Reporte Práctica 2

Autómata Celular

Juan Pablo Chávez Granados

A 22 de Agosto de 2017

Autómata celular

El autómata celular (A.C.) es un modelo matemático para un sistema dinámico que evoluciona con el tiempo. Este tipo de modelo es adecuado para el modelado de sistemas naturales que pueden ser descritos como una colección masiva de objetos simples que interactúan unos con otros.

Estos sistemas fueron descubiertos dentro del campo de la física computacional por John von Neumann en la década de 1950.

La utilidad de los autómatas celulares les da diversas aplicaciones ya que pueden ser utilizados para modelar numerosos sistemas físicos, estos sistemas deben caracterizarse por tener un gran número de componentes homogéneas y que interactúen entre sí. Los candidatos para el uso de estos modelos a los que se les puedan aplicar los conceptos "vecindad", "estados de los componentes" y "función de transición".

Algunos ejemplos de áreas en donde se utilizan los autómatas celulares son:

- Modelado del flujo de tráfico y de peatones.
- Modelado de fluidos (gases o líquidos).
- Modelado de la evolución de células o virus como el VIH.
- Modelado de procesos de percolación.[1,2]

Práctica

En la práctica de autómata celular, se ha analizado cuidadosamente cada detalle del autómata ejemplo del juego de la vida, donde el juego de la vida dice que se tendrá una celda viva si se tienen tres vecinos vivos. Se realizó un experimento para determinar el número de iteraciones que dura la simulación sin que se mueran todas las celdas en función de la probabilidad inicial de celda viva, esto se realizó modificando el código inicial.

- Código inicial

```
library(parallel)
dim <- 10
num <-  dim^2
actual <- matrix(round(runif(num)), nrow=dim, ncol=dim)
suppressMessages(library("sna"))
png("p2_t0.png")
plot.sociomatrix(actual, diaglab=FALSE, main="Inicio")
```

```
graphics.off()

paso <- function(pos) {
   fila <- floor((pos - 1) / dim) + 1
   columna <- ((pos - 1) %% dim) + 1
   vecindad <-  actual[max(fila - 1, 1) : min(fila + 1, dim),
                  max(columna - 1, 1): min(columna + 1, dim)]
   return(1 * ((sum(vecindad) - actual[fila, columna]) == 3))
}

cluster <- makeCluster(detectCores() - 1)
clusterExport(cluster, "dim")
clusterExport(cluster, "paso")

for (iteracion in 1:9) {
   clusterExport(cluster, "actual")
   siguiente <- parSapply(cluster, 1:num, paso)
   if (sum(siguiente) == 0) { # todos murieron
      print("Ya no queda nadie vivo.")
      break;
   }
   actual <- matrix(siguiente, nrow=dim, ncol=dim, byrow=TRUE)
   salida = paste("p2_t", iteracion, ".png", sep="")
   tiempo = paste("Paso", iteracion)
   png(salida)
   plot.sociomatrix(actual, diaglab=FALSE, main=tiempo)
   graphics.off()
}
stopCluster(cluster)
```

Este código genera números aleatorios los cuales son redondeados a ceros y unos, estos se acomodan en una matriz la cual permite ver el acomodo de las celdas, generando celdas de color negro donde se encuentran los unos y quedando en blanco los lugares ocupados por los ceros.

Posteriormente se modificó el código inicial y se utilizó el código de Beatriz Alejandra García Ramos[3] como ejemplo del uso de los “for” como generadores de loops.

- Código Pablo

```
library(parallel)
Iteraciones <- c()
data <- data.frame()
Viva <- seq(0, 0.9, 0.1)
dim <- 10
num <-  dim^2
suppressMessages(library("sna"))
```

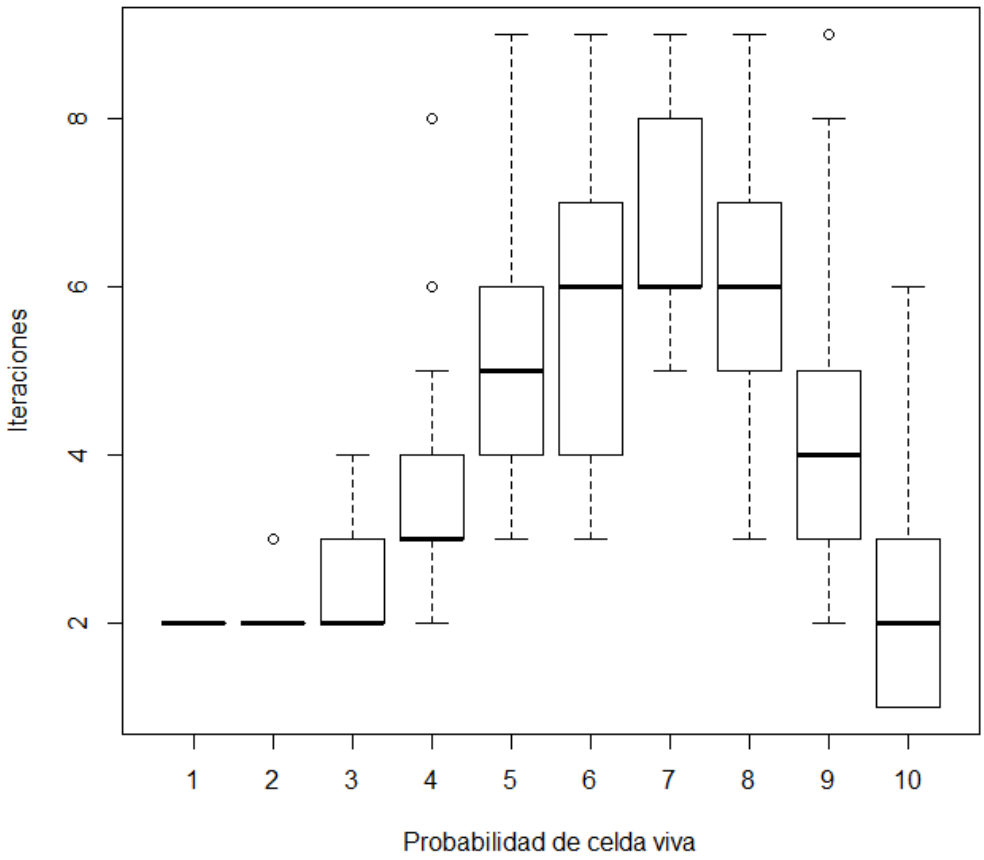
```
paso <- function(pos) {
  fila <- floor((pos - 1) / dim) + 1
  columna <- ((pos - 1) %% dim) + 1
  vecindad <-  actual[max(fila - 1, 1) : min(fila + 1, dim),
                      max(columna - 1, 1): min(columna + 1, dim)]
  return(1 * ((sum(vecindad) - actual[fila, columna]) == 3))
}

cluster <- makeCluster(detectCores() - 1)
clusterExport(cluster, "dim")
clusterExport(cluster, "paso")

for(v in Viva){
  for (i in 1:30){
actual <- matrix(1*(runif(num)> v), nrow=dim, ncol=dim)

for (iteracion in 1:9) {
  clusterExport(cluster, "actual")
  siguiente <- parSapply(cluster, 1:num, paso)
  if (sum(siguiente) == 0) { # todos murieron
    break;
  }
  actual <- matrix(siguiente, nrow=dim, ncol=dim, byrow=TRUE)

}
Iteraciones[i] <- iteracion

}
 data <- rbind(data, Iteraciones)
}
boxplot(t(data), xlab = "Probabilidad de celda viva", ylab= "Iteraciones")
stopCluster(cluster)
```

En este código se conservó activo el paquete “parallel”, el cual permite hacer múltiples cálculos haciendo uso de los núcleos en la computadora, se eliminaron las líneas de código que generaban los archivos con extensión .png, los cuales representan las matrices. En el código se generaron los vectore “Iteraciones<- c()”, para guardar los resultados de las iteraciones, una variable “viva” que sirve para formar una secuencia donde se dará una probabilidad que inicia de cero a uno con incrementos de 0.1 y por último se nombró una variable llamada “data” para guardar los datos de las iteraciones en las diferentes probabilidades.

Resultados

Como resultados se obtuvieron el mínimo, el máximo y la media de los datos de probabilidad e iteraciones y se graficaron en un boxplot. En esta se observa la media de las probabilidades y como

se distribuye en forma de campana, el código dice que mientras más celdas estén muertas al inicio habrá muy pocas o ninguna viva en la primer iteración, y sucede de la misma manera para el caso en el que la mayoría de las celdas estén vivas.



Algo más que se puede observar de la gráfica caja de bigotes es como de manera gráfica nos expresa las probabilidades donde las celdas podrán vivir más al tener una probabilidad de vida mayor al 50% pero menor al 100% de celdas vivas dando así un equlibrio de celdas en las que se puede obtener un óptimo crecimiento y mayor taza de vida para las celdas.

Bibliografía

1) von Neumann, J., (1966), *The Theory of Self-reproducing Automata*, ed. Univ. of Illinois Press, Urbana, IL.
2) Lahoz-Beltrá, Rafael, (2004), *Bioinformática: Simulación, vida artificial e inteligencia artificial*, Ed. Díaz de Santos.
3) Beatriz Alejandra García Ramos, Codigo de Github.