P6

edgardjfc

March 2022

1 Introduction

For this assignment we studied the behaviour of a population of particles in a pandemic, where an initial amount of infected particles spread out the contagion to other particles. With this code we could observe the development of the infected people over time and how the particles recovered from the infection. A goal for this assignment was to observe how social distancing could affect the infection, by making the infected particles move at half the speed of the regular experiment. With the initial code done by our teacher E. Schaeffer [2].

2 Code

```
l = 1.5
n = 50
pi = 0.05
pr = 0.02
v = l / 30
r = 0.1
tmax = 100
runs = 10
```

With these initial conditions we can create the parameters for the behaviour of the particles in the code.

```
def contagiados():
    for i in range(n):
        a1 = agentes.iloc[i]
        if a1.estado == 'I':
            for j in range(n):
                a2 = agentes.iloc[j]
                if a2.estado == 'S':
                    d = sqrt((a1.x - a2.x)**2 + (a1.y - a2.y)**2)
                    if d < r:
                        if random() < (r - d) / r:
```

```
                            contagios[j] = True
    return contagios
```

This function is in charge of designating the first infected particles that will kickstart the pandemic.

```
 agentes['x'] = [uniform(0, l) for i in range(n)]
    agentes['y'] = [uniform(0, l) for i in range(n)]
    agentes['estado'] = ['S' if random() > pi else 'I' for i in range(n)]
    agentes['dx'] = [uniform(-v, v) for i in range(n)]
    agentes['dy'] = [uniform(-v, v) for i in range(n)]
```

Here we see the code in charge of placing randomly the particles across the space and developing the movement by giving them a value for their velocity. This was used for the control group of the experiment where particles do not perform "social distancing".

```
agentes['x'] = [uniform(0, l) for i in range(n)]
    agentes['y'] = [uniform(0, l) for i in range(n)]
    agentes['estado'] = ['S' if random() > pi else 'I' for i in range(n)]
    agentes['dx'] = [uniform(-v/2, v/2) if  agentes.at[i, 'estado'] == 'I'\
                     else uniform(-v, v) for i in range(n)]
    agentes['dy'] = [uniform(-v/2, v/2) if  agentes.at[i, 'estado'] == 'I'\
                     else uniform(-v, v) for i in range(n)]
```

For the speed variable of the experiment we changed this part of the code. We can observe the changes done to the dx and dy components that dictate the velocity of each particle, this was done to half the average velocity at which every infected particle moves. This changed was taken from the work done by the classmate with the alias FeroxDeitas [1].

```
epidemia = []
    for tiempo in range(tmax):
        conteos = agentes.estado.value_counts()
        infectados = conteos.get('I', 0)
        epidemia.append(infectados)
        contagios = [False for i in range(n)]
        if infectados == 0:
            break
        contagios = contagiados()
```

This part of the code is in charge of counting the infected at any given time, until the number of infected becomes 0, ending the cycle.

```
for i in range(n):
            a = agentes.iloc[i]
            if contagios[i]:
```

```
            agentes.at[i, 'estado'] = 'I'
        elif a.estado == 'I':
            if random() < pr:
                agentes.at[i, 'estado'] = 'R'
```

This part of the code calculates the probability for each particle to recover from the infection and stop spreading it to others.

```
for i in range(n):
            a = agentes.iloc[i]
            if contagios[i]:
                agentes.at[i, 'dx'] /= 2
                agentes.at[1, 'dy'] /= 2
                agentes.at[i, 'estado'] = 'I'
            elif a.estado == 'I':
                if random() < pr:
                    agentes.at[i, 'estado'] = 'R'
```

Here in the alternate code we add two lines in order to half the movement for the infected particles.

```
x = a.x + a.dx
            y = a.y + a.dy
            x = x if x < l else x - l
            y = y if y < l else y - l
            x = x if x > 0 else x + l
            y = y if y > 0 else y + l
            agentes.at[i, 'x'] = x
            agentes.at[i, 'y'] = y
```

This part of the code calculates the movement for each particle.

3 Results

4 Interpretation

We see each figure representing the average behaviour for each experiment. We have the violin graph for the max infected and the position of the peak for both figures, showing us the point in the experiment where the infection is the highest.

5 Conclusion

We can see the dramatic difference of the max amount of infected for both cases, where the second case of fig. 2 we see a lot less infected, due to the change in behaviour.

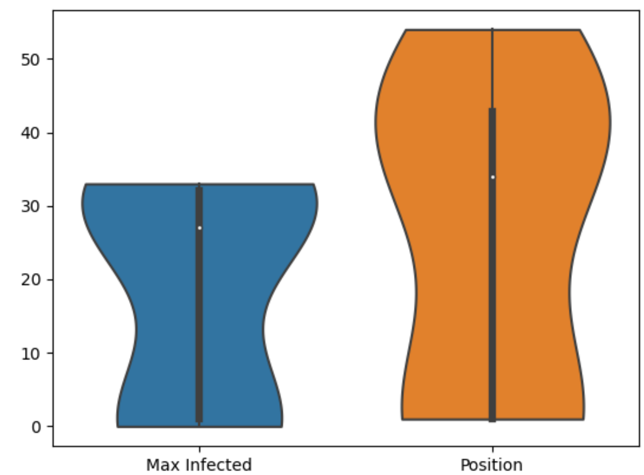

Figure 1: No change in behaviour

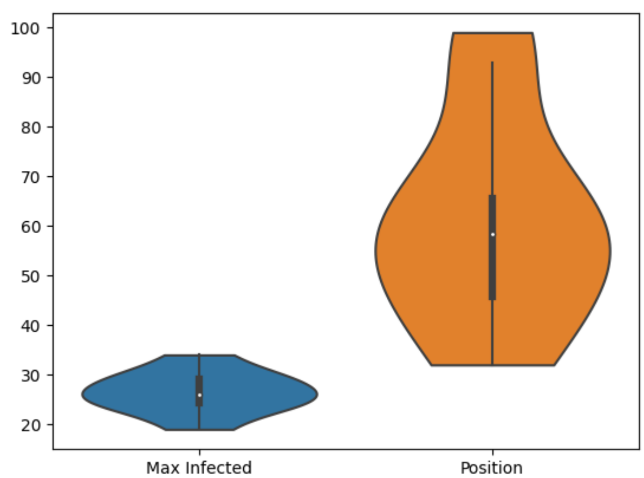

Figure 2: Social distancing as infected

We also see how the position of the peak of each pandemic changes, where the peak for the fig. 1 happens much sooner than the peak of fig. 2. We could attribute this to the speed of the particles, where the peak infection happens a lot sooner.

References

[1] J.FeroxDeitas. *GitHub,P6*. URL: `https://github.com/FeroxDeitas/Simulacion-Nano/blob/main/Tareas/P6`.

[2] E. Schaeffer. *GitHub,MonteCarlo*. URL: `https://github.com/satuelisa/Simulation/tree/master/MultiAgent`.