Extinción y reproducción de autómatas celulares en R

Un autómata celular (A.C.) es un modelo matemático para un sistema dinámico que consiste en una rejilla formada por celdas llamadas “células” que pueden cambiar de estado o no, según leyes que evalúan los estados de las células vecinas. Los autómatas celulares fueron inventados a fines de los cuarenta por Stanislaw Ulam (1909-1984) y John von Neumann (1903-1957)[1]

El juego de la vida

El juego consiste en una cuadrícula donde se coloca al inicio un patrón de células “vivas” o “muertas”. La vecindad para cada célula son las ocho celdas que la rodean. De manera repetida, se aplican simultáneamente sobre todas las células de la cuadrícula las siguientes 3 reglas:

1. Nacimiento: se reemplaza una célula muerta por una viva si dicha célula tiene exactamente 3 vecinos vivos.
2. Muerte: se reemplaza una célula viva por una muerta si dicha célula no tiene más de 1 vecino vivo (muerte por aislamiento) o si tiene más de 3 vecinos vivos (muerte por sobrepoblación).
3. Supervivencia: una célula viva permanecerá en ese estado si tiene 2 o 3 vecinos vivos.[2]

[1]von Neumann, J. (1966) The Theory of Self-Reproducing Automata, ed. Univ. of Illinois Press, Urbana.

[2]Gardner, M. (1970) Mathematical Games: The fantastic combinations of John Conway's new solitaire game "Life", Scientific American

Objetivo

Diseñar y ejecutar un experimento en el cual se determina el número de iteración que procede la simulación hasta fue mueran todas las celdas cuando se varia la probabilidad inicial viva entre ceo y uno en pasos de 0.05.

Retos

1. Modificar la simulación para lograr algún crecimiento en la microestructura de un material, estableciendo que los núcleos que inician en una sola celda se van expandiendo a una tasa constante a celdas vecinas hasta llenar por completo la matriz, añadiendo todos los núcleos al inicio y expandiéndose todos con la misma tasa decrecimiento.
   Examinar la distribución de los tamaños de los núcleos que no toquen el borde al finalizar la simulación, seleccionando el tamaño de la zona y el numero de semillas de tal forma que sean por lo menos la mitad

2. Modificar el reto anterior a que nuevos núcleos puedan aparecer en momentos distintos. examinar como afecta el cambio a la distribución de los tamaños

Simulaciones y Resultados

Objetivo 1

Se inicia la simulación partiendo de una matriz de 30*30, se llevan a cabo 20 iteraciones para determinar si queda alguna célula viva o si todos mueren, con un rango de exploración de 0 a 1 con un aumento de .05, dando 2 posibles resultados al final de las iteraciones deseadas:

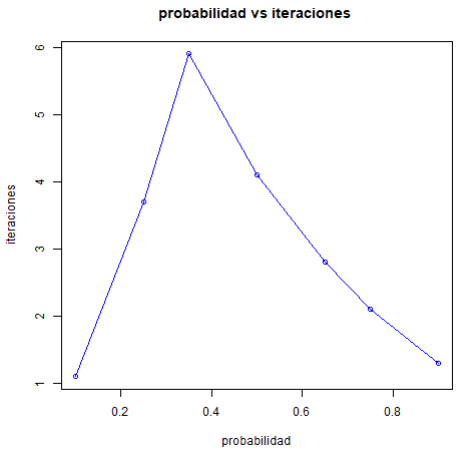
1. (1) Se encuentra 1 célula viva que está representada por una celda negra en el gráfico, mientras
2. (0) Todos Mueren están representadas por una celda blanca en el gráfico.

Numero de iteraciones.

Se plantea el análisis del número de iteraciones máximas que tiene el autómata sin llegar a la muerte, una vez que se simula el código genera y guarda los gráficos obtenidos.

```r
library(parallel)
dimensiones <- 30
probabilidades <-c(0.05, 0.1, 0.15, 0.2, 0.35, 0.40, 0.45, 0.50, 0.55, 0.60, 0.65, 0.70, 0.75, 0.80, 0.85, 0.90,
resultados <- data.frame()
veciteraciones <- c()
for(w in dimensiones){
promedioiteraciones <- c()
for(s in probabilidades){
datospromedio <- c()
for(z in 1:30){
dim <- 30
num <-  dim^2
proba <- s
actual <- matrix(round(runif(num)<proba)*1, nrow=dim, ncol=dim)
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() - 0)
clusterExport(cluster, "dim")
clusterExport(cluster, "paso")
numeroiteraciones <- 0
for (iteracion in 1:30) {
clusterExport(cluster, "actual")
siguiente <- parSapply(cluster, 1:num, paso)
if (sum(siguiente) == 0) { # todos murieron
```

Figure 1) Código de simulación para el objetivo 1



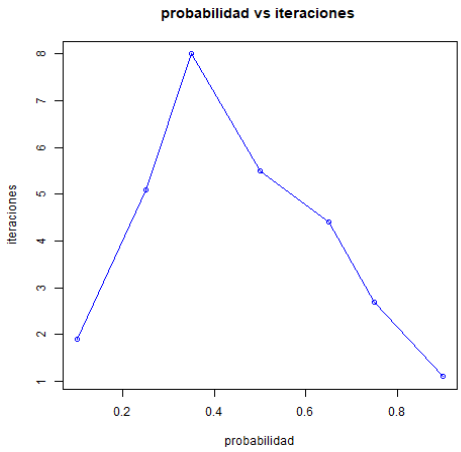


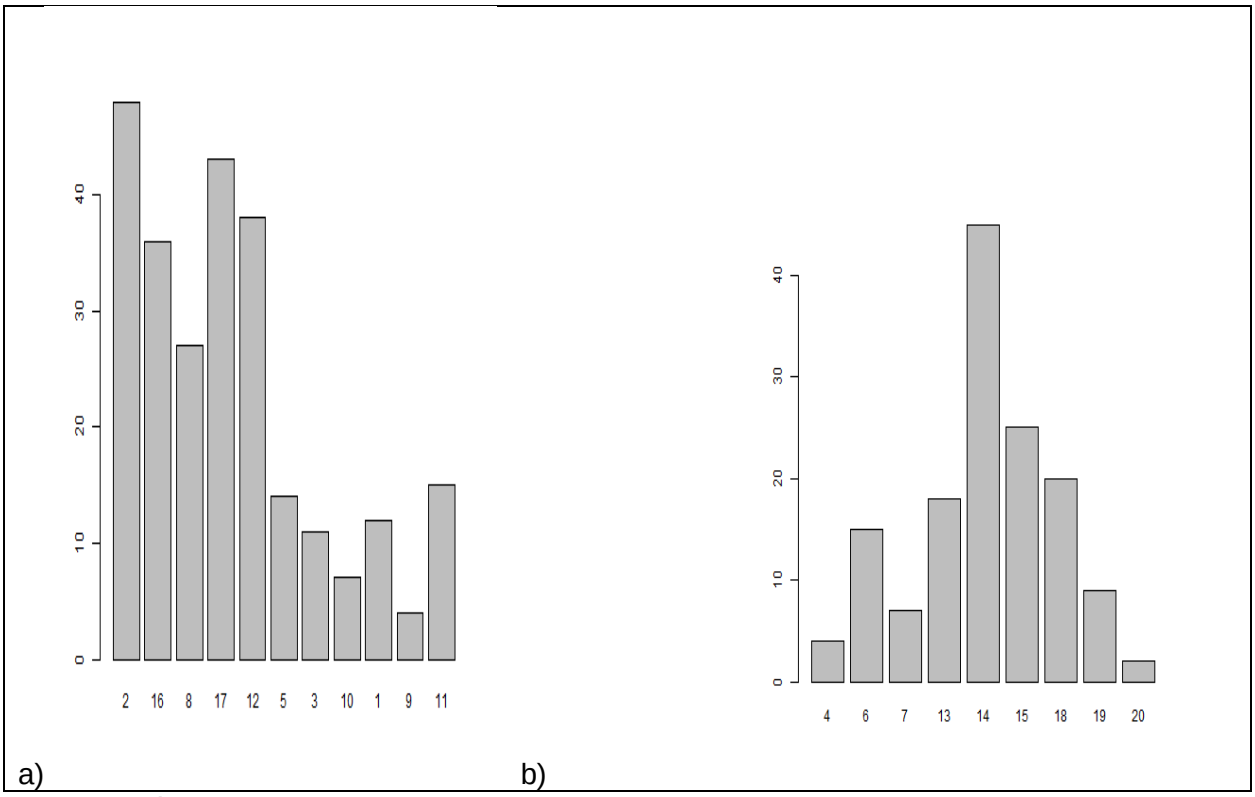
Figura 2 y 3

En las figuras 2 y 3 se observa que el máximo de iteraciones se alcanza cuando la probabilidad es de ~ 0.38 esto indica que cuando la probabilidad es baja, el número de iteraciones es muy bajo, recíprocamente, si la probabilidad es alta, el número de iteraciones también es bajo.

2) Distribución de los autómatas que llegan al borde o no

El código se modifica ya que desde el inicio se determina la cantidad de núcleos y se le asigna una probabilidad de 0.4 de crecimiento, esto le permite elegir de la vecindad, alguna celda vacía y con la probabilidad antes mencionada, invadirla. También se quiere analizar el tamaño de aquellos nácelos que no llegan a tocar el borde al final de la ultima iteración, para este experimento continuamos con una matriz de 30 y con 30 núcleos iniciales.

Resultados



a) b)

Figura 3) Núcleos que tocan el borde(a) y no tocan el borde(b)

Tabla 1

La tabla 1 muestra una descripción a las gráficas anteriores.

Núcleos que tocan el borde	Núcleos que no tocan el borde
Son más los núcleos	Son menos núcleos
Mayor tamaño	Menor tamaño

Se creo un archivo GIF con el programa GIFANIMATOR (el cual se puede encontrar en la carpeta llamada practica 2), en el cual se puede observar cada iteración generada a partir de los núcleos iniciales.