Epidemiology – Human and Water Borne Illnesses

Marc Los Huertos

March 20, 2020

COVID-19 began rapidly spreading around the world in late 2019. And by the spring, Italy went into lock down, California declared a state of emergency, schools and universities around the globe suspended in person classes and events, and businesses reduced travel and pushed tele-work policies. All of this was designed to slow the spread of the disease. These efforts are broadly referred to as social distancing.

0.1 Value of Social Distancing

The idea is to reduce person-to-person contact to make spreading the disease less likely. The effects of this are often illustrated in images such as those in the chart below, where the red plot is flattened to spread out the disease as much as possible. This helps to ensure that there are sufficient resources available for a sick population, which will help improve survival rates.

Flattening the curve to keep infection manageable (Source: Fast.ai). How do we determine the value of such distancing strategies and model this spread?

0.2 Modeling Disease – SEIR

We walk through a SEIR epidemiological model and simulate it with R. The first model is the basic SEIR without social distancing, then we add social distancing to show how the potential effectiveness of these strategies.

The SEIR model is a compartmental model for modeling how a disease spreads through a population. It's an acronym for Susceptible, Exposed, Infected, Recovered. When a disease is introduced to a population, the people move from one of these classes (or compartments) to the next. When they reach the R state, they're no longer able to be infected, depending on your interpretation, they either survived the disease and are now immune or succumbed to the illness and are out of the population.

This is an extension of the classic SIR model and simply adds one more equation to show those who are exposed.

The Kermack-McKendrick model is an SIR model for the number of people infected with a contagious illness in a closed population over time. It was

proposed to explain the rapid rise and fall in the number of infected patients observed in epidemics such as the plague (London 1665-1666, Bombay 1906) and cholera (London 1865). It assumes that the population size is fixed (i.e., no births, deaths due to disease, or deaths by natural causes), incubation period of the infectious agent is instantaneous, and duration of infectivity is same as length of the disease. It also assumes a completely homogeneous population with no age, spatial, or social structure.

The model consists of a system of three coupled nonlinear ordinary differential equations,

$$(dS)/(dt) = -\beta SI \tag{1}$$

$$(dI)/(dt) = \beta SI - \gamma I \tag{2}$$

$$(dR)/(dt) = \gamma I, \tag{3}$$

where t is time, S(t) is the number of susceptible people, I(t) is the number of people infected, R(t) is the number of people who have recovered and developed immunity to the infection, beta is the infection rate, and gamma is the recovery rate.

The key value governing the time evolution of these equations is the so-called epidemiological threshold,

$$R_0 = (\beta S)/\gamma \tag{4}$$

Note that the choice of the notation R_0 is a bit unfortunate, since it has nothing to do with R. R_0 is defined as the number of secondary infections caused by a single primary infection; in other words, it determines the number of people infected by contact with a single infected person before his death or recovery.The full model is given below:

We have four ODE's in the time domain, with three parameters: α, β, γ.

- α is the inverse of the incubation period $(1/t_{incubation})$
- β is the average contact rate in the population
- γ is the inverse of the mean infectious period $(1/t_{infectious})$

Equation (1) is the change in people susceptible to the disease and is moderated by the number of infected people and their contact with the infected. Equation (2) gives the people who have been exposed to the disease. It grows based on the contact rate and decreases based on the incubation period whereby people then become infected.

1 Models with Susceptibility and Infection

1.1 Model Structure

The model relies on spatially explicit movements of 'subjects' that basically bounce around in a box, i.e. petri dish. Each subject is assigned a random location and trajectory (angle) of movement and velocity of movement vector. I based the model on an Washington Times article by XXXX (published). Although his model, written in Java, is probably much cleaner, I wanted to create something that students could use to adjust parameters and evaluate different policy strategies, using XX's visualization as a foundation.

To track subject movements, contact with others, etc I have defined an array to track subject, thus, have to do some backflips to track overlap at each time step to see if people are sharing space and pathogens.

I'm not a programmer, so this is going to need some TLC as it's pretty ineffecient.

1.2 Model Parameterization

I have define the following parameters for the model, which will be further described in each section below.

	Model	N	Velocity	tstep	Stationary	Inf_Dist	init_infact	asymp	symp	rec
1	Model1	50.00	5.00	2000.00	0.00	25.00	2.00	24.00	96.00	720.00
2	Model2	100.00	5.00	2000.00	0.00	25.00	2.00	24.00	96.00	720.00
3	Model3	100.00	5.00	3000.00	0.40	25.00	2.00	24.00	96.00	720.00
4	Model4	100.00	5.00	3000.00	0.40	25.00	2.00	60.00	60.00	720.00
5	TBD									720.00
6	Test	5.00	5.00	20.00	0.00	500.00	1.00	24.00	96.00	720.00

But for now, I'd like to see how these align with the equations described above.

1.3 Model Functions

First, functions are useful when you have repetitive actions to make – in this case, I have defined functions to move sujects based on their speed and direction of travel.

Since these functions rely on the data array within another function (thus, not globally available, until it it 'returned' at the end fo the function), I had to include a refernece to the data in the function so that the functions could find the array.

```
#create function to move the subject
move_x = function(data_array, s, t){
data_array[s, 4, t] * cos(data_array[s,3, t]*pi/180)}
```

```
move_y = function(data_array, s, t){
data_array[s, 4, t] * sin(data_array[s,3, t]*pi/180)}

# check function
# move_y(data_arr, 1, 1); move_x(data_arr, 1,1)

# move function
move = function(data_array, s, t){
  c(data_array[s, 1, t-1]+ move_x(data_array, s, t-1),
    data_array[s, 2, t-1]+ move_y(data_array, s, t-1),
    data_array[s, 3, t-1],
    data_array[s, 4, t-1],
    data_array[s, 5, t-1],
    data_array[s, 6, t-1])
}

# Test functions
# data_arr[1,,1]
# move(data_arr, 1, 2)
# Model1; i = 2
```

```
# Contact Infection Function
contact = function(data_array, t, Infect_dist){

# testfunction
  #data_array=Model1_50; t=88; infect_dist = 5
  pairs = cbind(as.vector(data_array[,"x",t]),
          as.vector(data_array[,"y",t])); pairs
  tmp = as.matrix(dist(pairs, method = "euclidean" )); tmp
  tmp[upper.tri(tmp, diag = TRUE)] <- NA; tmp
  pairs <- which(tmp < Infect_dist ,arr.ind = TRUE);pairs
  tmp2 = cbind(as.numeric(rownames(tmp)[pairs[, 1]]),
               as.numeric(colnames(tmp)[pairs[, 2]])); tmp2
  contacts = dim(tmp2)[1]; contacts
  if(contacts>0){
    for(i in 1:contacts){
      # If one is exposed, susceptable is exposed
      if(data_array[tmp2[i,1],5,t]==2 & data_array[tmp2[i,2],5,t]==1){
      data_array[tmp2[i,2],5,t] = 2
      }
      # If one is exposed, susceptable is exposed
      if(data_array[tmp2[i,2],5,t]==2 & data_array[tmp2[i,1],5,t]==1){
      data_array[tmp2[i,1],5,t] = 2
      }
```

```
      # If one is symptomatic, the susceptable is exposed
      if(data_array[tmp2[i,1],5,t]==3 & data_array[tmp2[i,2],5,t]==1){
      data_array[tmp2[i,2], 5, t] = 2
      }
      # If one is symptomatic, the susceptable is exposed
      if(data_array[tmp2[i,1],5,t]==1 & data_array[tmp2[i,2],5,t]==3){
      data_array[tmp2[i,1], 5, t] = 2
      }
    }

  }
   # data_array[,,t]
return(data_array)
# print(data_array[,5,t])
} # close function

#contact(Model1, 87, 5)
```

Next, I'm going to figure out how to create a function to text for euclidian distances to spread disease, but this will take some more effort ...

1.4 Model 1: Modeling Susceptability and Infections with No Behavior Changes and Recovery

For this model, people get instantenously sick and start spreading it around – not very realistic because when people get sick they are usually quarantined and stop bouncing around the environment. Nevertheless this sets up a good base line, where the poputation gets ill pretty quickly.

```
MoveSubjects = function(m=0){
# m = 0
N = params[m, 2]
Velocity = params[m, 3]
tstep = params[m, 4]
Stationary = params[m,5];
Infect_dist=params[m, 6];
Init_infect=params[m, 7]
asymp = params[m,8]
symp = params[m,9]
rec = params[m,10]

# Define Data Array
location = 6 # number of parameters to track
data_arr = array(dim=c(N, location, tstep))
dimnames(data_arr)[[2]] <- c("x", "y", "theta", "velocity", "status", "count")
```

```
data_arr[,,1]
# Initialize Start Locations
set.seed(2763)
subj_x = runif(N, coord_range[1],  coord_range[2]); subj_x
subj_y = runif(N, coord_range[3],  coord_range[4]); subj_y

# Set Up Population Characteristics and Behaviors
sheltered = sample(N*Stationary); sheltered
velocity = rep(Velocity,N); velocity[sheltered]=0; velocity
theta = round(runif(N, 0, 360),0); theta
SEIR = rep(1, N); SEIR[sheltered] = 0
SEIR[sample(N, Init_infect)] = 3; SEIR
count = rep(0, N)

# Initial Locations
data_arr[,,1] = c(subj_x, subj_y, theta, velocity, SEIR, count)
data_arr[,,1]
# t = 2; s=1
for(t in 2:tstep){
  for(s in 1:N){
    # move subjects based on theta and speed
    data_arr[s,,t] = move(data_arr, s, t)
    # coarse corrections when hitting a boundary
    # Min x-boundary
    if(data_arr[s,1,t] < coord_range[1]){
      data_arr[s,3,t-1]=180-data_arr[s,3,t-1]
      data_arr[s,,t] = move(data_arr, s, t)
    }
    #Max x-boundary
    if(data_arr[s,1,t] > coord_range[2]){
      data_arr[s,3,t-1]=180-data_arr[s,3,t-1]
      data_arr[s,,t] = move(data_arr, s, t)
    }
    #Min y-boundary
    if(data_arr[s,2,t] < coord_range[3]){
      data_arr[s,3,t-1]=360-data_arr[s,3,t-1]
      data_arr[s,,t] = move(data_arr, s, t)
    }
    #Max y-boundary
    if(data_arr[s,2,t] > coord_range[4]){
      data_arr[s,3,t-1]=360-data_arr[s,3,t-1]
      data_arr[s,,t] = move(data_arr, s, t)
    }
    # Change Status based on Time Steps
    data_arr[s,6,t] = data_arr[s,6,t] + 1
```

```
    if(data_arr[s,5,t]==1 | data_arr[s,5,t]==0) data_arr[s,6,t]=0 # Reset Susceptables
   # Exposed becomes symptomatic
  if(data_arr[s,5,t]==2 & data_arr[s,6,t] > asymp){
    data_arr[s,5,t]=3}
    # Symptomatics move very little
  if(data_arr[s,5,t]==3 & data_arr[s,6,t] > symp){
    data_arr[s,4,t] = Velocity/2}
  # Time to Recovery
  if(data_arr[s,5,t]==3 & data_arr[s,6,t] > rec){
    data_arr[s,5,t] = 4
    data_arr[s,4,t] = Velocity
    }
  } # End of Subj Moving

  # Time Step Proceesing
  # Test for Euclidean Distances Here!

data_arr <- contact(data_arr, t, Infect_dist)

} # End of Time Step
return(data_arr)
}
```

1.5 Model Runs

For after running the models, I visually evaluated the models to make sure there weren't unintended consequences, i.e. usually from coding errors. Below are the models that I ran and the estimated run times.

```
library(tictoc)
#N=3, tstep=4, speed=5, init_infect=1, stationary = 0)
params

tic("Model0")
Test<- MoveSubjects(m=6)
toc() # .02s

tic("Model1")
Model1<- MoveSubjects(m=1)
toc() # 900s; 314s

tic("Model2")
Model2<- MoveSubjects(m=2)
toc() #1800s; 1158s; 1313s
```

```
tic("Model3")
Model3 <- MoveSubjects(m=3)
toc() #3308; 2437s; 2599s

tic("Model4")
Model4 <- MoveSubjects(m=4)
toc() #3308s; 2332s; 2616s
```

1.6 Tracking the Disease Transfer

```
#load("/home/CAMPUS/mwl04747/github/beginnersluck.RData")
# Susceptable 1, Exposed-Asymptomatic 2, Symptomatic 3, Recovered 4
SEIRcol = c("steelblue", "orange","red", "green")

trackresults=function(model,m,tstep){
results = data.frame(Days = NA, Sh = NA, S=NA, E=NA, I=NA, R=NA)
for(t in 1:tstep){
days = t/24
sh = sum(as.vector(model[,5,t])==0)
s = sum(as.vector(model[,5,t])==1, sh)
e = sum(as.vector(model[,5,t])==2)
i = sum(as.vector(model[,5,t])==3)
r = sum(as.vector(model[,5,t])==4)
results[t,] <- data.frame(Days = days, Sh = sh, S=s, E=e, I=i, R=r)
}
head(results)
par(mfrow=c(1,1), mar=c(4, 5, 3, 2)+.1, las=1)
plot(S~Days, data=results, ty='l', col=SEIRcol[1], ylab="", ylim=c(0,params[m,2]), xlab="Day
lines(E~Days, data=results, col=SEIRcol[2], lwd=2)
lines(I~Days, data=results, col=SEIRcol[3], lwd=2)
lines(R~Days, data=results, col=SEIRcol[4], lwd=2)
}

m = 0; trackresults(Test, m, params[m,4])

png(filename =
  "/home/CAMPUS/mwl04747/github/beginnersluck/Epidemiology/Model1.png");
  m = 1; trackresults(Model1, m, 1440); dev.off()
png(filename =
  "/home/CAMPUS/mwl04747/github/beginnersluck/Epidemiology/Model2.png");
  m = 2; trackresults(Model2, m, 1440); dev.off()
png(filename =
  "/home/CAMPUS/mwl04747/github/beginnersluck/Epidemiology/Model3.png");
```

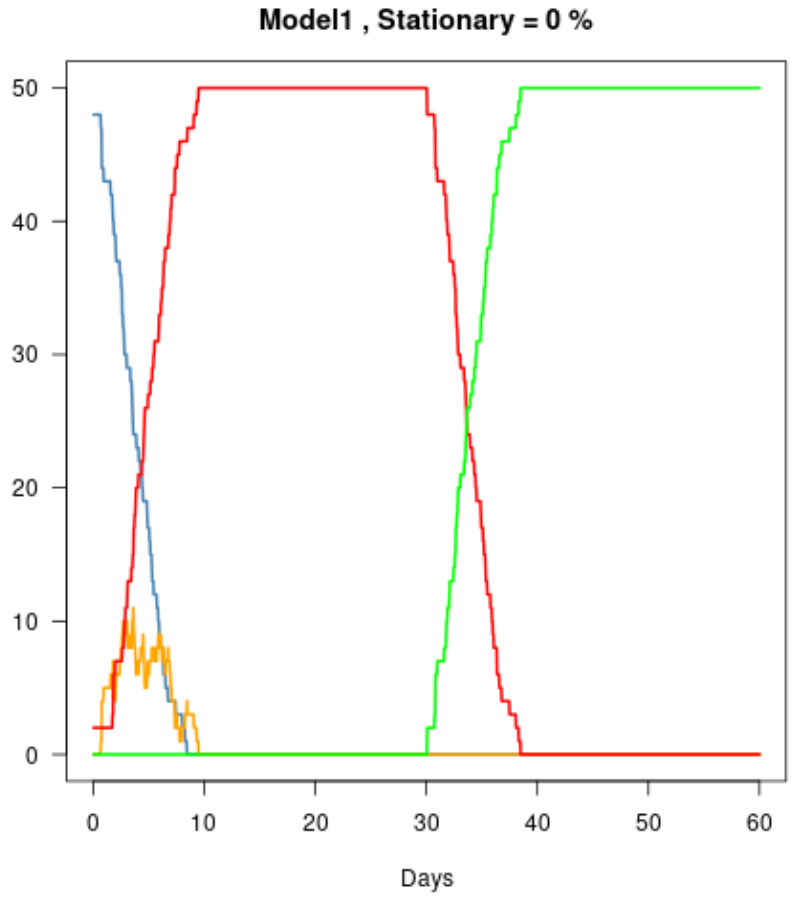

Figure 1: N = 50

```
  m = 3; trackresults(Model3, m, 1440); dev.off()
png(filename =
  "/home/CAMPUS/mwl04747/github/beginnersluck/Epidemiology/Model4.png");
  m = 4; trackresults(Model4, m, 1440); dev.off()
```

2 Plot Results

2.1 Static plot

Because we are looking at a dynamic system, I don't find the static plot all that useful.

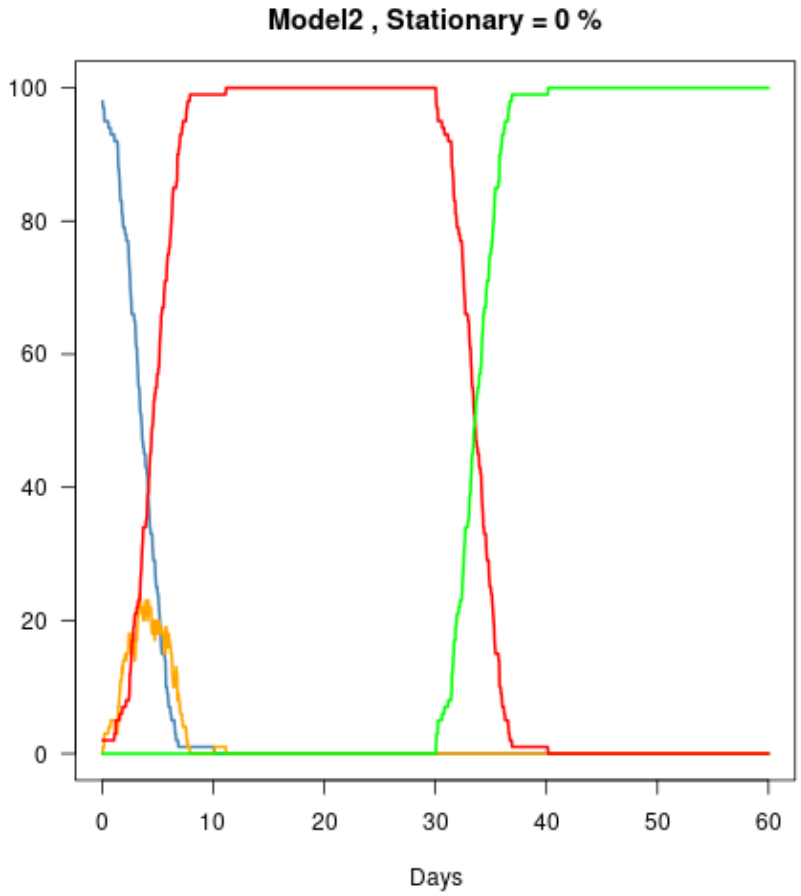

Figure 2: N=100

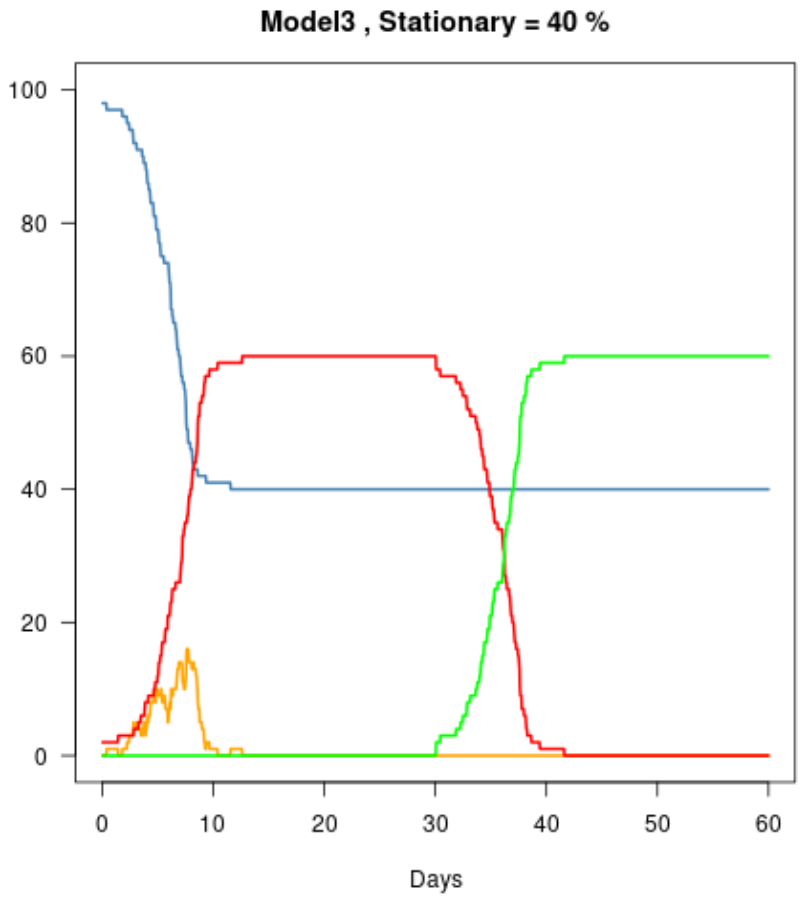

Figure 3: 40% shelter in place

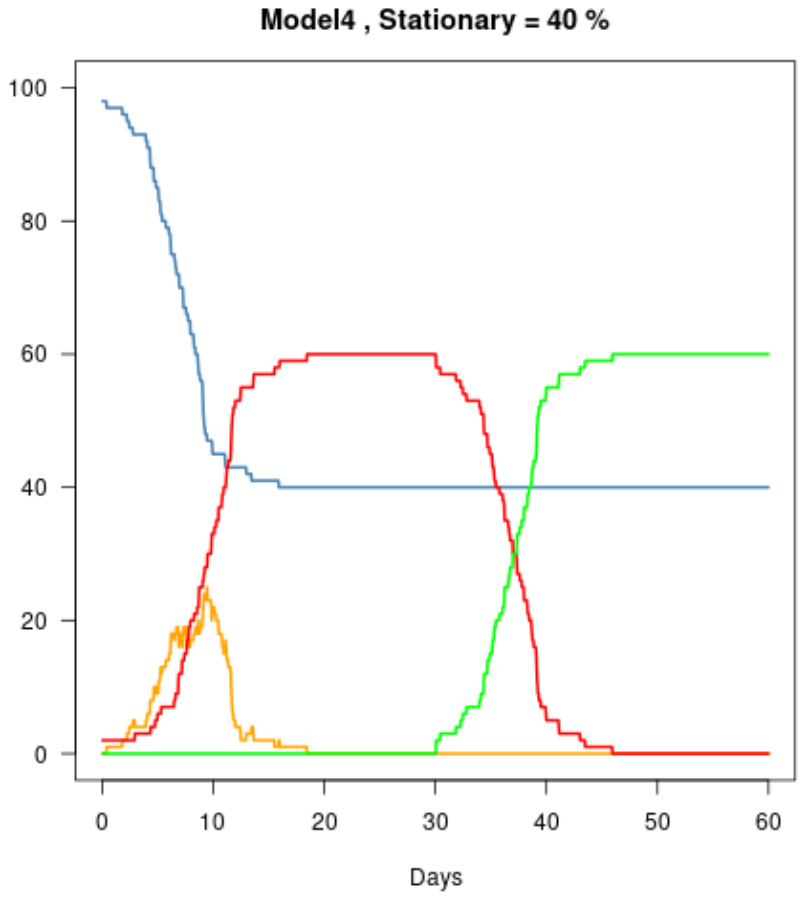

Figure 4: 40% shelter in place, exposed without symptoms.

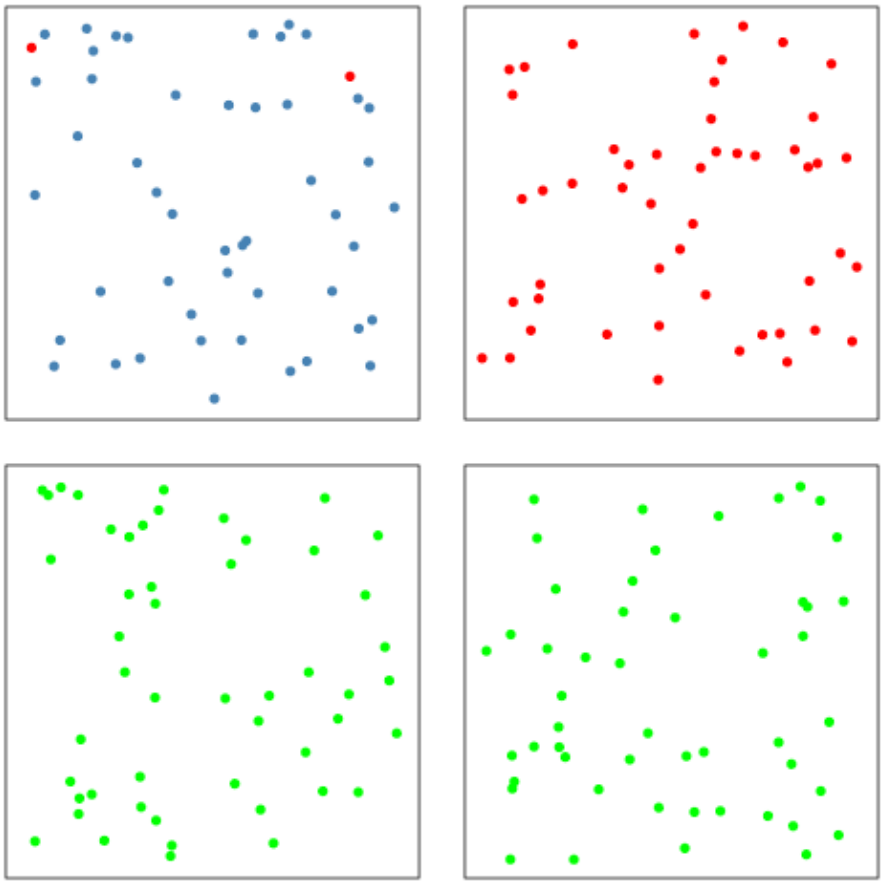

Figure 5: N = 50

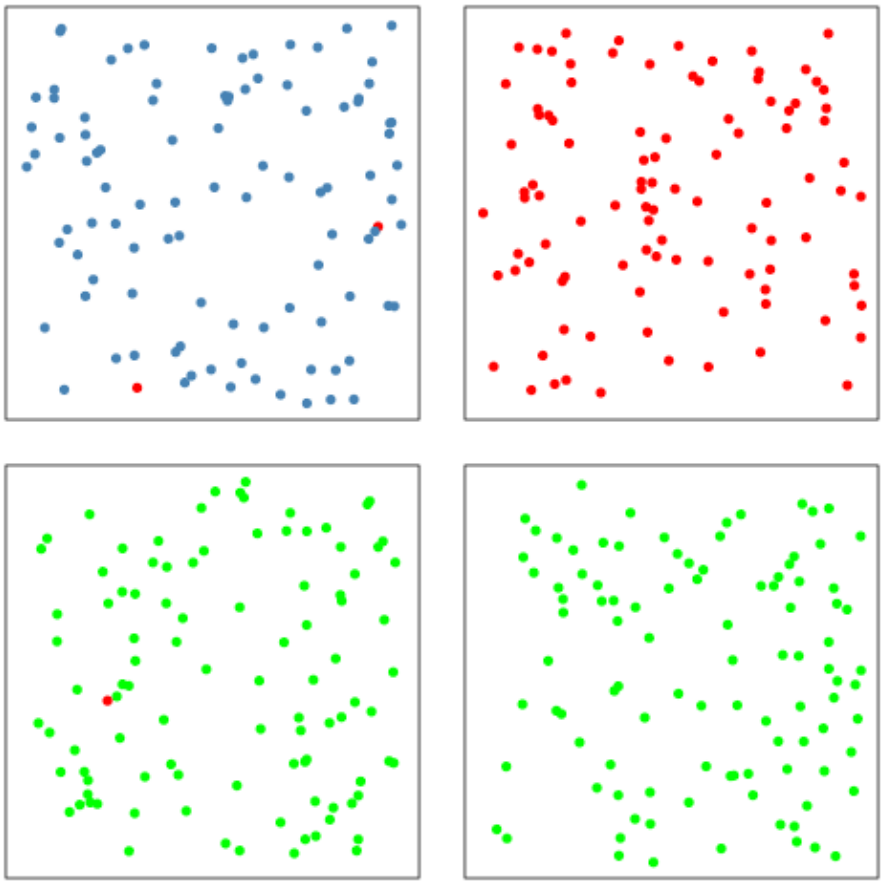

Figure 6: N=100

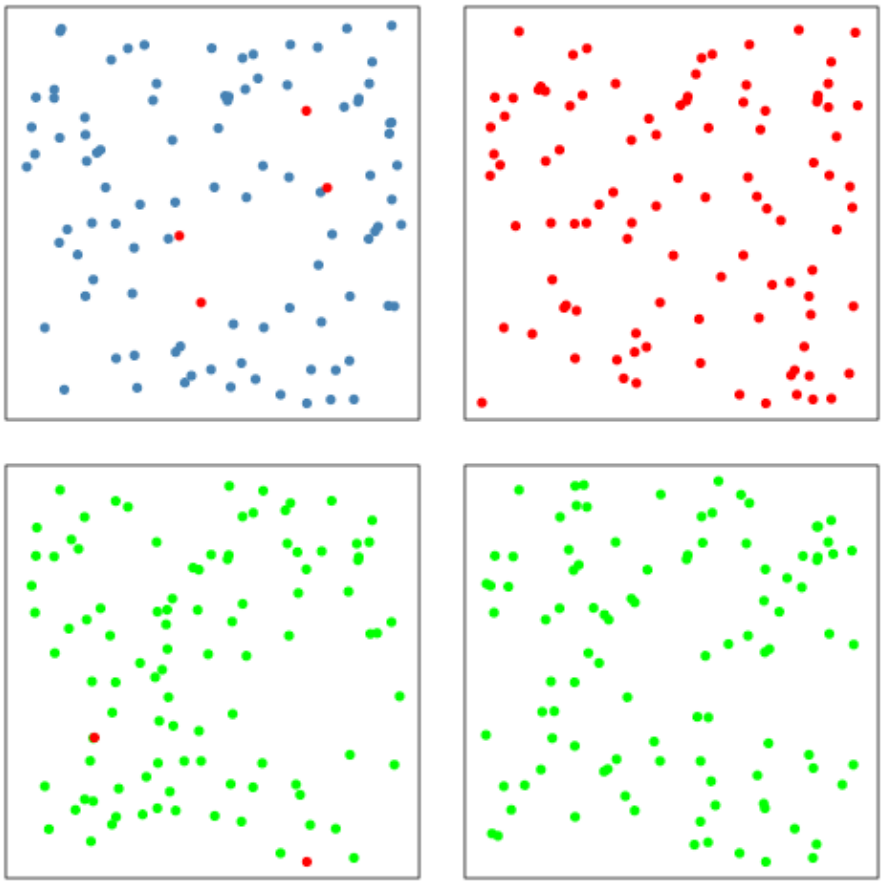

Figure 7: 40% shelter in place

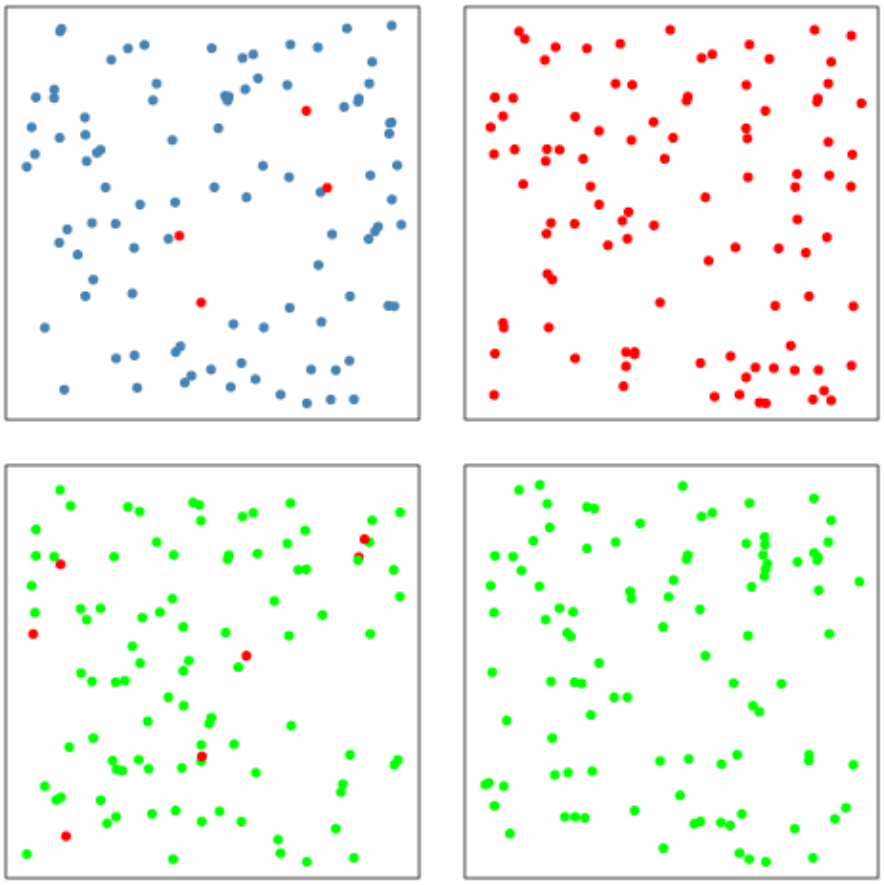

Figure 8: 40% shelter in place, exposed without symptoms.

2.2 Animating the Results

Still working on this... I have to export the file to my laptop then use photoshop to make a movie.

it would be nice to do this in r and then embedd in a pdf...