Sistema multiagente

Alberto Benavides

17 de septiembre de 2017

Objetivos

1. Paralelizar un sistema multiagente en el que haya agentes Susceptibles, Infectados y Recuperados en R con alguno de los paquetes disponibles para hacerlo.
2. Vacunar con cierta probabilidad agentes desde su creación asignándoles el estado de Recuperado.
3. Estudiar el efecto de la probabilidad inicial de infección en el porcentaje máximo de infectados durante la simulación.

Simulación y resultados

Esta práctica se corrió en una computadora portátil con sistema operativo Windows 10 Home Single Language, procesador Intel(R) Core(TM) i7-7500U CPU @ 2.70 GHz, 2904 MHz de dos núcleos principales y cuatro lógicos.

El sistema multiagente se simuló en un espacio bidimensional cuyos extremos se comunican como si de un toroide se tratara. En este espacio se sitúan de manera aleatoria cincuenta agentes que toman una ubicación bidimensional al azar con valores entre cero y uno, un vector de velocidad aleatorio cuyas componentes horizontal y vertical adquieren velocidades de entre $-1/20$ a $1/20$. Los posibles estados de estos agentes son **susceptible**, **infectado** y **recuperado**. Un agente susceptible puede ser infectado con una probabilidad (p_c) que depende de la distancia entre éste y un agente infectado (d), tal como expresa la fórmula

$$p_c = \frac{0.1 - d}{d}$$

donde p_c toma un valor de 0 para $d = 0$.

Se ha decidido iterar la probabilidad inicial de infección de 0.05 a 0.5 en pasos de 0.05 con un máximo de cien generaciones para cada una de estas probabilidades iniciales. Además, la probabilidad inicial de un agente vacunado, cuyo estado se

actualizará a recuperado, es de 0.03 y su probabilidad de recuperarse una vez infectado es de 0.02 en cada generación que permanezca infectado.

```
totalAgents <- 50
maxVelocity <- 1 / 20
recuperationProbability = 0.02
vaccineProbability = 0.03
infectionRadius <- 0.1
maxTime <- 100

agents <- data.frame(
  x = runif(totalAgents),
  y = runif(totalAgents),
  dx = runif(totalAgents, -maxVelocity, maxVelocity),
  dy = runif(totalAgents, -maxVelocity, maxVelocity),
  state = sample(
    c("S", "I", "R"),
    totalAgents,
    replace = T,
    prob = c(
      1 - infectionProbability - vaccineProbability,
      infectionProbability,
      vaccineProbability
    )
  )
```

Una vez generados los agentes con estas condiciones, se pasan por la función `levels` para poder tratar como factores los caracteres con que se designan sus estados. Como `levels` también posee la propiedad de cambiar los caracteres que se tienen asignados alfabéticamente por los que se especifican en el orden dado, es necesario hacer una comprobación de los estados presentes para evitar cambiar accidentalmente los estados iniciales.

```
if(nrow(aR) > 0 & nrow(aI) > 0){
  levels(agents$state) <- c("I", "R", "S")
} else if(nrow(aR) == 0 & nrow(aI) == 0){
  levels(agents$state) <- c("S", "I", "R")
} else if(nrow(aR) > 0){
  levels(agents$state) <- c("R", "S", "I")
} else if(nrow(aI) > 0){
  levels(agents$state) <- c("I", "S", "R")
}
```

Ahora bien, la función a paralelizar incluye los cálculos concernientes a los contagios y actualizaciones de posición de los agentes, de modo que cada generación se comprueba si los agentes susceptibles pueden ser infectados de la manera antes descrita; si los infectados pueden recuperarse con base en la tasa de recuperación definida; y, posteriormente, se actualizan sus posiciones a partir de la velocidad

que inicialmente se les había asignado de manera aleatoria.

```
update <- function(){
  agent <- agents[i, ]
  if(agent$state == "S"){
    for (j in 1:totalAgents) {
      infectedAgent <- agents[agents$state == "I", ]
      d = sqrt(
        (agent$x - infectedAgent$x) * (agent$x - infectedAgent$x) +
        (agent$y - infectedAgent$y) * (agent$y - infectedAgent$y)
      )
      if(d < infectionRadius){
        p <- (infectionRadius - d) / infectionRadius
        if(runif(1) < p){
          agent$state <- "I"
        }
      }
    }
  } else if(agent$state == "I"){
    if(runif(1) < recuperationProbability){
      agent$state <- "R"
    }
  }
  agent$x <- agent$x + agent$dx
  agent$y <- agent$y + agent$dy
  if(agent$x > space){
    agent$x <- agent$x - space
  } else if(agent$x < 0){
    agent$x <- agent$x + space
  }
  if(agent$y > space){
    agent$y <- agent$y - space
  } else if(agent$y < 0){
    agent$y <- agent$y + space
  }
  return(agent)
}
```

Esta función se paraleliza un total de cincuenta veces, o sea el número de agentes que se había definido al inicio, con tres núcleos en un `foreach` que utiliza la función `%dopar%` del paquete `doParallel` y sus posiciones y estados actualizados se almacenan para ser tratados la siguiente iteración hasta que finalice el ciclo.

```
cluster <- makeCluster(detectCores() - 1)
registerDoParallel(cluster)
clusterExport(cluster, "agents")
nextGeneration <- foreach(i = 1:totalAgents, .combine=rbind) %dopar% update()
```

```
agents <- nextGeneration
stopImplicitCluster()
```

Con la finalidad de dar una idea visual de lo que sucede a los agentes con el paso de las generaciones, se registraron los agentes en una gráfica bidimensional en todas las generaciones para la probabilidad inicial de 0.05 infectados. En esta representación se asignó el cuadrado color verde a los agentes susceptibles, el círculo rojo a los infectados y el triángulo naranja a los recuperados. En la figura 1 (p. 4) se recopilaron las generaciones 1, 25, 50 y 75.

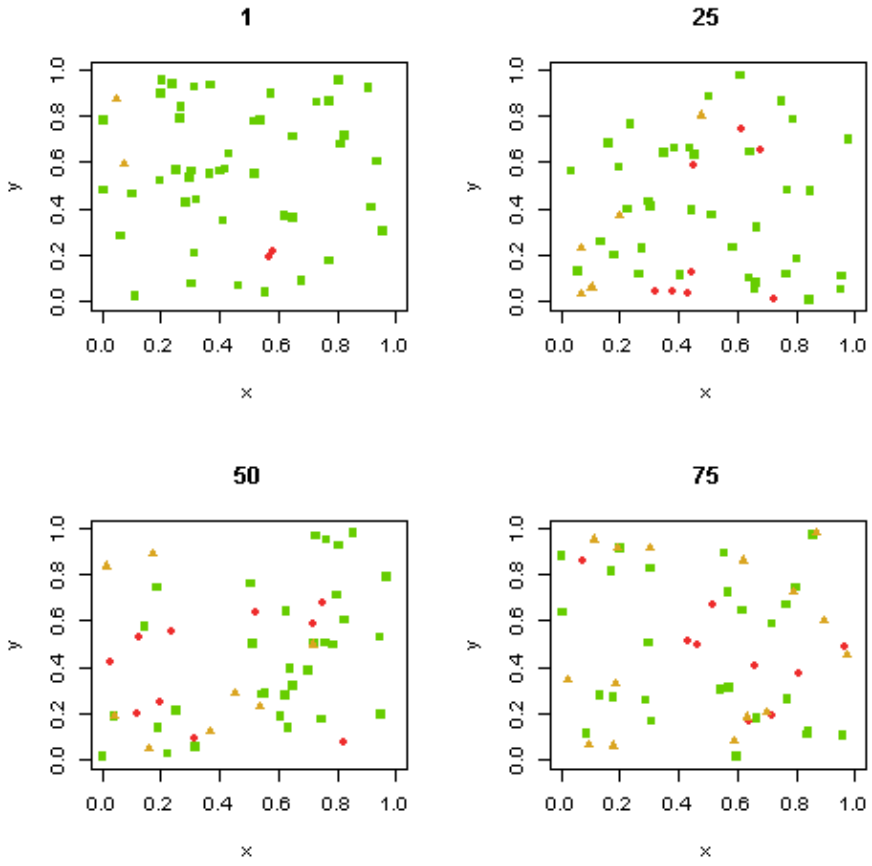

Figura 1: Representación en un espacio bidimensional de los agentes susceptibles (cuadrados verdes), infectados (círculos rojos) y recuperados (triángulos naranjas) en las generaciones 1, 25, 50 y 75 para la probabilidad inicial de 0.05 infectados.

El porcentaje de infectados por generación se almacena de manera que, transcurridas cien generaciones o acabados los infectados, se puedan comparar estos datos por porcentaje inicial de infectados. La figura 2 (p. 5) muestra los porcentajes de

infectados a lo largo del tiempo para todas las probabilidades iniciales con que se realizó el experimento. En este gráfico se puede ver que en las probabilidades iniciales de infección de 0.05 y 0.1 crecen los porcentajes de infectados más de 20 % hasta estabilizarse alrededor de las veinticinco generaciones, manteniéndose en un rango entre 30 % y 45 % la primera y entre 18 % y 25 % la segunda. Las probabilidades iniciales comprendidas entre 0.15 y 0.35, ambos incluidos, se mantienen en rangos que no superan el 20 % de infectados, salvo para la probabilidad inicial de 0.2 que a las dieciséis generaciones superó en 20 % su estado inicial para luego descender por debajo de su porcentaje inicial de infectados a partir de las sesenta generaciones. Finalmente, las probabilidades iniciales de 0.4, 0.45 y 0.5 inician con los mayores porcentajes de infectados, sin embargo, a partir de las veinte generaciones sus porcentajes disminuyen y no dejan de hacerlo a lo largo de las restantes iteraciones hasta llegar a 30 %, 20 % y 10 % de infectados respectivamente.

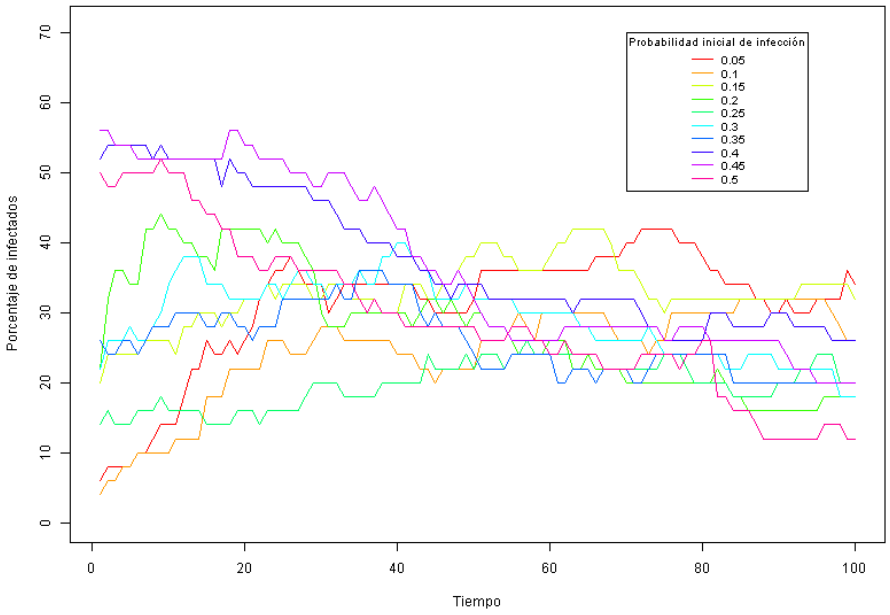

Figura 2: Resumen de porcentajes de infección totales por probabilidades de infección iniciales para esta simulación.

Conclusiones

1. Probabilidades iniciales de infectados menores a 0.2 crecen en las primeras veinticinco generaciones para estabilizarse después.
2. Probabilidades iniciales de infectados entre 0.2 y 0.35, ambos incluidos, mantienen en su mayoría constantes los porcentajes de infectados a lo largo de las cien primeras generaciones.
3. Probabilidades de infectados iniciales mayores o iguales a 0.4 disminuyen continuamente sus infectados en las cien primeras generaciones.