Sistema multiagente

Francisco Gerardo Meza Fierro

1. Introducción

En esta práctica se implementará un sistema multiagente aplicado a epidemiología. Se trabajará con un modelo SIR, donde sus agentes, clasificados como susceptibles, infectados y recuperados, se mueven sobre un toroide y propagan cierta epidemia: los agentes infectados, bajo cierta probabilidad p_i, contagian a los agentes susceptibles y los agentes infectados se recuperan bajo cierta otra probabilidad p_r; los agentes recuperados ya no podrán infectarse ni tampoco podrán propagar la epidemia. Además, se intentará paralelizar el código propuesto, se agregará un cuarto agente (vacunado) el cual aparecerá con cierta probabilidad p_v al inicio de la simulación y no podrá infectarse ni propagar la epidemia y finalmente se estudiará con la versión que se intentó paralelizar el efecto de la probabilidad p_i en el porcentaje de infectados al final de la simulación.

2. Paralelización del código

La intención que se tiene al paralelizar un código es repartir el trabajo que hará el código en distintos núcleos de la computadora, con objetivo de disminuir el tiempo requerido en que se ejecute el código. En este caso, se encontraron dos segmentos del código propuesto candidatos a ser paralelizados debido a que podrían ser, digamos, excluidos del código y ser llamados a él mediante funciones. A continuación se muestran las dos funciones que se crearon a los dos mencionados segmentos:

```
contagiar <- function(){
  if (!contagios[j]) {
    a2 <- agentes[j, ]
    if (a2$estado == "S") {
      dx <- a1$x - a2$x
      dy <- a1$y - a2$y
      d <- sqrt(dx^2 + dy^2)
      if (d < r) {
        p <- (r - d) / r
        if (runif(1) < p) {
          return(TRUE)
        }
        else{
          return(FALSE)
        }
      }
      return(FALSE)
    }
    else if(a2$estado == "I" || a2$estado == "R") {
      return(TRUE)
    }
  }
  else{
    return(TRUE)
  }
}
```

```
actualizar <- function(){
  a <- agentes[i, ]
  if (contagios[i] & a$estado == "S") {
    a$estado <- "I"
  } else if (a$estado == "I") {
    if (runif(1) < pr) {
      a$estado <- "R"
    }
  }
  a$x <- a$x + a$dx
  a$y <- a$y + a$dy
  if (a$x > 1) {
    a$x <- a$x - 1
  }
  if (a$y > 1) {
    a$y <- a$y - 1
  }
  if (a$x < 0) {
    a$x <- a$x + 1
  }
  if (a$y < 0) {
    a$y <- a$y + 1
  }
  return(as.vector(a))
}
```

Teniendo identificados esos segmentos, se procedió a paralelizar haciendo uso del paquete `parallel`, pero al momento de correr el código, a pesar de compilar sin errores, el único error que se tuvo fue que no actualizaba los estados de los agentes. Pensado que este error se debía a la falta de conocimientos sólidos en programación, se decidió hacer uso del paquete `foreach` y, a diferencia de los resultados anteriores, el error que se tenía fue solucionado, por lo que el resto de la práctica se continuó usando el paquete `foreach`.

Otro dato interesante que hay que resaltar es que, pese a estar paralelizado, haciendo uso de la función `system.time`, el tiempo de ejecución no disminuyó ni se asemejó a comparación del tiempo de ejecución del código propuesto, sino que aumentó drásticamente. Pensando que el error se debía a que alguna de las dos funciones mencionadas no debiera ser paralelizada, se procedió a analizar los tiempos de ejecución al hacer dos paralelizaciones nuevas, una para cada función esperando obtener tiempos menores al del código propuesto; nuevamente, para ambas paralelizaciones, se obtuvieron tiempos mayores al del código propuesto, aunque menores al del primer código paralelizado. Comparando estos dos nuevos tiempos, el menor de ellos fue cuando únicamente se paralelizó la función `actualizar`, por lo que para el resto de la práctica se trabajará con esa última paralelización.

3. Vacunar los agentes

Ahora se propone que al inicio de la simulación existan agentes que estén vacunados, es decir que estén recuperados; esto es que desde el inicio no puedan ser contagiados. La figura 1 refleja una idea de cómo se vería el inicio de una simulación implementado estos agentes vacunados (los puntos verdes son agentes susceptibles, los rojos son infectados y los amarillos son los recuperados los cuales, al estar presentes al inicio, representan los vacunados).

Para esto, simplemente se agregó una probabilidad más p_v. Para ver qué efecto tendría el ir aumentando el valor de esa probabilidad, se agregó una función `for` que permitiera variar la probabilidad, teniendo diez valores distintos entre 0,05 y 0,5. Ahora se propone que al inicio de la simulación existan agentes que estén vacunados, es decir que estén recuperados; esto es que desde el inicio no puedan ser contagiados. Para esto, simplemente se agregó una probabilidad más p_v. Para ver qué efecto tendría el ir aumentando el valor de esa probabilidad, se agregó una función `for` que permitiera variar la probabilidad, teniendo diez valores distintos entre 0,05 y 0,5.

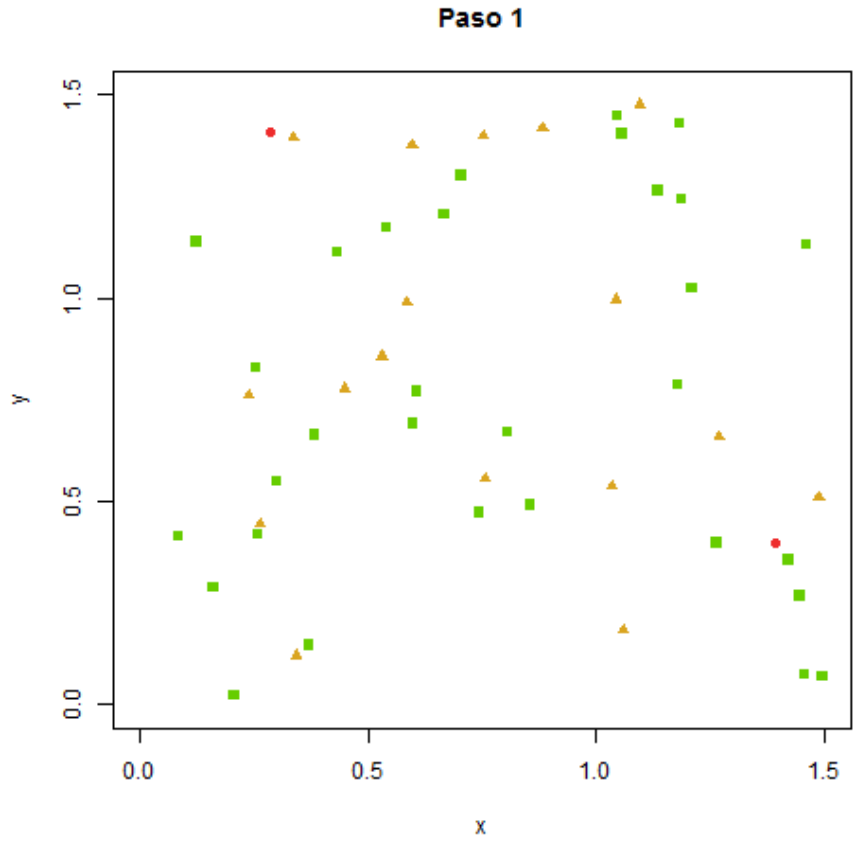

Figura 1: Porcentajes máximos de agentes infectados por probabilidad

El cuadro 1 muestra los porcentajes máximos y mínimos de los agentes infectados a lo largo de cada una de las simulaciones creadas.

Cuadro 1: Porcentajes de infectados

Probabilidad	Porcentaje máximo	Porcentaje mínimo
0.05	0.58	0.06
0.10	0.52	0.02
0.15	0.6	0.04
0.20	0.5	0.02
0.25	0.48	0.02
0.30	0	0
0.35	0.38	0.04
0.40	0.54	0.04
0.45	0.4	0.04
0.50	0.04	0

El porcentaje máximo en la probabilidad de 0,30 resultó cero porque al inicio de la simulación no surgió ningún agente infectado bajo la probabilidad $p_i = 0{,}05$.

Se esperaba que a mayor la probabilidad de que un agente estuviera vacunado al inicio, el porcentaje máximo de agentes infectados alcanzado a lo largo de esa corrida fuera menor, y así lo fue con un porcentaje del cuatro por ciento.

Teniendo ahora un valor para el cual el porcentaje máximo de infectados es muy cercano a cero, resultaría interesante ahora cómo afectaría a los agentes si se incrementa la probabilidad

de que uno de ellos resulte infectado al inicio de la simulación. De manera análoga se varió la probabilidad p_i con ayuda de una función `for` que lo hacía variar entre valores de 0,05 y 0,50. Para cada probabilidad p_i distinta se crearon diez replicas, esperando que a mayor la probabilidad p_i, mayor el porcentaje máximo de agentes infectados. Y en efecto estos resultados se obtuvieron, los cuales están reflejados en la figura 2.

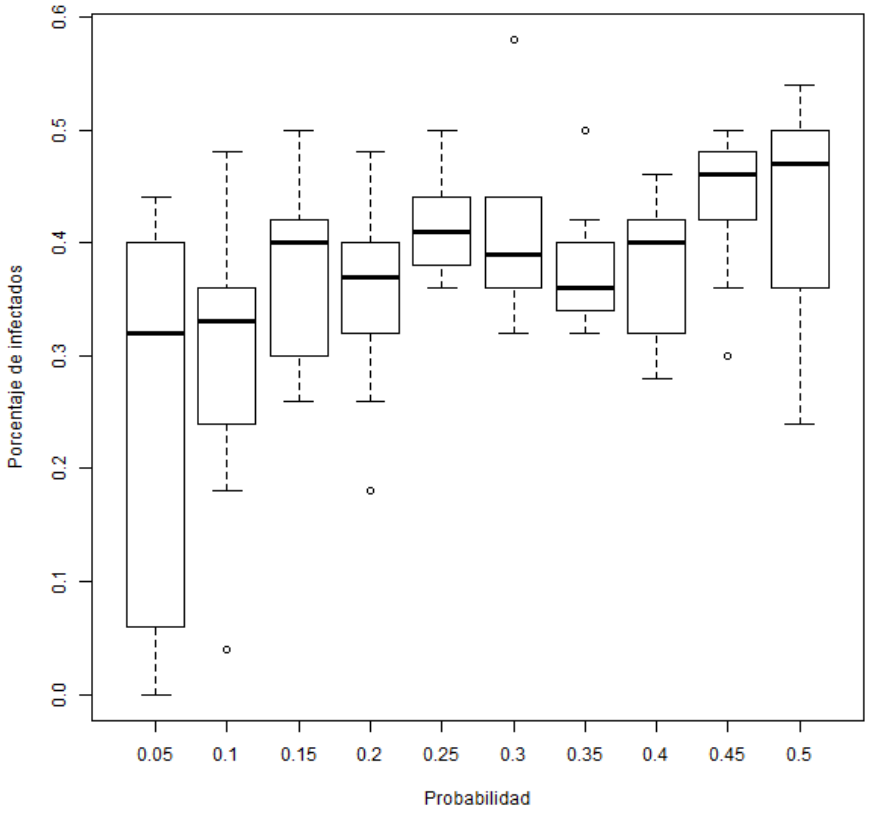

Figura 2: Porcentajes máximos de agentes infectados por probabilidad