José María de Albornoz Caratozzolo

Joshua Daniel Hernandez Coronado

Ronaldo Angel Lopez Diaz

Sebastián Gonzalez Figueroa

Oscar Alejandro Ángeles Ramirez

Reporte

El código utiliza una clase llamada Persona que tiene un atributo de estado (healthy, infected, vaccinated, dead) y un atributo de cuantos días se lleva recuperando.

La población se representa con una lista de objetos de clase Persona, sobre la cual se va iterando para actualizar los estados de las personas de acuerdo con las reglas del problema.

Cada día se itera sobre la población y se actualizan sus estados, tomando en cuenta las vacunas, las muertes, las recuperaciones y las infecciones, en ese orden.

Se guarda el número de infectados, saludables, vacunados y muertos en arreglos que son el resultado de la simulación.

Para los histogramas, se corren las 100 simulaciones y se almacenan las cifras de infectados y saludables del último día de cada simulación.

Código:

import numpy as np

import random as rd

import matplotlib.pyplot as plt

simulaciones=100

dias=30

N=1000

inf\_0=1

vacs=3

min\_recovery=14

p\_infect=0.00027

p\_muerte=0.01

p\_recovery=0.2

class persona:

    def \_\_init\_\_(self, estado, dias\_rec):

        self.estado=estado #h,i,d,v

        self.dias\_rec=0

def evolucionar(Poblacion):

    i=0

    while i<3:

        indice=rd.randint(0,N-1)

        if Poblacion[indice].estado=="h":

            i+=1

            Poblacion[indice].estado="v"

    infectados=0

    for pers in Poblacion:

        if pers.estado=="i":

            infectados+=1

            if rd.random()<=p\_muerte:

                pers.estado="d"

            else:

                pers.dias\_rec+=1

            if pers.dias\_rec>=min\_recovery and rd.random()<=p\_recovery:

                pers.estado="h"

                pers.dias\_rec=0

    for pers in Poblacion:

        if pers.estado=="h":

            if rd.random()<=p\_infect\*infectados:

                pers.estado="i"

    return Poblacion

def simulacion():

    Poblacion=[]

    for i in range(N):

        Poblacion.append(persona("h", 0))

    Poblacion[rd.randint(0,N-1)].estado="i"

    H=[]

    I=[]

    V=[]

    D=[]

    for k in range(dias):

        h=0

        i=0

        v=0

        d=0

        Poblacion=evolucionar(Poblacion)

        for pers in Poblacion:

            if pers.estado=="h":

                h+=1

            elif pers.estado=="i":

                i+=1

            elif pers.estado=="v":

                v+=1

            elif pers.estado=="d":

                d+=1

        #print(k,h,i,v,d)

        H.append(h)

        I.append(i)

        V.append(v)

        D.append(d)

    return H,I,V,D

def hist():

    He=[]

    In=[]

    Va=[]

    De=[]

    for i in range(simulaciones):

        H,I,V,D=simulacion()

        He.append(H[-1])

        In.append(I[-1])

        Va.append(V[-1])

        De.append(D[-1])

    return He,In,Va, De

def hist\_plot():

    fig, axs = plt.subplots(2, 2, figsize=(12, 8))

    fig.suptitle('Distribution of Final States Across Simulations')

    state\_labels = ['Healthy', 'Infected', 'Dead', 'Vaccinated']

    colors = ['green', 'orange', 'red', 'blue']

    He,In,Va,De=hist()

    cosa=[He,In,De,Va]

    for i, ax in enumerate(axs.flat):

        ax.hist(cosa[i], bins=10, color=colors[i], alpha=0.7)

        ax.set\_title(f'{state\_labels[i]} Individuals')

        ax.set\_xlabel('Number of Individuals')

        ax.set\_ylabel('Frequency')

    plt.tight\_layout(rect=[0, 0, 1, 0.96])

    plt.show()

hist\_plot()

Resultados:

A group of colored bars

Description automatically generated with medium confidence