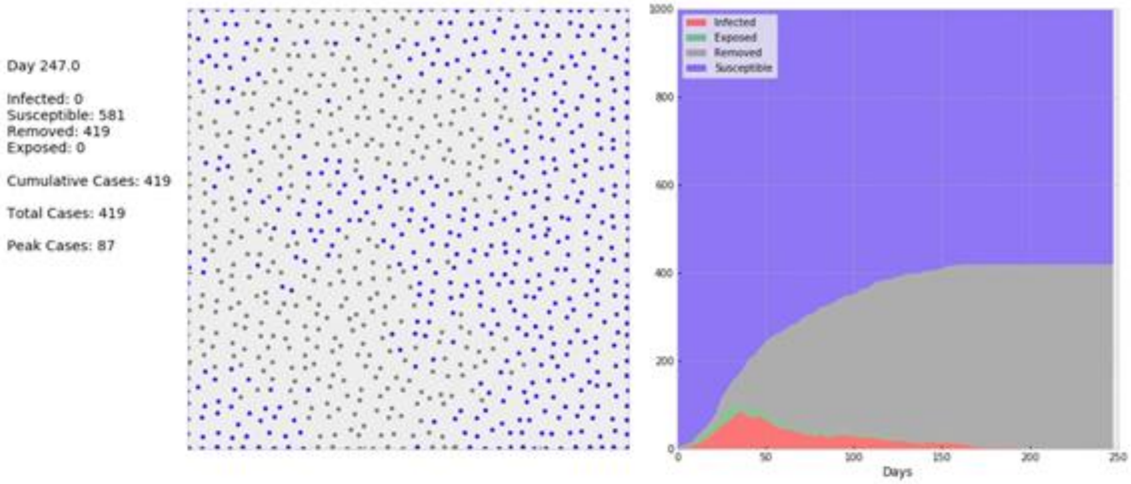
We see a lot of variability in these simulations. Where 80% quarantining comes across as an effective policy, 50% quarantining is only marginally better than the control. Therefore, it is extremely important to know what level of quarantine is possible for a given disease in a given location in order to determine if quarantining is an effective tool to combat disease spread.





100% Social Distancing

This simulation models the scenario whereby everyone exercises social distancing after 100 people are infected. Interestingly, only 41.9% of the agents get infected (remember that 10% had already been infected before the policy began). During the peak, 8.7% of agents are infected, however, the disease now takes more than 160 days to die down.



Day 247.0

Infected: 0
Susceptible: 581
Removed: 419
Exposed: 0

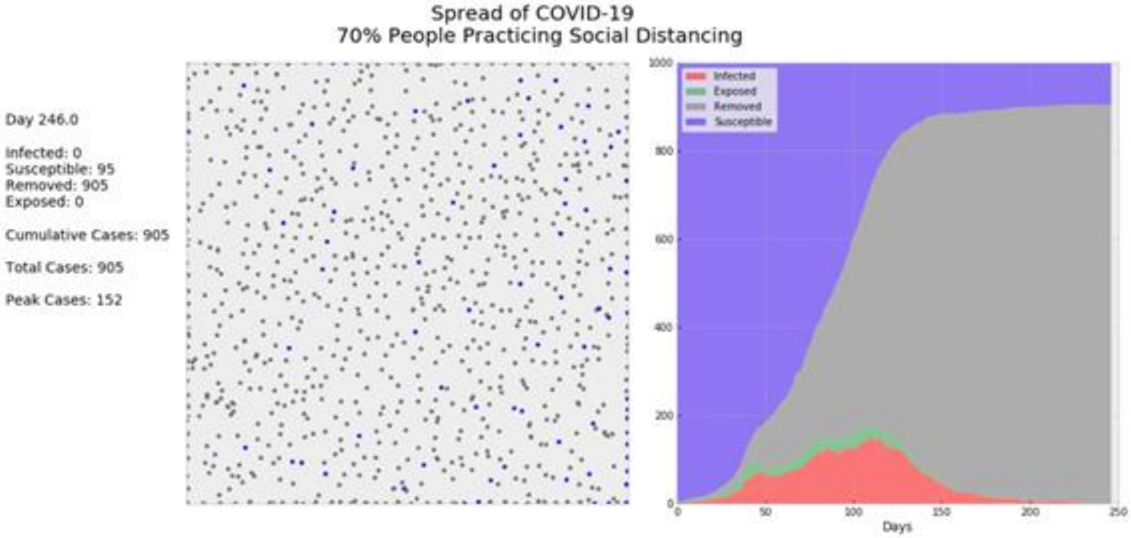
Cumulative Cases: 419

Total Cases: 419

Peak Cases: 87

70% Social Distancing

In this simulation, 30% of the population does not adhere to the social distancing policy. In this case, 90.5% of the agents are infected with 15.2% of agents sick at the peak. Along with that, the disease is present for more than 200 days.



Day 246.0

Infected: 0
Susceptible: 95
Removed: 905
Exposed: 0

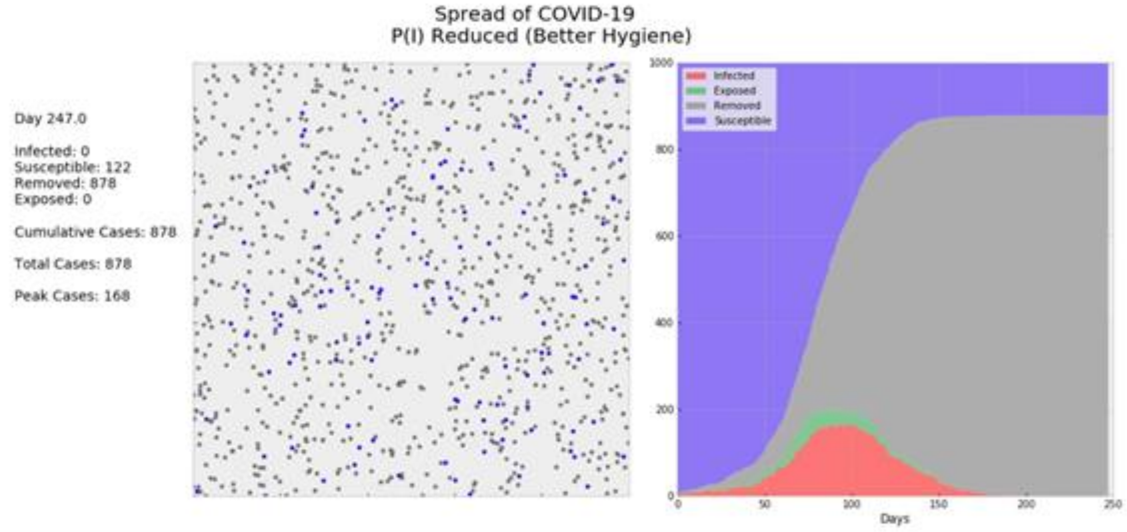
Cumulative Cases: 905

Total Cases: 905

Peak Cases: 152

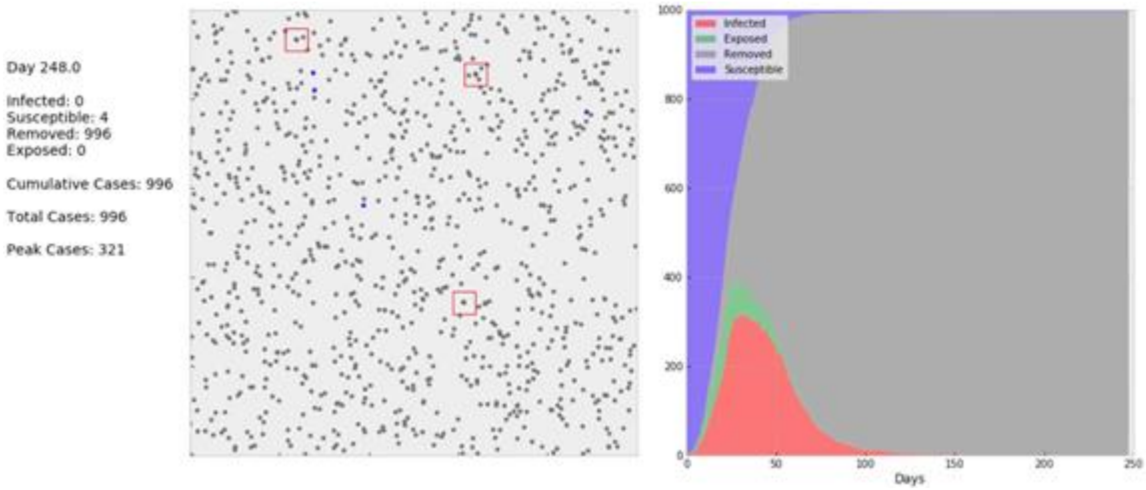
Better Hygiene

This simulation represents the situation where people become more careful about their hygiene (i.e. they wear masks, gloves, wash hands regularly, etc.). These actions tend to reduce the probability of transmitting disease. However, while taking better care of hygiene improves the simulation numbers, it is only marginally better than the control. Therefore, improving hygiene standards while everything else is kept the same, will have a minimal effect.



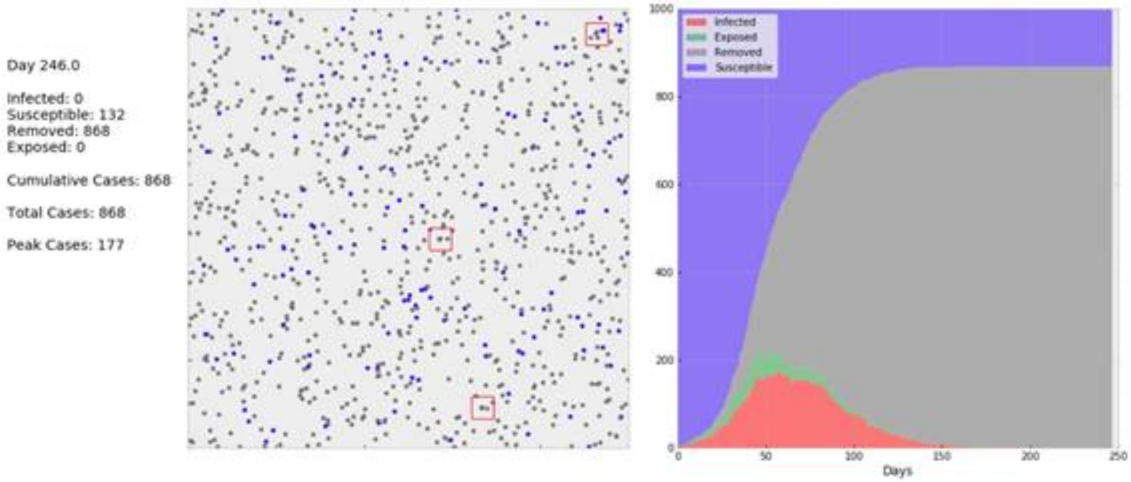
Central Market

In an attempt to better replicate life in NYC, we now introduce central markets, which represent places where people accumulate in bulk (e.g. schools, grocery stores, concerts, etc.). With 3 central markets in place, we see that 100% of agents get the disease with 32.1% of the population sick at the peak. The only reason the peak wasn't bigger, is because there weren't any people left to infect.



Central Market with Better Hygiene

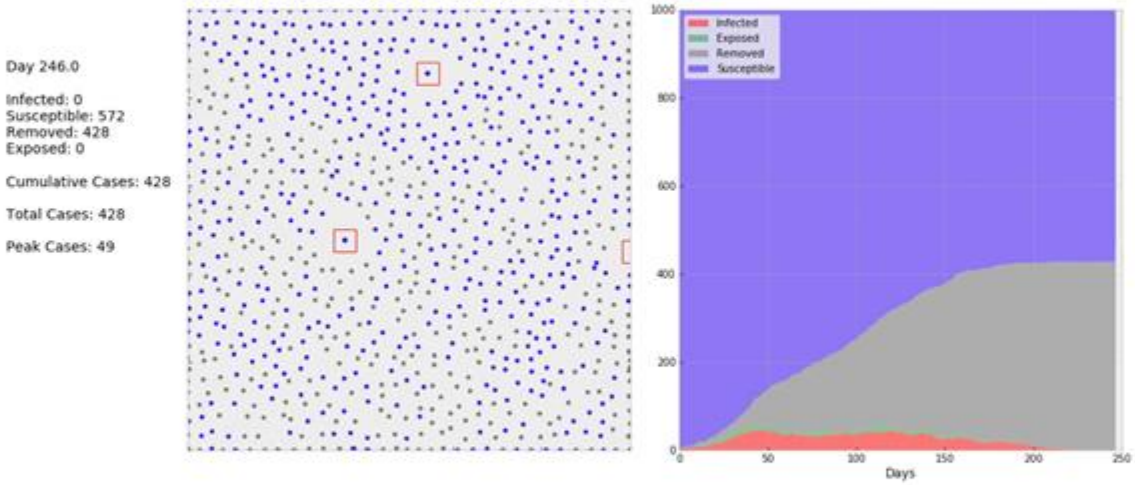
We are next interested in a situation in which people go about their daily business but exercise better hygiene. This should reduce the probability of transmission. In fact, while 86.8% of the agents get infected, there are fewer infected during the peak.



```
Day 246.0

Infected: 0
Susceptible: 132
Removed: 868
Exposed: 0

Cumulative Cases: 868

Total Cases: 868

Peak Cases: 177
```

Central Market with Better Hygiene and Social Distancing

Finally, we modeled a situation in which agents improve hygiene and exercise social distancing (after 100 are first infected). These actions spread out the curve to more than 200 days, but only 42.8% of agents get sick and only 4.9% of them are sick during the peak. This shows that combining behavioral changes can help mitigate the spread. One thing to note here, people are still going to the central markets so not everyone is exercising social distancing 100% of the time. Realistically, this makes sense because we all still need to buy groceries or get exercise.



```
Day 246.0

Infected: 0
Susceptible: 572
Removed: 428
Exposed: 0

Cumulative Cases: 428

Total Cases: 428

Peak Cases: 49
```

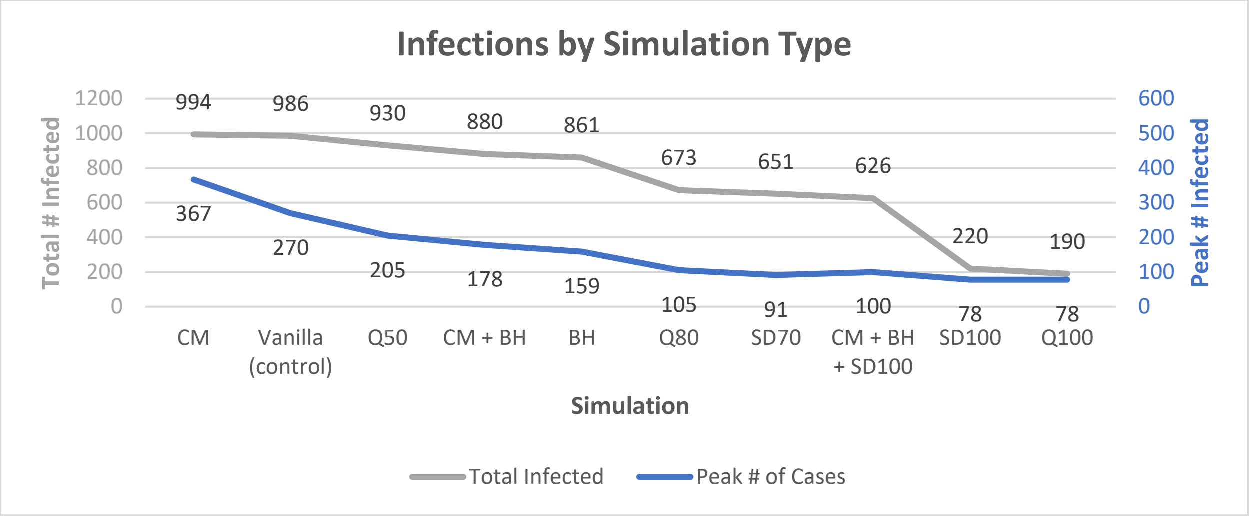
Analysis

Overview

After 50 replications were run for each simulation, the parameters were averaged and the following table was compiled. Interestingly, peak cases and total number of infected agents is highest in the same three scenarios (see below, in red). Similarly, peak cases and total infected agents were lowest for two of the three best simulations (see below, in green). This seems to indicate that there is a high level of correlation between these parameters.

#	Simulation	Peak # of Cases	Total Infected
I	Vanilla (control)	270	986
II	Quarantine 50	205	930
III	Quarantine 80	105	673
IV	Quarantine 100	78	190
V	Better Hygiene	159	861
VI	Social Distancing 70	91	651
VII	Social Distancing 100	78	220
VIII	Central Market	367	994
IX	Central Market + Better Hygiene	178	880
X	Central Market + Better Hygiene + Social Distancing 100	100	626

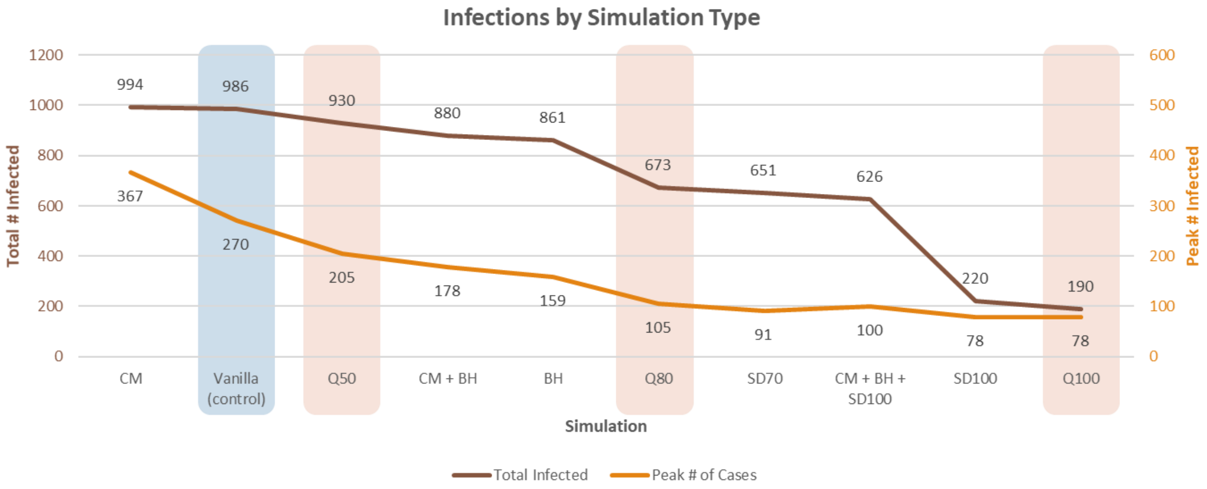
Despite the correlations, the graph of the infection results shows that the two variables exhibit different characteristics (see below). The total number of infected individuals drops slowly with policy changes until the “better hygiene” model, at which point it drops sharply. Then, it resumes its gradual decline until the “social distancing 100” simulation leads to a drop of almost 60%. Meanwhile, the peak number of cases drops rapidly throughout the first set of policy implementations, but the decline essentially stabilizes after the “80% quarantine” simulation.



The fact that the two curves exhibit different behaviors is not surprising. Policy changes that slow the spread of infection may increase the overall duration of the outbreak. As such, ff the number of cases does not drop to zero, these policy changes may not have much of an effect on total cases.

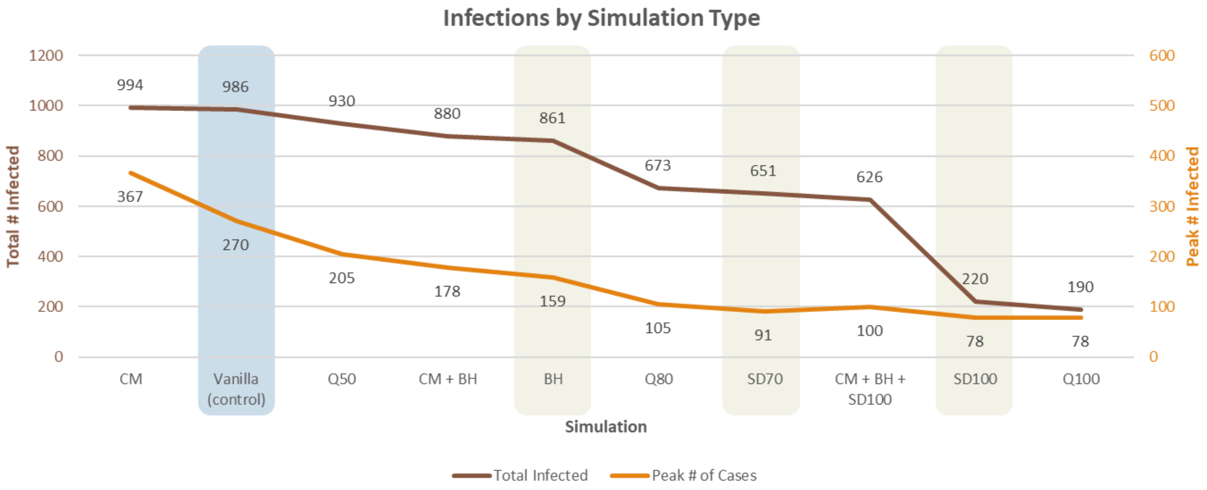
Quarantine Models

It is interesting to note how much variability is present across the quarantine models (shown below). As expected, the 100% quarantine model was most effective, but it is highly impractical for implementation in NYC and equally impractical based on the characteristics of the COVID-19 disease. On the other hand, 50% quarantine was just slightly better than the control, and is not likely to be worth the effort of implementation.



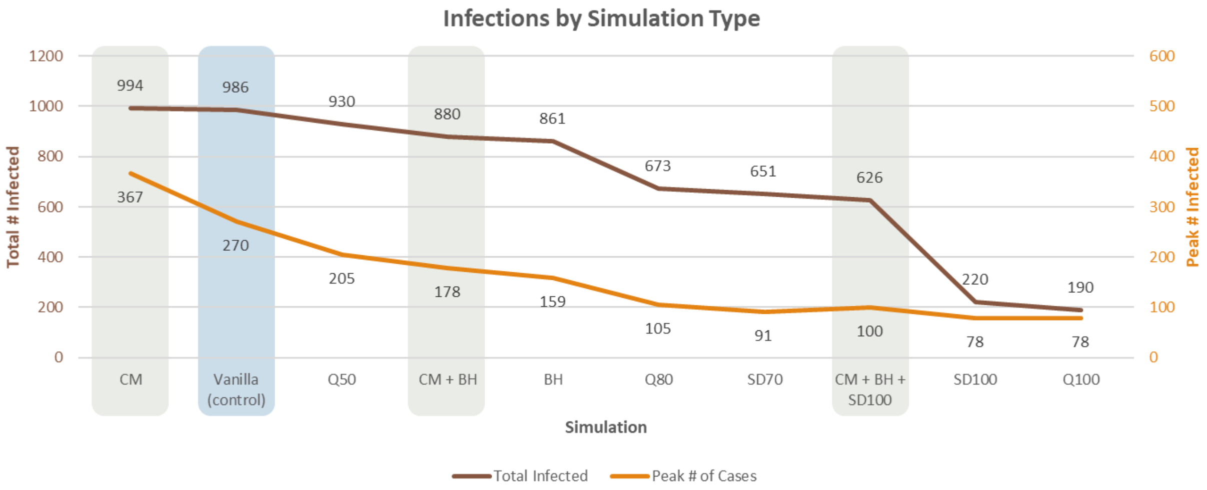
Behavioral Change Models

As opposed to the quarantine models, the behavioral change models appeared to have less variability. Between these models, the better hygiene model was the least effective of the three (see below). It is also noteworthy that the 100% social distancing model was almost as effective as the 100% quarantine model, and far more practical to implement for New York City and for COVID-19.



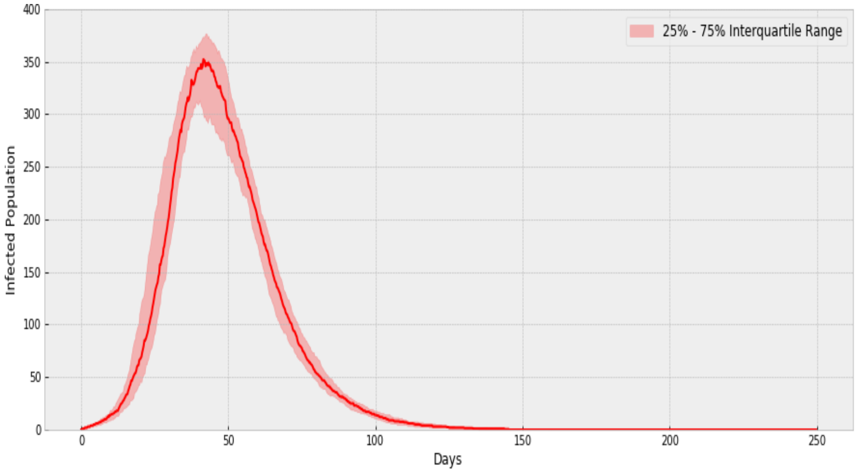
Central Market Models

The central market model is the most realistic model of pre-COVID-19 life. However, because there are more opportunities for individuals to be in close proximity, it is also the worst performing model that we ran. This is the main contributing factor for the central market + better hygiene performing worse than the better hygiene model alone. The takeaway here, however, is that even with central markets, adding better hygiene and social distancing initiatives can greatly slow the spread of COVID-19.

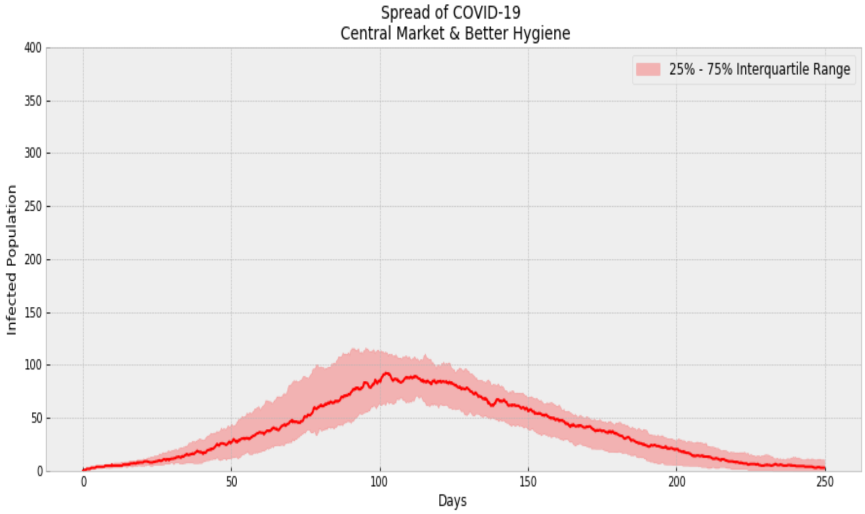


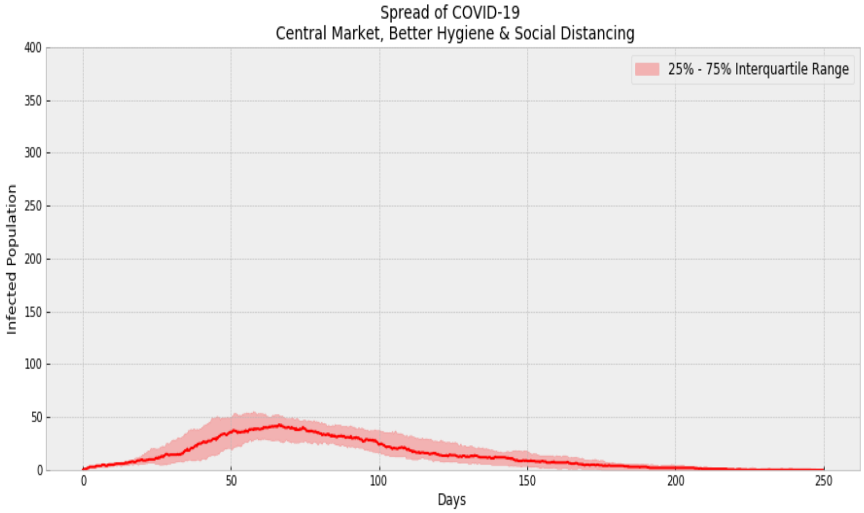
Curve Flattening

Based on the aggregate graph of peak infections, the central market model is very concerning when considering the potential for overrunning the health care system (see below).



However, we see that by increasing hygiene efforts and implementing social distancing, the peak drops significantly (see below). These drops have huge implications for the health care system, and may ultimately save lives.





Discussion & Recommendations

Number of peak cases is an important measure of effectiveness because an influx of infected people at one time will overwhelm the health care system. As such, it is important that the peak number of cases remain at a level at which there are sufficient resources for all patients. The second measure of effectiveness in our simulation models is the total number of infections at the end of the simulation period.

Focusing on the quantitative measurements from each run, 100% Quarantine shows the best results in terms of lowest number of peak cases and lowest number of total infections. However, 100%

quarantine means that everyone who is infected must be quarantined. The only way to quarantine everyone who is infected, is by testing the entire population. Providing COVID-19 tests for everyone is not possible in the real world due to the lack of resources.

While post-outbreak quarantine efforts may be inadequate in the current fight to control COVID-19, behavioral efforts are not. Both better hygiene and social distancing proved to have a positive effect, and this type of intervention can be implemented simultaneously.

Ultimately, our central market model demonstrated the importance of reducing the number of contact points, as the most effective way to slow the spread of COVID-19 is to reduce the number of interactions with other people. This means all events, stores, and businesses that are deemed to be non-essential should be closed. Social distancing should be enforced for 100% of the population where possible, as it proved to be effective in our models.

Limitations & Model Drawbacks

Because COVID-19 is still new to humanity, our knowledge is plagued with uncertainties. As a result, our model parameters are often based on assumptions. As research continues to focus on COVID-19, future analysts will be able to employ parameters with far more accuracy.

Based on the team's hardware and budget limitations, it was not feasible to implement some elements into our model. For instance, little information was available regarding transportation distances by method of transport, population movements into and out of NYC, and borough specific parameters. It was also not possible to implement contact-level differences based on housing type, and we were unable to collect information regarding the percentage of individuals who were deemed essential. Combined, these limitations only allowed us to get a very basic idea of what was and could happen with respect to COVID-19.

Finally, units of measures in our simulation were arbitrary given that the simulation is built on a population of 1000 individuals, and NYC has over 8 million people. However, the simulation results can still be used to measure relative successes and impact differences between initiatives. Conclusions can be drawn on the effectiveness of one initiative relative to another.

Conclusion

The most practical and effective way to reduce the number of peak cases and to reduce the total number of infections is a combination of practicing social distancing and safe hygiene. This means all non-essential people should remain at home, and must maintain a distance of 6 feet apart if leaving the house. At the same time, practicing good hygiene is important to combat the spread, and this means, washing hands more frequently, covering the mouth when coughing or sneezing, and wearing face masks in public.

When we started this project in February 2020, COVID-19 was not yet a crisis in New York City. There was no mention of social distancing, no businesses were closed, and no jobs lost. However, in March, 2020, New York State closed all non-essential businesses, and social distancing was enforced. As a result, our findings have essentially been validated - the state's actions align perfectly with our simulation results and recommendations.

References

About Middle East Respiratory Syndrome (MERS). (2019, August 2). Retrieved March 25, 2020, from Centers for Disease Control and Prevention: https://www.cdc.gov/coronavirus/mers/about/index.html

About Severe Acute Respiratory Syndrome (SARS). (2017, December 6). Retrieved March 25, 2020, from Centers for Disease Control and Prevention: https://www.cdc.gov/sars/about/fs-sars.html

Johns Hopkins Coronavirus Resource Center. (2020). Retrieved March 26, 2020, from https://coronavirus.jhu.edu/

Li, Qun, Xuhua Guan, Peng Wu, et al. (2020). Early Transmission Dynamics in Wuhan, China, of Novel Coronavirus–Infected Pneumonia. N Engl J Med, 382:1199-1207. doi: 10.1056/NEJMoa2001316

Liu, Tao, Jianxiong Hu, Jianpeng Xiao, et al. (2020). Time-varying transmission dynamics of Novel Coronavirus Pneumonia in China. bioRxiv. doi: 10.1101/2020.01.25.919787

Liu, Ying, Albert A. Gayle, Annelies Wilder-Smith and Joacim Rocklöv. (2020). The reproductive number of COVID-19 is higher compared to SARS coronavirus. Journal of Travel Medicine, 27(2). doi: 10.1093/jtm/taaa021

Read, Jonathan M., Jessica R.E. Bridgen, Derek A.T. Cummings, et al. (2020). Novel coronavirus 2019-nCoV: early estimation of epidemiological parameters and epidemic predictions. medRxiv. doi: 10.1101/2020.01.23.20018549

Riou, Julien and Christian L. Althaus. (2020). Pattern of early human-to-human transmission of Wuhan 2019 novel coronavirus (2019-nCoV), December 2019 to January 2020. Euro Surveillance, 25(4). doi: 10.2807/1560-7917.ES.2020.25.4.2000058

Sanderson, Grant. (2020). Simulating an Epidemic [Video File]. Retrieved from: https://youtu.be/gxAaO2rsdIs

Wu, Joseph T., Kathy Leung, Gabriel M. Leung. (2020). Nowcasting and forecasting the potential domestic and international spread of the 2019-nCoV outbreak originating in Wuhan, China: a modelling study. The Lancet, 395(10225):689-697. doi: 10.1016/S0140-6736(20)30260-9

You, Chong, Yuhao Deng, Wenjie Hu, et al. (2020). Estimation of the Time-Varying Reproduction Number of COVID-19 Outbreak in China. medRxiv. doi: 10.1101/2020.02.08.20021253

Appendix A: Attribute Descriptions

Agents/People

Attribute	Description
person	Person’s number
velocity	Velocity of the person
position	Position of the person on the 1x1 grid
state	Denotes a person’s state: 0 for “Susceptible”, 1 for “Exposed”, 2 for “Infected” and 3 for “Removed”
shopping	Whether the person is actively visiting a central market currently
SocialDistancing	Whether a person is participating in Social Distancing currently
quarantine	Whether a person is currently in quarantine
Shopping Location	Location of closest central market that the person will shop at.
Old Position	The position a person who is currently shopping was previously at, and hence, will return to after the trip to the central market
SocialDistancing	Whether a person has decided to participate in Social Distancing at all throughout the simulation.
Escaper	Whether this person will be able to avoid getting quarantined if they are infected.

Appendix B: Behavioral/Policy Parameters

Quarantine

Unless specified all other parameters are the same as the Vanilla control case.

	Quarantine Level		
Parameter	**100% Quarantine**	**80% Quarantine**	**50% Quarantine**
Quarantine	True	True	True
QuarantineTestingPeriod	2	2	2
EscapePercentage	0	0.2	0.5

Social Distancing

	Participation Level	
Parameters	**100% Participation**	**70% Participation**
SocialDistancingOn	True	True
SocialDistancingThreshold	100	100
socialradius	1	1
SocialDistanceFactor	0.05	0.05
SocialDistancePercentage	0.7	1

Central Markets

Parameters	Central Markets
shopping	True
shoppingprob	0.05
N_markets	3

Better Hygiene

We reduced the rate of transmission of the disease to simulate how better hygiene reduces this factor.

Parameters	Better Hygiene
beta	0.07

Central Market & Better Hygiene

Parameters	Central Market & Better Hygiene
beta	0.07
shopping	True

shoppingprob	0.05
N_markets	3

Central Market, Better Hygiene & Social Distancing

Parameters	**Central Market, Better Hygiene & Social Distancing**
beta	0.07
shopping	True
shoppingprob	0.05
N_markets	3
SocialDistancingOn	True
SocialDistancingThreshold	100
socialradius	1
SocialDistanceFactor	0.05
SocialDistancePercentage	0.7

Appendix C: Simulation Code

For the sake of completeness and transparency we have included the complete code for the simulations used in this report. There are two parts of the code: the first is for creating animations, and the second is for creating batches of simulations and analyzing them.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation
import matplotlib.patches as patches


RunName =  "Test"
Description = RunName;
TitlePhrase = "Spread of COVID-19 \n Central Market, Better Hygiene & Social Distancing"


##Simulation Parameters
days = 250; dt = 0.25; Nt = int(days/dt);
N = 1000;


##Infection Parameters
β = 0.2 #Transmission Probability (Probability that the virus is transmitted after a day of contact with an infected person)
σ = 1/3 #Reciprocal of the expected number of days the virus is in it's latent period
γ = 1/14 #Reciprocal of the expected Recovery Time of the virus
radius = 0.03 #Radius of transmission of an infected person


##Movement Parameters
maxvelocityset = 0.5; #Maximum Movement Velocity
noise = 0.05; #Noise level of Random kicks as people undergo brownian motion


##Social Distancing Parameters
SocialDistancingOn = False;
SocialDistancingThreshold = 100 #Infected People
socialradius = 1; #Radius of force for social distancing.
SocialDistanceFactor = 0.05; #Amount of Social Distancing
SocialDistancePercentage = 0.7 #Percentage of people following Social Distancing


##Shopping Parameters
shopping = False;
shoppingprob = 0.05; #Probability
N_markets = 3 # Number of Markets
```

```
##Quarantine Parameters
quarantine = False;
QuarantineTestingPeriod = 2; #Days before infected are put into quarantine
EscapePercentage = 0.5; #Percentage of people who can escape quarantine


Parameters = {'beta': β,'sigma':
σ,'gamma':γ,'radius':radius,'days':days,'dt':dt,'N':N,'maxvelocityset':maxvelocityset,'noise':noise, \
'SocialDistancingOn':SocialDistancingOn,'SocialDistancingThreshold':SocialDistancingThreshold,'socialradiu
s':socialradius, \
'SocialDistanceFactor':SocialDistanceFactor,'SocialDistancePercentage':SocialDistancePercentage, \

'shopping':shopping,'shoppingprob':shoppingprob,'N_markets':N_markets,'quarantine':quarantine,'QuarantineT
estingPeriod':QuarantineTestingPeriod, \
             'EscapePercentage':EscapePercentage,'Description':Description,\
              'RunName': RunName,'TitlePhrase':TitlePhrase}


def RunSimulation(Parameters):


    s = time.time()


    ##SimulationParameters
    days = Parameters['days'];
    dt = Parameters['dt'];
    Nt = int(days/dt);
    N = Parameters['N'];


    ##Infection Parameters
    β = Parameters['beta'] #Transmission Probability (Probability that the virus is transmitted after a
day of contact with an infected person)
    σ = Parameters['sigma'] #Reciprocal of the expected number of days the virus is in it's latent period
    γ = Parameters['gamma'] #Reciprocal of the expected Recovery Time of the virus
    radius = Parameters['radius'] #Radius of transmission of an infected person


    ##Movement Parameters
    maxvelocityset = Parameters['maxvelocityset']; #Maximum Movement Velocity
    noise = Parameters['noise']; #Noise level of Random kicks as people undergo brownian motion


    ##Social Distancing Parameters
    SocialDistancingOn = Parameters['SocialDistancingOn'];
    SocialDistancingThreshold = Parameters['SocialDistancingThreshold']; #Infected People
    socialradius = Parameters['socialradius']; #Radius of force for social distancing.
    SocialDistanceFactor = Parameters['SocialDistanceFactor']; #Amount of Social Distancing
    SocialDistancePercentage = Parameters['SocialDistancePercentage']; #Percentage of people following
Social Distancing

    ##Shopping Parameters
    shopping = Parameters['shopping'];
    shoppingprob = Parameters['shoppingprob']; #Probability
    N_markets = Parameters['N_markets'] # Number of Markets

    ##Quarantine Parameters
    quarantine = Parameters['quarantine'];
    quarantineOn = False;
    QuarantineTestingPeriod = Parameters['QuarantineTestingPeriod']; #Days before infected are put into
quarantine
    EscapePercentage = Parameters['EscapePercentage']; #Percentage of people who can escape quarantine

    ##Visualization Parameters
    TitlePhrase = Parameters["TitlePhrase"]

    People = np.zeros((N,Nt),dtype = [('person',np.int,1),
                                      ('velocity',np.float,2),
                                      ('position', np.float, 2),
                                      ('state', np.int, 1),
                                      ('shopping',np.bool,1),
                                      ('SocialDistancing',np.bool,1),
                                      ('quarantine',np.bool,1)])

    PeopleStats = np.zeros(N,dtype = [('ShoppingLocation',np.float,2),
                                      ('OldPosition',np.float,2),
                                      ('SocialDistancing',np.bool,1),('Escaper',np.bool,1)])

    People['person'][:,0] = np.arange(0,N,1);
    People['position'][:,0] = np.random.rand(N,2);
    People['state'][1:6,0] = 1;
    People['velocity'][:,0] = np.zeros((N,2));

    if SocialDistancingOn:
        PeopleStats['SocialDistancing'] = False;
        PeopleStats['SocialDistancing'][np.random.choice(N, int(SocialDistancePercentage*N),
replace=False)] = True
    if quarantineOn:
        PeopleStats['Escaper'][np.random.choice(N, int(EscapePercentage*N), replace=False)] = True



    CentralMarkets = np.zeros(N_markets,dtype=[('bottom-left',np.float,2),('top-right',np.float,2)]);
    CentralMarkets['bottom-left'] = np.random.rand(N_markets,2);
    CentralMarkets['top-right'] = CentralMarkets['bottom-left'] + np.array([0.05,0.05]);

    QuarantineCounts = np.zeros(Nt,dtype=np.int);

    SocialDistancing = False;

    for day in range(1,Nt):
        update_progress(day / Nt)
        People['state'][:,day] = People['state'][:,day-1].copy();

        #Check TotalCases to turn on SocialDistancing
        TotalCasesAsOfToday = ((People['state'][:,day-1] == 2) | (People['state'][:,day-1] == 3)).sum();
        if ((TotalCasesAsOfToday >= SocialDistancingThreshold) and (SocialDistancingOn)):
            SocialDistancing = True;
        if ((TotalCasesAsOfToday >= 100) and (quarantineOn)):
            quarantine = True;
        FinalSocialDistanceFactor = SocialDistancing*SocialDistanceFactor;

        #print(quarantine, TotalCasesAsOfToday)

        for person in range(N):
            #I->R
            if (People['state'][person,day-1] == 2):
                if np.random.rand() < γ*dt:
                    People['state'][person,day] = 3;
            #E->I
            if (People['state'][person,day-1] == 1):
                if np.random.rand() < σ*dt:
                    People['state'][person,day] = 2;

            #Infections
            if ((People['state'][person,day-1] == 2)):
                #Find all people close by
                ClosebyPeopleIndices = np.argwhere(np.linalg.norm(People['position'][:,day-1] -
People['position'][person,day-1],axis=1) < radius).flatten()
                #Only pick the susceptible people
                Susceptibles = ClosebyPeopleIndices[np.argwhere((People['state'][ClosebyPeopleIndices,day-1] == 0) & (ClosebyPeopleIndices != person))];
                #Roll a die for each person
                InfectedPeopleIndices = np.argwhere(np.random.rand(Susceptibles.size) < θ*dt).flatten();
                #Infect the people
                People['state'][Susceptibles[InfectedPeopleIndices],day] = 1;


            #Make a person a shopper
            if ((shopping) and (np.random.rand() < shoppingprob*dt) and (~People['shopping'][person,day-1]) and (~People['quarantine'][person,day-1])):
                oneday = int(1/dt)
                People['shopping'][person,day:(day+oneday)] = True
                centers = 0.5*(CentralMarkets['bottom-left'][:]+CentralMarkets['top-right'][:])
                closestindex = np.argmin(np.linalg.norm(centers - People['position'][person,day-1],axis=1)).flatten()
                PeopleStats['OldPosition'][person] = People['position'][person,day-1];
                People['position'][person,day] = np.random.rand(2)*(CentralMarkets['top-right'][closestindex[0]]-CentralMarkets['bottom-left'][closestindex[0]]) + CentralMarkets['bottom-left'][closestindex[0]]
                PeopleStats['ShoppingLocation'][person] = centers[closestindex[0]]


            #Quarantine a person
            if ((quarantine) and (People['quarantine'][person,day-1] == False) and (People['state'][person,day-1] == 2) and (~PeopleStats['Escaper'][person]) and ((People['state'][person,:day] == People['state'][person,day]).sum()*dt >= QuarantineTestingPeriod)):
                PeopleStats['OldPosition'][person] = People['position'][person,day-1];
                People['position'][person,day] = np.array([-1,-1]);
                print("Quarantined person ", person, "Who was infected for", (People['state'][person,:day] == People['state'][person,day]).sum()*dt, " days")
                People['quarantine'][person,day] = True;


            #Movement
            acceleration = np.array([0.,0.])


            if ((~People['shopping'][person,day-1]) and (~People['quarantine'][person,day-1])):
                if PeopleStats['SocialDistancing'][person]:
                    ##Social Distancing
                    ClosebyPeoplIndices = np.argwhere(np.linalg.norm(People['position'][:,day-1] - People['position'][person,day-1],axis=1) < 0.2).flatten()
                    dists =np.linalg.norm(People['position'][ClosebyPeoplIndices,day-1] - People['position'][person,day-1],axis=1);
                    distcube = (dists**3).clip(min=0.00001)
                    accels = -FinalSocialDistanceFactor*(People['position'][ClosebyPeoplIndices,day-1]-People['position'][person,day-1]) / distcube[:,None]
                    accels = accels.mean(axis=0)
                    acceleration += accels

                #RandomMovement
                acceleration += noise*2*(np.random.rand(2)-0.5);

                #Updates
                People['velocity'][person,day] = People['velocity'][person,day-1] + acceleration*dt

                #Heavy velocity Damping
                if ((PeopleStats['SocialDistancing'][person]) and (SocialDistancing)):
                    maxvelocity = maxvelocityset/2;
                else:
                    maxvelocity = maxvelocityset;
                vnorm = np.linalg.norm(People['velocity'][person,day])
                People['velocity'][person,day] =
np.clip(vnorm,a_min=None,a_max=maxvelocity)*(People['velocity'][person,day]/vnorm)

                #Position Update
                People['position'][person,day] = People['position'][person,day-1] +
(1/5)*People['velocity'][person,day-1]*maxvelocity*dt;

                #Bouncing off walls
                if ((People['position'][person,day][0] < 0)):
                    People['velocity'][person,day][0] = -People['velocity'][person,day][0]
                    People['position'][person,day][0] = 0
                if ((People['position'][person,day][1] < 0)):
                    People['velocity'][person,day][1] = -People['velocity'][person,day][1]
                    People['position'][person,day][1] = 0;
                if ((People['position'][person,day][0] > 1)):
                    People['velocity'][person,day][0] = -People['velocity'][person,day][0]
                    People['position'][person,day][0] = 1
                if ((People['position'][person,day][1] > 1)):
                    People['velocity'][person,day][1] = -People['velocity'][person,day][1]
                    People['position'][person,day][1] = 1
            else:
                if People['shopping'][person,day-1]:
                    People['position'][person,day] = PeopleStats['ShoppingLocation'][person]
                if People['quarantine'][person,day-1]:
                    #print("Quarantined person ", person)
                    People['position'][person,day] = np.array([-1,-1])

             #If a person was shopping yesterday and isn't today, take them back
            if (People['shopping'][person,day-1] and ~People['shopping'][person,day]):
                People['position'][person,day] = PeopleStats['OldPosition'] [person];

            #If in Quarantine keep until removed
            if ((quarantine) and (People['quarantine'][person,day-1] == True)):
                if People['state'][person,day-1] == 2:
                    People['quarantine'][person,day] = True;
                else:
                    People['position'][person,day] = PeopleStats['OldPosition'][person]
                    People['quarantine'][person,day] = False;

        QuarantineCounts[day] = People['quarantine'][:,day].sum();

    print(time.time()-s," seconds")

    ## This part calculates the number of susceptible, exposed, infected and removed people for each point
in time and puts
    # them into a dataframe.
    df = pd.DataFrame({});

    #For each timestep
    for i in range(Nt):
        #Count the unique states and their counts
        unique, counts = np.unique(People['state'][:,i], return_counts=True)
        cases = dict(zip(unique, counts))

        #Initialise a dictionary which has the data we want and fill in the numbers
        NumberOfCases = {"TimeStep": i, "Days": np.round(i*dt,3), "Susceptible": 0, "Exposed":
0,"Infected": 0,"Removed": 0,"Quarantined":QuarantineCounts[i]}
        for k in unique:
            if k == 0.0:
                NumberOfCases["Susceptible"] = cases[k];
            if k == 1.0:
                NumberOfCases["Exposed"] = cases[k];
            if k == 2.0:
                NumberOfCases["Infected"] = cases[k];
            if k == 3.0:
                NumberOfCases["Removed"] = cases[k];

        #Add that dictionary to the dataframe.
        df = df.append(NumberOfCases,ignore_index=True);
    ##This Part Makes Plots
    #print(df[df['TimeStep'] == df['TimeStep'].last()]['Removed'])


    plt.style.use("bmh")


    fig = plt.figure(figsize=(20, 8))
    grid = plt.GridSpec(1, 12, hspace=1, wspace=1)
    ax1 = fig.add_subplot(grid[:,2:7], xticklabels=[],yticklabels=[])
    ax2 = fig.add_subplot(grid[:,7:12])
    TextArea = fig.add_subplot(grid[:,0:2],xticklabels=[], yticklabels=[],frameon=False)
    TextArea.set_facecolor('w')
    TextArea.grid(False)
    TextArea.get_xaxis().set_visible(False)
    TextArea.get_yaxis().set_visible(False)
    axes = [ax1,ax2]
    fig.set_tight_layout({"pad": .0})


    #fig, axes = plt.subplots(1,2,figsize=(15,7));
    #EISR
    pal = ["#2f00ff","#1da135", "#ff0000", "#6e6e6e"]
    #red = "#e41a1c"; blue = "#377eb8"; green = "#4daf4a"
    #pal = [blue,red,green,"#34495e"]


    i = 0
    colors = [pal[0] if x==0.0 else pal[1] if x==1.0 else pal[2] if x==2.0 else pal[3] for x in
People['state'][:,i]]
    size = [15 if x==0.0 else 150 if x==1.0 else 200 if x==2.0 else 15 for x in People['state'][:,i]]
    scat = axes[0].scatter(People['position'][:,i,0], People['position'][:,i,1],c=colors,s=size,alpha =
1);
    axes[0].set_ylim(0,1); axes[0].set_xlim(0,1); axes[0].grid(False);
    #DayText = fig.get_axes()[0].text(x = 0.45,y=1.2,s= f"Day {np.round(i*dt,3)}");
    a = df[df['TimeStep'] <= i];

axes[1].stackplot(a["Days"],a["Infected"],a["Exposed"],a["Removed"],a["Susceptible"],labels=["Infected","E
xposed","Removed","Susceptible"],colors = [pal[2],pal[1],pal[3],pal[0]], alpha=0.5);
    axes[1].set_ylim(0,N);
    suptitl = fig.suptitle(TitlePhrase,fontsize=20)
    TextBox = TextArea.text(x=0.1,y=0.45,s="Farwa",fontsize=14)




    for market in range(N_markets):
        position_market = CentralMarkets['bottom-left'][market]
        size_market = CentralMarkets['top-right'][market] - position_market
        rect =
patches.Rectangle((position_market[0],position_market[1]),size_market[0],size_market[1],linewidth=1,edgeco
lor='r',facecolor='none')
        axes[0].add_patch(rect)


    #rect = patches.Rectangle((0,0),1,1,linewidth=2,edgecolor='k',facecolor='none')
    #axes[0].add_patch(rect)


    def animate(j):
        i = int(j)
        colors = [pal[0] if x==0.0 else pal[1] if x==1.0 else pal[2] if x==2.0 else pal[3] for x in
People['state'][:,i]]
        size = [15 if x==0.0 else 15 if x==1.0 else 15 if x==2.0 else 15 for x in People['state'][:,i]]
        scat.set_offsets(People['position'][:,i,:]);
        scat.set_color(colors);
        scat.set_sizes(size);


        axes[1].clear()
        a = df[df['TimeStep'] <= i];

axes[1].stackplot(a["Days"],a["Infected"],a["Exposed"],a["Removed"],a["Susceptible"],labels=["Infected","E
xposed","Removed","Susceptible"],colors = [pal[2],pal[1],pal[3],pal[0]], alpha=0.5);
        axes[1].set_ylim(0,N);
        axes[1].set_xlim(0,days);
        axes[1].legend(loc="upper left");
        axes[1].set_xlabel("Days");


        suptitl.set_text(TitlePhrase)
        DayDisp = f"Day {np.round(i*dt)}";


        InfectedDisp = f"Infected: {int(a['Infected'].values[-1])}";
        SusceptibleDisp = f"Susceptible: {int(a['Susceptible'].values[-1])}";
        RemovedDisp = f"Removed: {int(a['Removed'].values[-1])}";
        ExposedDisp = f"Exposed: {int(a['Exposed'].values[-1])}";
        CurrentStats = f' \n'.join([InfectedDisp,SusceptibleDisp,RemovedDisp,ExposedDisp]);


        CumulativeDisp = f"Cumulative Cases: {int(a['Infected'].values[-1] +
np.round(a['Removed'].values[-1]))}"


        TotalCasesDisp = f"Total Cases: {int(df['Infected'].values[-1] + np.round(df['Removed'].values[-
1]))}"
        PeakCasesDisp = f"Peak Cases: {int((df['Infected']).max())}"
        TextList = [DayDisp,CurrentStats,CumulativeDisp,TotalCasesDisp,PeakCasesDisp]

        if GoingHomeOn:
            PeopleAtHome = f"At Home: {int(People['home'][:,i].sum())}"
            TextList.append(PeopleAtHome);

        if quarantine:
            QuarantineDisp = f"Quarantined: {int(QuarantineCounts[i])}"
            TextList.append(QuarantineDisp);

        TextToDisplay = f" \n \n".join(TextList)
        TextBox.set_text(TextToDisplay)
        update_progress(i / Nt)
        return scat,TextBox

    # create animation using the animate() function
    myAnimation = animation.FuncAnimation(fig, animate, frames=np.arange(0.0, Nt, 1), \
                                          interval=40, blit=True, repeat=True)

    myAnimation.save("Covid - "+RunName+".mp4", writer= 'ffmpeg' , fps=30)

    TotalCasesDisp = int(df['Infected'].values[-1] + df['Removed'].values[-1])
    PeakCasesDisp = int((df['Infected']).max());
    PeakDay = df.iloc[df['Infected'].argmax()]['Days'];
    Enough = ((df['Infected'] + df['Exposed']) < 1).any()
    if Enough:
        DiseaseDeath = df[(df['Infected'] + df['Exposed']) < 1]['Days'].min()
    else:
        DiseaseDeath = days + 1

    Aggregates =
{'TotalCases':TotalCasesDisp,'PeakCases':PeakCasesDisp,'PeakDay':PeakDay,'DiseaseEnd':DiseaseDeath}
    #print(df[df['TimeStep'] == df['TimeStep'].last()]['Removed'])
    Entry = {'RunName': RunName,'Parameters': Parameters, 'TimeSeries': df,'Aggregates':Aggregates}

    return People,QuarantineCounts, Entry


### We added a little progress bar to see how far along the code has come (this snippet is adapted from
the stack exchange article https://stackoverflow.com/questions/3160699/python-progress-bar)
```

```
import time, sys
from IPython.display import clear_output
def update_progress(progress):
    bar_length = 20
    if isinstance(progress, int):
        progress = float(progress)
    if not isinstance(progress, float):
        progress = 0
    if progress < 0:
        progress = 0
    if progress >= 1:
        progress = 1

    block = int(round(bar_length * progress))

    clear_output(wait = True)
    text = "Progress: [{0}] {1:.1f}%".format( "#" * block + "-" * (bar_length - block), progress * 100)
    print(text)

### Finally Run the simulation
People,QuarantineCounts,Entry = RunSimulation(Parameters)
```

Batch Processing

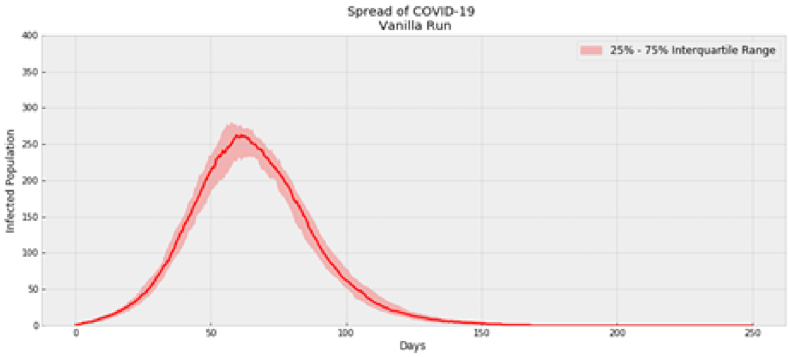
For batch processing, we removed the part of the above script that makes the animations (since it is resource intensive and contributes significantly to processing time), and added the following script to run the simulation 50 times. The results are saved and then processed.

```
AllResults = []
import time
s = time.time()
for i in range(50):
    print(i)
    Entry = RunSimulation(Parameters);
    AllResults.append(Entry);
totaltime = time.time()

print("Run Took ", totaltime - s, " seconds")

import pickle
pickle.dump( AllResults, open( "Aggregate Results " + RunName +".p", "wb" ) )
```

The saved pickles are then imported into python for making the graphs and aggregated results:

```
ResultFileName = "<Insert file name here>.p"
Results = pickle.load( open( ResultFileName, "rb" ) )


InfectedAggs = np.array([Results[i]['TimeSeries']['Infected'].values for i in range(50)]);
t_conf_95 = 2.01


plt.style.use('bmh')


fig = plt.figure(figsize=(15, 6))
grid = plt.GridSpec(1, 12, hspace=1, wspace=1)
ax1 = fig.add_subplot(grid[:,0:12])
ax1.set_xlabel('Days')
ax1.set_ylabel('Infected Population')


Nt = int(Results[0]['Parameters']['days']/Results[0]['Parameters']['dt'])


Upper = np.quantile(InfectedAggs, q=0.75,axis=0)
Lower = np.quantile(InfectedAggs, q=0.25,axis=0)




Days = np.arange(0,Results[0]['Parameters']['days'],Results[0]['Parameters']['dt'])
ax1.plot(Days,np.quantile(InfectedAggs, q=0.5,axis=0),color='r')
ax1.set_title(Results[0]['Parameters']['TitlePhrase'])
ax1.fill_between(Days, Lower, Upper,alpha=0.25,color='r',label=r"25% - 75% Interquartile Range")
ax1.set_ylim(0,400)


ax1.legend(prop={'size': 12})


fig.savefig("Aggregate Results " + Results[0]['Parameters']['RunName'] +".png")


plt.show()
```

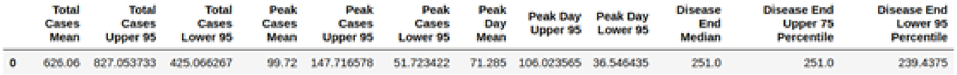
An example of a result:



Now we find the aggregate results and save them to a csv:

```
#Average Numbers


df = pd.DataFrame([Results[i]['Aggregates'] for i in range(50)])
t_conf_95 = 2.01


TotalCases = []; PeakCases = []; PeakDay = []; DiseaseEnd = [];
for i in range(50):
    TotalCases.append(Results[i]['Aggregates']['TotalCases']);
    PeakCases.append(Results[i]['Aggregates']['PeakCases']);
    PeakDay.append(Results[i]['Aggregates']['PeakDay']);
    #DiseaseEnd.append(Results[i]['Aggregates']['DiseaseEnd']);
    df = Results[i]['TimeSeries'];
    Enough = ((df['Infected'] + df['Exposed']) < 1).any()
    if Enough:
        DiseaseEnd.append(df[(df['Infected'] + df['Exposed']) < 1]['Days'].min())
    else:
        DiseaseEnd.append(Results[i]['Parameters']['days'] + 1)


TotalCases = np.array(TotalCases);
PeakCases = np.array(PeakCases);
PeakDay = np.array(PeakDay);
DiseaseEnd = np.array(DiseaseEnd);


returning_dict = {'Total Cases Mean': TotalCases.mean(),
'Total Cases Upper 95': TotalCases.mean() + t_conf_95*TotalCases.std(),
'Total Cases Lower 95': TotalCases.mean() - t_conf_95*TotalCases.std(),
'Peak Cases Mean': PeakCases.mean(),
'Peak Cases Upper 95': PeakCases.mean() + t_conf_95*PeakCases.std(),
'Peak Cases Lower 95': PeakCases.mean() - t_conf_95*PeakCases.std(),
'Peak Day Mean': PeakDay.mean(),
'Peak Day Upper 95': PeakDay.mean() + t_conf_95*PeakDay.std(),
'Peak Day Lower 95': PeakDay.mean() - t_conf_95*PeakDay.std(),
'Disease End Median': np.percentile(DiseaseEnd,50),
'Disease End Upper 75 Percentile': np.percentile(DiseaseEnd,75),
'Disease End Lower 95 Percentile': np.percentile(DiseaseEnd,25)}


ReturningAggregates = pd.DataFrame([returning_dict]);
ReturningAggregates.to_csv("Aggregate Results " + Results[0]['Parameters']['RunName'] +".csv")
ReturningAggregates
```

An example result is shown below, and a csv file is also saved:



	Total Cases Mean	Total Cases Upper 95	Total Cases Lower 95	Peak Cases Mean	Peak Cases Upper 95	Peak Cases Lower 95	Peak Day Mean	Peak Day Upper 95	Peak Day Lower 95	Disease End Median	Disease End Upper 75 Percentile	Disease End Lower 95 Percentile
0	626.06	827.053733	425.066267	99.72	147.716578	51.723422	71.285	106.023565	36.546435	251.0	251.0	239.4375