DELHI TECHNOLOGICAL UNIVERSITY

MATHEMATICAL MODELLING AND SIMULATION
(MC 409)

Midterm Innovative Project

SUBMITTED BY:
Aiman Siddiqua - 2K18/MC/008
Apoorva - 2K18/MC/019

SUBMITTED TO:
Ankit Sharma

WORLD WAR Z:

Simulating a Zombie Outbreak using Monte Carlo Method

Modelling and simulating the spread of a Zombie Virus in a specific geographical location over time using Monte Carlo methods, a modified SIR model and Random Walk.

INDEX

INTRODUCTION

A zombie, according to pop culture and folklore, is usually either a reawakened corpse with a ravenous appetite or someone bitten by another zombie infected with a "zombie virus."

Zombies appear as villains in all forms of fictional media, but they can also be used as a metaphor. In essence, Zombie-ism is basically an epidemic of a life-altering virus. Mapping a pandemic is extremely relevant in today's day and age, where the coronavirus has changed the lives of everyone in the world.

In this project, we attempted to model an outbreak of the zombie virus, with a population confined to a specific geographical location. The simulation allows zombies to transmit the virus via person-to-person contact, and their transmission mode was modelled using a Random Walk.

In the publication, W*hen zombies attack!: Mathematical modelling of an outbreak of zombie infection*, Munz et al defined a slightly more complicated SIR model called the SZR model in order to characterize the spread of a zombie contagion. This model has been implemented in our code.

THE MODEL

The spread of zombie infection over time was modelled in *"When zombies attack!: Mathematical modelling of an outbreak of zombie infection"*, *Munz et al*, 2009 as follows:

We consider the following base three classes:

1) Susceptible (S)
2) Zombie (Z)
3) Removed (R)

Susceptibles can become deceased through 'natural' causes, i.e. non-zombie related death (parameter δ). The **Removed** class consists of individuals who have died, either through attack or natural causes. Humans in the removed class can resurrect and become a zombie (parameter ζ). Susceptibles can become zombies through transmission via an encounter with a zombie (transmission parameter β).

New zombies can only come from two sources:

1) The resurrected from the newly deceased (removed group)
2) Susceptibles who have 'lost' an encounter with a zombie

The initial population p is a constant. The zombies move to the Removed class after being defeated (parameter α).

The basic model is given by:

$$S' = p - \beta SZ - \delta Z$$

$$Z' = \beta SZ + \zeta - \alpha SZ$$

$$R' = \delta + \alpha SZ = \zeta R$$

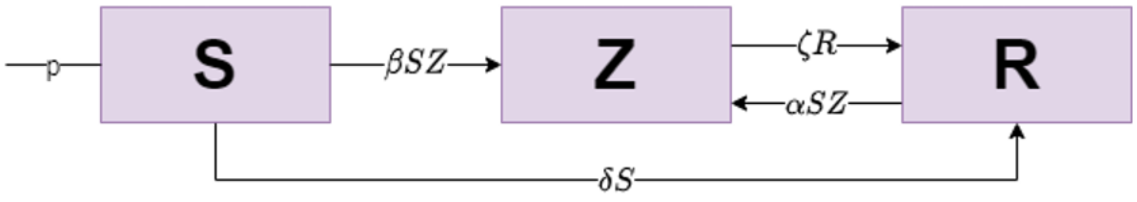


Fig. The basic SZR model for zombie outbreak

This model is slightly more complicated than the basic SIR models that usually characterise infectious diseases because this model has two mass action transmissions. Thus, there are more than one nonlinear term in the model.

Mass-action incidence specifies that an average member of the population makes contact sufficient to transmit infection with βN others per unit time, where N is the total population without infection.

In this case, the infection is zombification. The probability that a random contact by a zombie is made with a susceptible is S/N; thus, the number of new zombies through this transmission process in unit time per zombie is:

$$\beta N.(\frac{S}{N}).Z = \beta SZ$$

We assume that a susceptible can avoid zombification through an altercation with a zombie by defeating the zombie during their contact, and each susceptible is capable of resisting infection (becoming a zombie) at a rate α.

So using the same idea as above with the probability Z/N of random contact of a susceptible with a zombie (not the probability of a zombie attacking a susceptible), we have the number of zombies destroyed through this process per unit time per susceptible is:

$$\alpha N.(\frac{Z}{N}).S = \alpha SZ$$

If we assume that the outbreak happens over a short timescale, then we can ignore birth and background death rates. Thus, we set $p = \delta = 0$,

$$-\beta SZ = 0$$

$$\beta SZ + \zeta R - \alpha SZ = 0$$

$$\alpha SZ - \zeta R = 0$$

From the first equation, we have either $S = 0$ or $Z = 0$. Thus, it follows from $S = 0$ that we get the 'doomsday' equilibrium:

$$(S'', Z'', R'') = (0, Z'', 0)$$

When $Z = 0$, we have the disease-free equilibrium:

$$(S'', Z'', R'') = (N, 0, 0)$$

These equilibrium points show that, regardless of their stability, human-zombie coexistence is impossible.

STABILITY

The Jacobian for the ODE is then

$$J(S,Z,R) = \begin{bmatrix} -\beta Z & -\beta S & 0 \\ -\beta Z - \alpha Z & \beta S - \alpha S & \zeta \\ \alpha Z & \alpha S & -\zeta \end{bmatrix}$$

The Jacobian at the disease-free equilibrium is

$$J(N,0,0) = \begin{bmatrix} 0 & -\beta N & 0 \\ 0 & \beta N - \alpha N & \zeta \\ 0 & \alpha N & -\zeta \end{bmatrix}$$

Hence,

$$det(J - \lambda I) = -\lambda[\lambda^2 + (\zeta - (\beta - \alpha)N)\lambda - \beta\zeta N]$$

It follows that the characteristic equation always has a root with a positive real part. Hence, the disease-free equilibrium is always unstable.

Next, we have

$$J(0,Z'',0) = \begin{bmatrix} -\beta Z & 0 & 0 \\ -\beta Z - \alpha Z & 0 & \zeta \\ \alpha Z & 0 & -\zeta \end{bmatrix}$$

Thus,

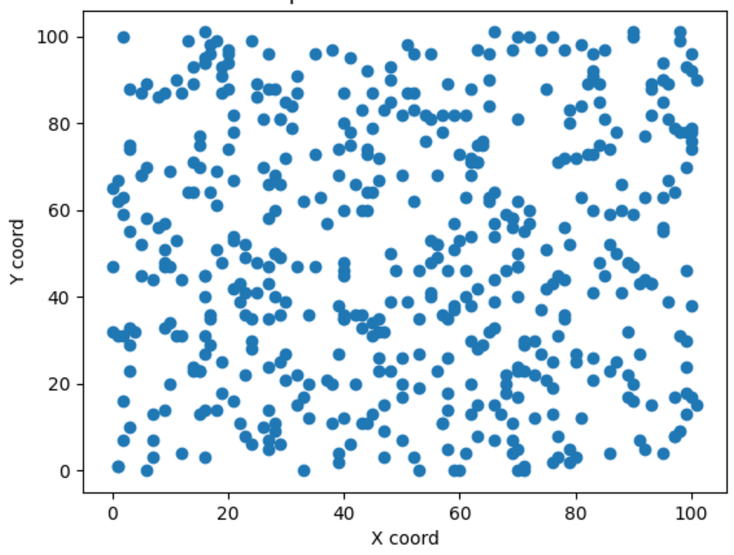
$$det(J - \lambda I) = -\lambda(\beta Z^{-} - \lambda)(-\zeta - \lambda)$$

Since all eigenvalues of the doomsday equilibrium are negative, it is asymptotically stable. It follows that, in a short outbreak, zombies will likely infect everyone.

SIMULATIONS

A small human population is confined to a "walled city", the humans are randomly assigned a starting position. These starting positions have been mapped as:

Figure 4
Plot of Population of humans at time = 0

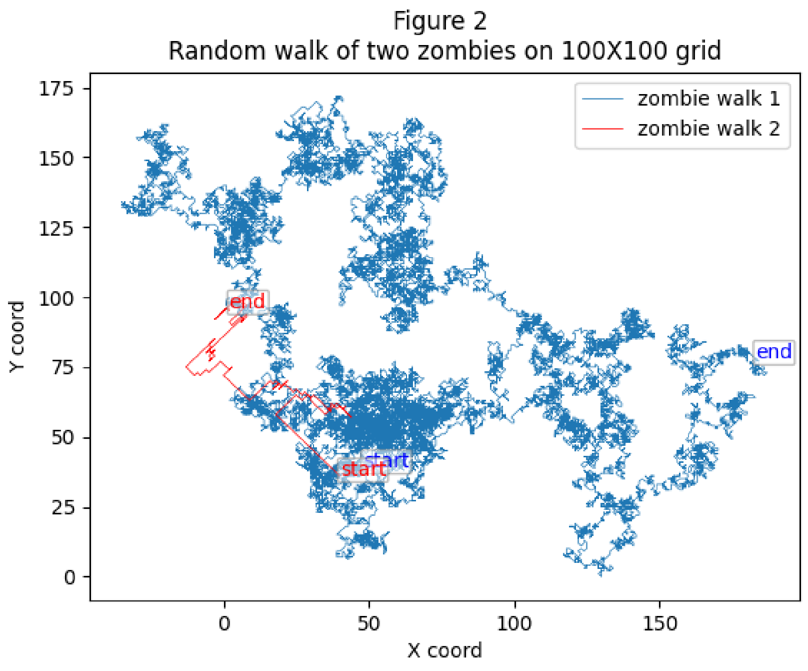


We have the following parameters for our simulation.

Starting Population(p): 500
Probability of Natural Death (δ): 0.001
Probability of Resurrection of a Zombie(ς): 0.03
Probability the Human Kills the Zombie(α): 0.4
Zombie Speed (Steps Taken in One Time Increment): 4
Time for which model is run: 5000
Initial Dead population: 50

Humans have a possibility (δ) of dying through natural causes. From the initial population (p), 1% of the population is initially dead. From this 1%, there is a probability(ς) that they may resurrect into Zombies.

Zombies can now do a random walk in this 2-D space. A sample of the random walk trajectory of a zombie in the simulation can be seen below. Zombie Walk 1 shows the path taken by all zombie number 482. Zombie Walk 2 is the path taken by zombie number 5.



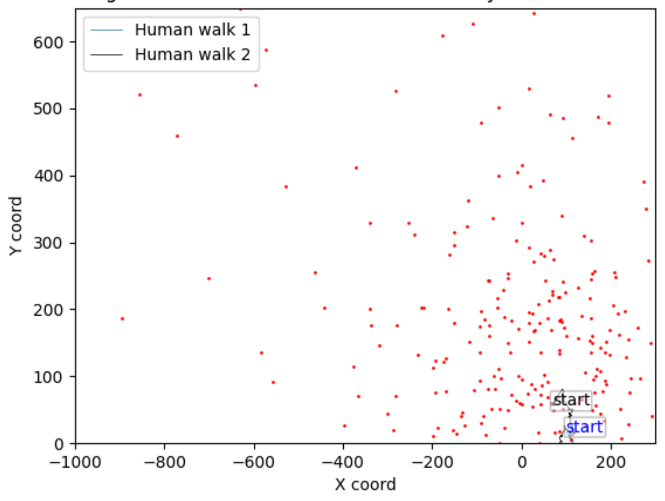
When a zombie encounters a human, there is a probability α) that the human "loses" this encounter and gets infected, thereby becoming a zombie. Else, the human wins and successfully kills the zombie.

These status changes are shown as:

```
Zombie 496 kills human 1!
Human 16 destroyed zombie 495!
Zombie 114 kills human 216!
Human 279 destroyed zombie 388!
Zombie 474 kills human 332!
Human 122 destroyed zombie 453!
Zombie 440 kills human 4!
Zombie 440 kills human 70!
Zombie 440 kills human 403!
Zombie 5 kills human 120!
Zombie 5 kills human 255!
Zombie 5 kills human 490!
Human 109 destroyed zombie 5!
Zombie 253 kills human 316!
Zombie 253 kills human 235!
Zombie 271 kills human 386!
Human 180 destroyed zombie 88!
Zombie 110 kills human 171!
Human 92 destroyed zombie 438!
Human 131 destroyed zombie 287!
Zombie 213 kills human 407!
Human 375 destroyed zombie 213!
Human 229 destroyed zombie 381!
Zombie 36 kills human 241!
Zombie 36 kills human 451!
Zombie 70 kills human 62!
Zombie 89 kills human 209!
Zombie 314 kills human 375!
Zombie 36 kills human 306!
Zombie 336 kills human 396!
Zombie 403 kills human 90!
Human 16 destroyed zombie 103!
Human 156 destroyed zombie 401!
Human 190 destroyed zombie 196!
Zombie 95 kills human 161!
Zombie 153 kills human 406!
Human 94 destroyed zombie 180!
Human 29 destroyed zombie 94!
Zombie 21 kills human 195!
Zombie 160 kills human 377!
Human 449 destroyed zombie 160!
Zombie 493 kills human 340!
Human 11 destroyed zombie 409!
Human 220 destroyed zombie 100!
Human 139 destroyed zombie 365!
Zombie 383 kills human 298!
Human 315 destroyed zombie 383!
Human 291 destroyed zombie 446!
Zombie 217 kills human 437!
Zombie 159 kills human 80!
Human 131 destroyed zombie 159!
Zombie 263 kills human 305!
Human 465 destroyed zombie 263!
Human 93 destroyed zombie 54!
Human 486 destroyed zombie 465!
Zombie 282 kills human 395!
Human 49 destroyed zombie 439!
Zombie 442 kills human 291!
```

In the model, we also added a factor called "human movement". A human or *susceptible* can tell which is the zombie nearest to him and run away from that direction. This trajectory has been visualized here:

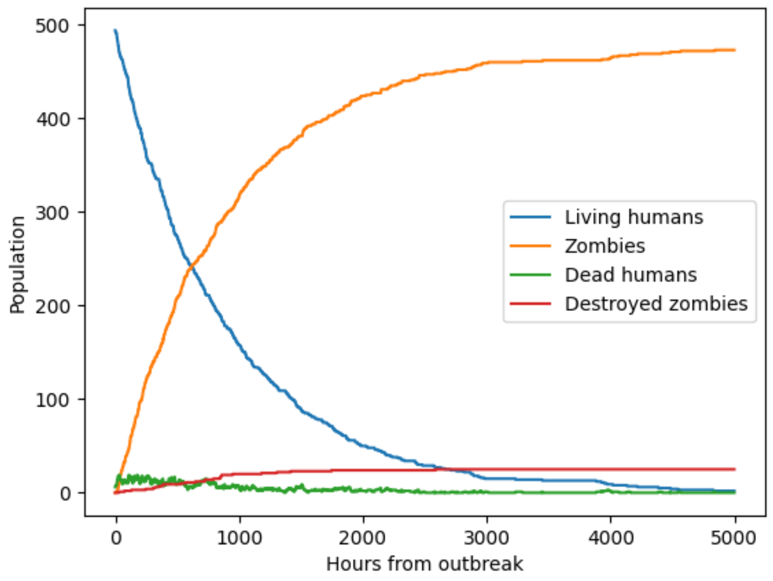
Figure 3 Deliberate Walk of Humans Away from Zombies



The end results for the simulation are:

```
End result: Zombies 473, Humans 2, Dead 0, Destroyed 25 in 5000 hours
```

Figure 1
Zombie infection simulation with random walk of zombies



RESULT

We ran several simulations from the code that implements this model.

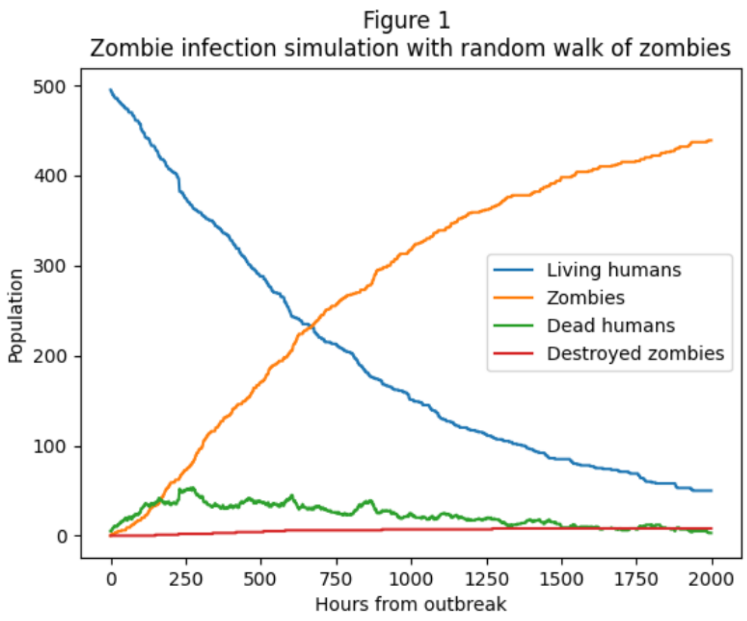
SIMULATION 1

```
p = 500 #starting population
d = 0.001 # probability of natural death
G = 0.01 #probability of resurrection of zombie
A = 0.2 #probability of human destroying zombie on encounter
mt = 2000 # Optional Length of time over which the model is run
zs = 5 #read zombie speed
di = int(p/100) # 1% of initial population is dead
```

```
Zombie 209 kills human 36!
Zombie 385 kills human 253!
Zombie 385 kills human 338!
Zombie 385 kills human 69!
Zombie 385 kills human 34!
Zombie 385 kills human 273!
Human 126 destroyed zombie 21!
Human 416 destroyed zombie 377!
Zombie 372 kills human 168!
Zombie 372 kills human 203!
Zombie 372 kills human 235!
Zombie 372 kills human 486!
Zombie 372 kills human 487!
Zombie 372 kills human 298!
Zombie 486 kills human 3!
Zombie 486 kills human 48!
Zombie 486 kills human 401!
Zombie 70 kills human 62!
Zombie 89 kills human 209!
Zombie 314 kills human 375!
Zombie 36 kills human 306!
Zombie 336 kills human 396!
Zombie 403 kills human 90!
Human 16 destroyed zombie 103!
Zombie 486 kills human 433!
Zombie 486 kills human 80!
Zombie 486 kills human 414!
```

```
Human 106 destroyed zombie 199!
Zombie 468 kills human 30!
Human 42 destroyed zombie 468!
Zombie 86 kills human 38!
Zombie 86 kills human 46!
Human 62 destroyed zombie 86!
Zombie 411 kills human 14!
Human 465 destroyed zombie 411!
Zombie 490 kills human 351!
Zombie 490 kills human 440!
Zombie 303 kills human 142!
Zombie 303 kills human 453!
Human 329 destroyed zombie 303!
Zombie 324 kills human 77!
Zombie 324 kills human 197!
Zombie 324 kills human 374!
Zombie 324 kills human 491!
Zombie 50 kills human 24!
Zombie 50 kills human 175!
Zombie 50 kills human 346!
Zombie 307 kills human 185!
Human 278 destroyed zombie 307!

End result: Zombies 439, Humans 50, Dead 3, Destroyed 8 in 2000 hours
```



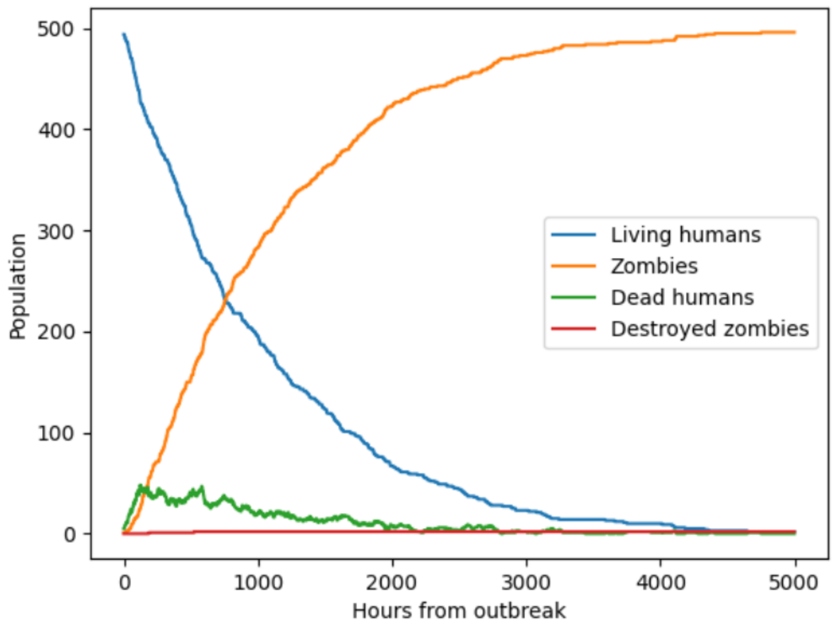
SIMULATION 2

```
p = 500 #starting population
d = 0.001 # probability of natural death
G = 0.01 #probability of resurrection of zombie
A = 0.1 #probability of human destroying zombie on encounter
mt = 5000 # Optional Length of time over which the model is run
zs = 4 #zombie speed
di = int(p/100) # 1% of initial population is dead
```

```
Zombie 498 kills human 369!
Zombie 73 kills human 195!
Zombie 73 kills human 466!
Zombie 185 kills human 201!
Zombie 112 kills human 433!
Human 87 destroyed zombie 402!
Zombie 452 kills human 156!
Zombie 452 kills human 341!
Zombie 452 kills human 379!
Zombie 385 kills human 69!
Zombie 385 kills human 34!
Zombie 385 kills human 273!
Human 126 destroyed zombie 21!
Human 416 destroyed zombie 377!
Zombie 372 kills human 168!
Zombie 372 kills human 203!
Zombie 372 kills human 235!
Zombie 372 kills human 486!
Zombie 372 kills human 487!
Zombie 372 kills human 298!
Zombie 111 kills human 43!
Zombie 161 kills human 120!
Human 239 destroyed zombie 161!
Zombie 469 kills human 287!
Zombie 194 kills human 5!
Zombie 194 kills human 254!
Zombie 194 kills human 442!
Zombie 194 kills human 206!
Zombie 194 kills human 318!

End result: Zombies 496, Humans 2, Dead 0, Destroyed 2 in 5000 hours
```

Figure 1
Zombie infection simulation with random walk of zombies



APPENDIX

```python
import numpy as np
import random
import matplotlib.pyplot as plt

def random_step(x,y):
    dx = random.choice([-1,0,1])
    dy = random.choice([-1,0,1])
    x += dx
    y += dy
    return (x,y)

def walk_subset(subset, d):
    for key, value in d.items():
        if key in subset:
            value.append(random_step(value[-1][0], value[-1][1]))
            #value.pop(1) #to only keep most recent coordinate otherwise comment out
    return d


def compare_locations(lista, listb):
    listadict = dict((k, population_locs[k][-1]) for k in lista)
    listbdict = dict((k, population_locs[k][-1]) for k in listb)
    listacoords = [v for v in listadict.values()]
    listbcoords = [v for v in listbdict.values()]
    listaout = [key for key, value in listadict.items() if value in listbcoords]
    listbout = [key for key, value in listbdict.items() if value in listacoords]
    return listaout, listbout


def outcome_test(prob):
    return random.random() < prob

def switch_list_vals(val, removelist, addlist):
    removelist.remove(val)
    addlist.append(val)
    return removelist, addlist

def return_outcomes(input_list, prob):
    newl = []
    for i in input_list:
        if outcome_test(prob):
            input_list, newl = switch_list_vals(i, input_list, newl)
    return input_list, newl
```

```python
def search_meetings(h, z, d, r):
    hmeet, zmeet = compare_locations(h, z)
    for hm in hmeet:
        for zm in zmeet:
            if hm in h and zm in z: #make sure not already changed state
                if outcome_test(A):
                    print("Human {} destroyed zombie {}!".format(hm, zm))
                    switch_list_vals(zm, z, r)
                else:
                    print("Zombie {} kills human {}!".format(zm, hm))
                    switch_list_vals(hm, h, d)
    return h, z, d, r

def add_current_counts(res, vals):
    for i in range(0,len(vals)):
        res[i].append(vals[i])
    return res

def compare_vals(v1, v2):
    if v1 < v2:
        return -1
    elif v1 > v2:
        return +1
    else:
        return random.randint(-1, 1)

#For human movement away from nearest zombie by Euclidean distance:
def move_humans(humans, zombies, population_locs):
    zombielocs = np.asarray([population_locs[i][-1] for i in zombies])
    for h in humans:
        coord = population_locs[h][-1]
        closest_zom = population_locs[zombies[np.argmin(np.sum((zombielocs -
coord)**2, axis=1))]][-1]
        population_locs[h].append((coord[0] + compare_vals(coord[0],closest_zom[0]),
coord[1] + compare_vals(coord[1], closest_zom[1])))
    return population_locs



p = 500 #starting population

d = 0.001 # probability of natural death
G = 0.01 #probability of resurrection of zombie
A = 0.2 #probability of human destroying zombie on encounter

mt = 2000 # Optional Length of time over which the model is run
zs = 5 #Number of zombie moves per time increment (read zombie speed)
```

```
di = int(p/100) # 1% of initial population is dead

# generate initial location dictionary for population on 100X100 grid
population_locs = dict([(i, [(random.randint(0, 101), random.randint(0,101))]) for i
in range(0,p)])

#Create initial lists
humans = [i for i in range(0,len(population_locs) - di)] #all pop but 1% initial dead
are human
dead = [i for i in population_locs if i not in humans] #dead is initial 1%
zombies = [] #No initial zombies in population
removed = [] #removed are zombies destroyed by humans

results = [[],[],[],[],[]]

"""Run while loop for the simulation...."""
t=0 #initialise t for while loop

#while len(humans) > 0 and t < mt: #if only want to run for specified time period
while len(humans) > 0: #or comment out above and run until all humans gone
    if len(zombies) > 0:
        population_locs = move_humans(humans, zombies, population_locs)
    humans, new_died = return_outcomes(humans, d)
    dead += new_died
    dead, new_zombies = return_outcomes(dead, G)
    zombies += new_zombies
    # check if humans / zombies meet, and update lists, walk zombies and check again
    for s in range(0, zs + 1):
        # check if zombies in same location as humans
        humans, zombies, dead, removed = search_meetings(humans, zombies, dead,
removed)
        # walk zombies
        population_locs = walk_subset(zombies, population_locs)

    results = add_current_counts(results, [t, len(humans), len(dead), len(zombies),
len(removed)])
    t += 1

print("End result: Zombies {}, Humans {}, Dead {}, Destroyed {} in {} hours"
    .format(len(zombies), len(humans),len(dead), len(removed),(t)))


results_arr = np.array(results)

#plot changes in totals over time
plt.figure()
plt.plot(results_arr[0], results_arr[1], label='Living humans')
plt.plot(results_arr[0], results_arr[3], label='Zombies')
plt.plot(results_arr[0], results_arr[2], label='Dead humans')
plt.plot(results_arr[0], results_arr[4], label='Destroyed zombies')
plt.xlabel('Hours from outbreak')
plt.ylabel('Population')
plt.title('Figure 1 \nZombie infection simulation with random walk of zombies')
plt.legend(loc=0)
plt.show()

####Plot zombie walk

zombie_walk1_x = [i[0] for i in population_locs[482]]
zombie_walk1_y = [i[1] for i in population_locs[482]]

zombie_walk2_x = [i[0] for i in population_locs[5]]
zombie_walk2_y = [i[1] for i in population_locs[5]]


plt.figure()
bbox_props = dict(boxstyle="round", fc="w", ec="0.5", alpha=0.5, pad = 0.1)
plt.text(zombie_walk1_x[0], zombie_walk1_y[0], 'start', color =
[0,0,1],bbox=bbox_props)
plt.text(zombie_walk1_x[-1], zombie_walk1_y[-1], 'end', color =
[0,0,1],bbox=bbox_props)
plt.text(zombie_walk2_x[0], zombie_walk2_y[0], 'start', color =
[1,0,0],bbox=bbox_props)
plt.text(zombie_walk2_x[-1], zombie_walk2_y[-1], 'end', color =
[1,0,0],bbox=bbox_props)
plt.plot(zombie_walk1_x, zombie_walk1_y, lw = 0.5, label = 'zombie walk 1')
plt.plot(zombie_walk2_x, zombie_walk2_y, lw = 0.5, color = [1,0,0], label = 'zombie
walk 2')
plt.xlabel('X coord')
plt.ylabel('Y coord')
plt.title('Figure 2 \nRandom walk of two zombies on 100X100 grid')
plt.legend(loc=0)
plt.show()

##Plot human fleeing zombies

human_walk1_x = [i[0] for i in population_locs[420]]
human_walk1_y = [i[1] for i in population_locs[420]]

human_walk2_x = [i[0] for i in population_locs[424]]
human_walk2_y = [i[1] for i in population_locs[424]]

plt.figure()
bbox_props = dict(boxstyle="round", fc="w", ec="0.5", alpha=0.5, pad = 0.1)
```

```
plt.text(human_walk1_x[0], human_walk1_y[0], 'start', color =
[0,0,1],bbox=bbox_props)
plt.text(human_walk1_x[-1], human_walk1_y[-1], 'end', color =
[0,0,1],bbox=bbox_props)
plt.text(human_walk2_x[0], human_walk2_y[0], 'start', color =
[0,0,0],bbox=bbox_props)
plt.text(human_walk2_x[-1], human_walk2_y[-1], 'end', color =
[0,0,0],bbox=bbox_props)
xlocs = [i[-1][0] for i in population_locs.values()]
ylocs = [i[-1][1] for i in population_locs.values()]
axes = plt.gca()
axes.set_xlim([-1000,300])
axes.set_ylim([0,650])
plt.scatter(xlocs, ylocs, color = [1,0,0], s=[0.9])
plt.plot(human_walk1_x, human_walk1_y, lw = 0.5, label = 'Human walk 1')
plt.plot(human_walk2_x, human_walk2_y, lw = 0.5, label = 'Human walk 2', color =
[0,0,0])
plt.xlabel('X coord')
plt.ylabel('Y coord')
plt.title('Figure 3 Deliberate Walk of Humans Away from Zombies')
plt.legend(loc=0)
plt.show()
```