

Programación II  
Curso 2020-2021

*Entrega Final*

## “Game of Life”

---

Daniel Aramburu, Jorge Salas, Marc Pastor

NIA's: 100430670, 100430639, 100430509

Madrid, 4 de diciembre de 2020

# Índice

<b>1. Estructuras de datos utilizadas .....</b>	<b>3</b>
<b>2. Estructuras de programación utilizadas .....</b>	<b>3</b>
<b>3. Funciones .....</b>	<b>4</b>
<b>4. Manual de usuario.....</b>	<b>5</b>

## 1. Estructuras de datos utilizadas

Las principales estructuras de datos que hemos utilizado son:

1. **Vectores:** hemos utilizado vectores principalmente para la entrada por teclado de los datos por parte del usuario. Mediante el uso *readline()* obteníamos un vector de *characters*, que posteriormente convertíamos a vector numérico, en caso que *grepl()* detectara dígitos enteros (aunque en formato carácter). También hemos usado muchos vectores de carácter lógico (*boolean*), para crear bucles *while* y condiciones que aseguran el buen funcionamiento del programa y detectan los errores del usuario al insertar los valores que se le piden.
2. **Matrices:** Hemos utilizado matrices para la creación de los tableros (tanto el inicial como los sucesivos). El motivo de su uso ha sido que el juego solo trata con un tipo de valores (en nuestro caso caracteres), y las matrices son la estructura óptima cuando se trata de datos que comparten clase (y tienen 2 dimensiones), ya que nos permite extender las propiedades de los vectores a 2 dimensiones. En el caso que tuviéramos que introducir tipos de datos distintos (por ejemplo números y caracteres), habríamos optado por el *data.frame()*, pero al no darse el caso, hemos creído que la matriz era más adecuada. En nuestro código hay 2 matrices: *grid\_gen\_0* (tablero 0) y *grid\_gen\_1* (tablero 1), sus dimensiones son las mismas y la especifica el usuario.

## 2. Estructuras de programación utilizadas

Las principales estructuras de programación que hemos usado son:

1. **Bucles:** Hemos utilizado bucles (principalmente *whiles*) para volver a pedir por teclado los datos al usuario, en caso de que los valores que ha introducido sean incorrectos y también para seguir en el menú de opciones, mientras el usuario no seleccione la opción de “Terminar”. El motivo es que no conocemos el número de veces que el usuario se equivocará al introducir los datos, o el número de células vivas que quiere añadir o eliminar. También hemos usado bucles *for()* principalmente para localizar a una célula y a sus vecinas (iterando por filas y columnas el tablero).
2. **Condicionales:** Esta ha sido la estructura de programación que más hemos usado. La hemos empleado para decidir si el programa debe continuar con su ejecución (cuando los datos introducidos son correctos) o meterse dentro de un bucle hasta que el usuario los introduzca correctamente, así como para determinar si una célula es vecina de otra, o si está viva o muerta.
3. **Funciones:** hemos utilizado muchas funciones base de R, para facilitarnos el trabajo como: *strsplit()*, *readline()*, *grepl()*, *as.integer()*, *nrow()*, *ncol()*, etc. Además hemos creado 2 funciones propias, en las que se basa el funcionamiento del programa.

### 3. Funciones

Hemos creado cuatro funciones para diseñar nuestro programa:

1. ***game\_set\_up()*** : Esta función es la que sustenta el programa. Nos permite pedir los datos necesarios al usuario, establecer las condiciones de cómo deben ser dichos datos (y en caso de que no sean correctos avisar al jugador), para posteriormente crear el tablero inicial y añadir/eliminar células vivas. Esta función contiene muchas variables, que principalmente se dividen en vectores de tipo *character* , mediante los que guardamos los datos que introduce el usuario, vectores lógicos para aplicar bucles y condiciones y matrices para crear las tablas. No depende de parámetros, sino que usa datos introducidos por teclado y al final de su ejecución nos devuelve el tablero inicial que hemos creado. No depende de ningún parámetro, aunque pide continuamente datos por teclado al usuario.
2. ***Presentar\_tablero()***: es la función usada para mostrar por pantalla las matrices usadas de tablero de juego que, en esencia, se basa en una llamada a la función *View()* que viene ya integrada en R. Esta función depende de un parámetro que es la tabla que se quiera presentar.
3. ***crear\_tabla\_n()*** : Esta función coge como input el tablero inicial generado por *game\_set\_up()*, al cual le aplica las reglas de reproducción, soledad, supervivencia y sobrepoblación, para luego devolver como output el tablero de la generación 1 en una ventana nueva. Al completar esta tarea guarda la nueva tabla que correspondería a la generación 1 y vuelve a aplicar las reglas anteriores para calcular la segunda, y así hasta que el usuario por teclado decida terminar el programa. Esta función depende de un parámetro *vecindario*, que le indica si el usuario ha escogido jugar en un vecindario normal o extendido.
4. ***game\_of\_life()***: La finalidad de esta función simplemente es recoger el programa completo en una sola función para que sea más cómodo su uso. Dentro de ella están todas las funciones anteriores que hacen que el programa funcione. No depende de ningún parámetro, simplemente ejecuta las anteriores funciones una detrás de otra.

## 4. Manual de usuario

*Game of Life* es un juego para 0 jugadores en el que a partir de unas reglas básicas lleva un conjunto de células dentro de un tablero de una generación a otra.

**1. Creación de la generación inicial:** Al principio le pedirá que introduzca por teclado las filas y columnas del tablero, es decir, la matriz en la que se va a ejecutar el juego. Es importante que siga todas las instrucciones de los mensajes de la consola al pie de la letra. Si introduce un valor que no sea un número entero positivo saltará un error, no se preocupe, todos cometemos errores, sólo inténtelo otra vez. **Estos mensajes de error aparecerán durante todo el proceso de creación así que si los ve revise los datos introducidos en la última inserción.**

```
Inserte el numero de filas del tablero por favor w  
[1] "El valor a introducir debe ser entero positivo"  
Vuelva a insertar el numero de filas del tablero por favor @  
[1] "El valor a introducir debe ser entero positivo"  
Vuelva a insertar el numero de filas del tablero por favor |
```

Una vez introducida una dimensión válida le preguntará si quiere introducir las células vivas manualmente, pudiendo elegir dónde colocar cada una de ellas, automáticamente, que generará células vivas repartidas aleatoriamente por el tablero, o gráficamente.

```
¿Como desea introducir las células vivas?:  
Escriba 1 para MANUAL  
Escriba 2 para AUTOMÁTICO  
Escriba 3 para GRÁFICAMENTE
```

En este momento solo tiene que introducir el número 1, 2 o 3.

Si selecciona la opción de hacerlo manualmente, le llevará a otra decisión:

```
Qué deseas hacer?  
1: Añadir célula viva  
2: Eliminar célula viva  
3: Terminar
```

1. Añadir Célula: crea una célula viva en la posición especificada. Si en las coordenadas que está intentando introducir una célula había otra anteriormente, el programa le avisará y volverá a la decisión anterior con todo el tablero invariado. El modo de introducir las coordenadas es x,y donde x e y son números naturales.
2. Eliminar célula: Convierte una célula viva en célula muerta o vacía. Si no hay células vivas en las coordenadas especificadas ocurrirá lo mismo que en el caso anterior.
3. Terminar: Como su propio nombre indica, termina el proceso de introducción de células del tablero. Una vez seleccione esta opción el programa usará las reglas para generar el tablero de la generación 1.

En el caso que se escoja la opción 1 o 2 el usuario deberá especificar las coordenadas de la célula que quiera añadir o eliminar.

**Inserte las 2 coordenadas separadas por una coma**  
**IMPORTANTE: ¡LAS COORDENADAS DEBEN SER NÚMEROS ENTEROS!**

En el caso que quiera añadir una célula que ya está viva, el programa la dejará viva, pero dará un aviso al usuario. Y en el caso que quiera eliminar una célula que ya está muerta también le avisará:

**Inserte las 2 coordenadas separadas por una coma**  
**IMPORTANTE: ¡LAS COORDENADAS DEBEN SER NÚMEROS ENTEROS!** 1,1  
 [1] "Esta célula ya está viva"

**Inserte las 2 coordenadas separadas por una coma**  
**IMPORTANTE: ¡LAS COORDENADAS DEBEN SER NÚMEROS ENTEROS!** 1,2  
 [1] "Esta célula ya está muerta"

Cuando haga una modificación en el tablero, se le mostrará en una ventana dentro de *Rstudio* simultáneamente (mediante *View()*).

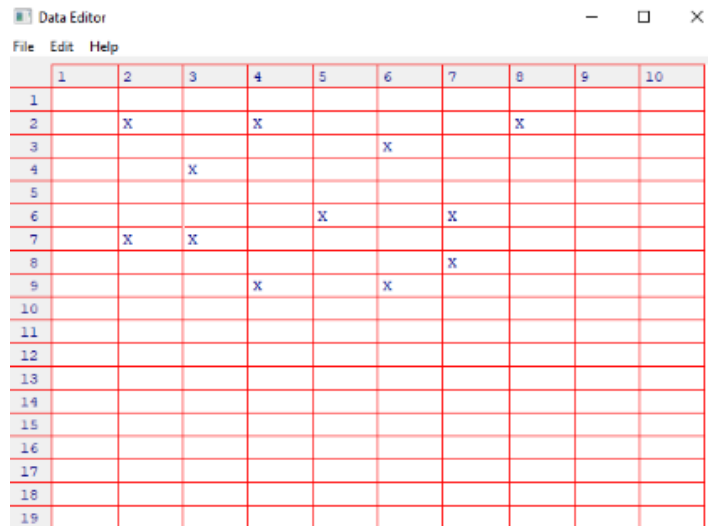
Al usar la opción automática simplemente le pedirá la cantidad de células vivas que quiere en el tablero

**Ahora, seleccione el número de células vivas que quiere introducir por favor |**

Si en este punto introduce un valor inválido (algo que no sea un número o un número de células superior al de la dimensión de la matriz) le avisará del error y le volverá a pedir que introduzca el número de células vivas

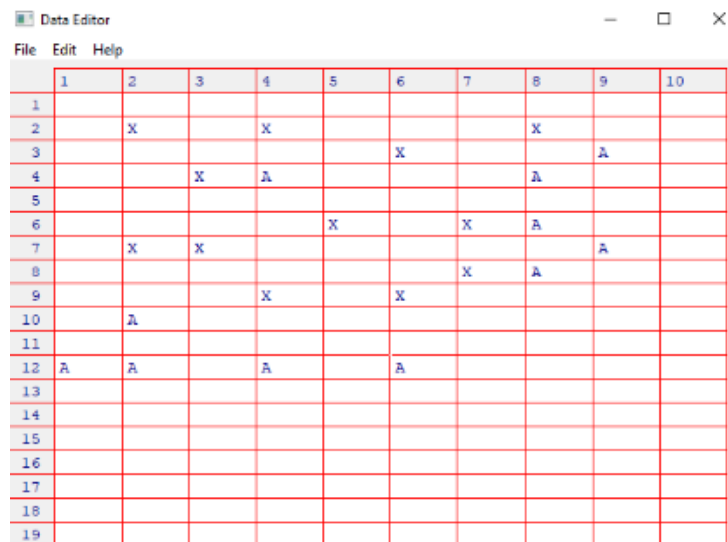
```
Ahora, seleccione el número de células vivas que quiere introducir por favor hdj
[1] "Debe introducir un número entero de células vivas, y este tiene que ser menor a 100 (el número de celdas del tablero)"
volvamos a intentarlo:
seleccione el número de células vivas que quiere introducir por favor
```

Si usa la tercera opción para introducir las células vivas, la opción gráfica, se le abrirá una tabla mediante la función *edit()* de R para que introduzca manualmente el carácter “X” en las casillas en las que quiera que haya una célula viva.



	1	2	3	4	5	6	7	8	9	10
1										
2		X		X				X		
3						X				
4			X							
5										
6					X		X			
7		X	X							
8							X			
9				X		X				
10										
11										
12										
13										
14										
15										
16										
17										
18										
19										

Si introduce un carácter distinto de “X” o introduce alguna célula fuera de la dimensión, como en la imagen siguiente: (A no es un valor válido y además en la fila 12 hay valores, cuando hemos definido un tablero de 10 x 10)

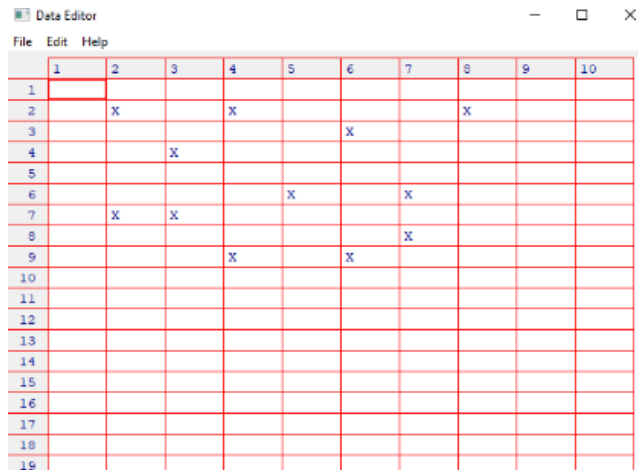


	1	2	3	4	5	6	7	8	9	10
1										
2		X		X				X		
3						X			A	
4			X	A				A		
5										
6					X		X	A		
7		X	X						A	
8							X	A		
9				X		X				
10		A								
11										
12	A	A		A		A				
13										
14										
15										
16										
17										
18										
19										

El programa le avisará por consola y le pedirá que vuelva a introducir los datos, borrando las entradas incorrectas automáticamente y mostrando por pantalla el tablero corregido.

```
[1] "Ha introducido caracteres incorrectos. "
```

```
[1] "Introduzca X en las casillas vivas y no introduzca nada en las demás."
```

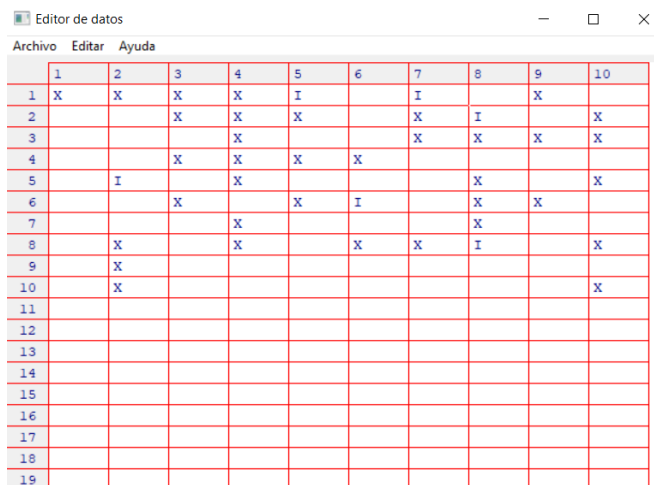


	1	2	3	4	5	6	7	8	9	10
1										
2		X		X				X		
3						X				
4			X							
5					X					
6							X			
7		X	X							
8							X			
9				X		X				
10										
11										
12										
13										
14										
15										
16										
17										
18										
19										

Al terminar de confeccionar la primera generación el programa le preguntará si quiere usar la variante en la que existen casillas inhabitables. Una casilla inhabitable será aquella que permanece “muerta” o “inactiva” durante todo el juego. Para elegir se usa el mismo sistema que en todas las demás elecciones:

```
¿Quieres tener casillas inhabitables?
1- Si
2- No
```

Si elige jugar con las casillas inhabitables se le abrirá la función un tablero mediante *edit* como si quisieras introducir las células vivas gráficamente pero esta vez debe introducir el carácter “I” para hacer la casilla inhabitable.



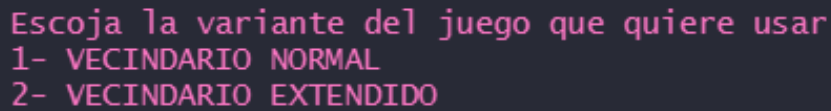
	1	2	3	4	5	6	7	8	9	10
1	X				I		I		X	
2		X	X	X	X			I		X
3			X	X			X	X	X	X
4			X	X	X	X				
5		I	X	X				X		X
6			X		X	I		X	X	
7				X				X		
8		X		X		X	X	I		X
9		X								
10		X								X
11										
12										
13										
14										
15										
16										
17										
18										
19										

(Véanse las casillas 1,5; 1,7 o 5,2)



Si introduce un carácter distinto de “I” le dará un mensaje de error y le pedirá que vuelva a intentarlo.

El siguiente paso en la configuración inicial del juego es decidir con qué vecindario quiere jugar, el normal o el extendido.



```
Escoja la variante del juego que quiere usar
1- VECINDARIO NORMAL
2- VECINDARIO EXTENDIDO
```

En caso que elija 2, el juego se ejecutará normalmente pero el vecindario de cada célula (la zona del tablero que usa para calcular las reglas), en lugar de ser un círculo de 3x3 alrededor de la célula se convertirá en uno de 5x5

**2. Guía para las reglas del juego:** En este apartado desarrollaremos las reglas mencionadas al principio de este manual.

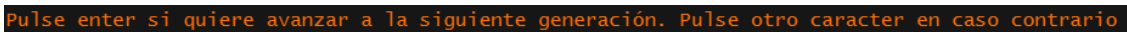
**1. Regla de Reproducción:** Si una célula muerta tiene exactamente 3 células vecinas en su generación (células vivas alrededor de la célula en cuestión), en la siguiente renacerá.

**2. Regla de Supervivencia:** Si una célula viva tiene 2 o 3 células vecinas en su generación, se mantendrá con vida en la siguiente.

**3. Regla de Soledad:** Si una célula viva tiene 1 o ninguna vecina en su generación, desaparecerá en la siguiente

**4. Regla de Superpoblación:** Si una célula viva tiene 4 o más vecinas en su generación, morirá en la siguiente.

Una vez esté creado el tablero se le mostrará en la consola el siguiente mensaje



```
Pulse enter si quiere avanzar a la siguiente generación. Pulse otro caracter en caso contrario
```

Si decide seguir avanzando a la derecha en la zona de *Plots* se le indicará la generación en la que está y cuando el juego termine se lo indicará también debajo en rojo. Recomendamos ampliar el área de *plots*, para poder visualizar mejor el contador de la generación.

## Generación 8

Juego Finalizado

**¡Esperamos que disfrute nuestra versión del Game of Life de John Conway!**