14-6-2019

El juego de la vida

Un estudio de simulación

ELENA GARRIDO JIMÉNEZ
TÉCNICAS DE SIMULACIÓN

Construye una función que simule el juego de la vida, a la que le pases (al menos) los argumentos: side, n (o pbinom) y steps y te devuelva o un gif con todas las iteraciones, o una colección de tableros o la solución de visualización que hayas utilizado.

El código creado es el siguiente:

```
library(caTools)

# Pedir al usuario lados del grid, la probabilidad de encontrar una ca
silla llena e iteraciones
lados <- 0
while (lados <= 1) {
  lados <- as.integer(readline(prompt = "Inserte el numero de lados qu
e conforma el grid. Debe ser mayor a 1: "))
}
p <- -1
while (p < 0 | p > 1) {
  p <- as.double(readline(prompt = "Inserte la probabilidad de que una
celda este viva. Debe estar comprendida entre 0 y 1: "))
}
iteraciones <- 0
while (iteraciones <= 0) {
  iteraciones  <- as.integer(readline(prompt = "Inserte el numero de i
teraciones que desee realizar. Ha de ser al menos 1: "))
}
delay <- -1
while (delay <= 0) {
 delay  <- as.integer(readline(prompt = "Inserte el delay que desea. S
e recomienda entre 5 y 20: "))
}

# Se pasan los datos a la funcion

funcion_juego_vida <- function(lados, p, iteraciones) {
  #   Se crea la martiz con los datos de "lados del grid"
  matriz_datos <- matrix(nrow = lados, ncol = lados)
  #   Se introducen las n primeras celulas vivas
  matriz_datos[,] <- rbinom(lados**2, 1, p)
  #   Se guarda la matriz para el gif
  lista_matrices <- array(0, dim=c(lados, lados, iteraciones+1))
  lista_matrices[,,1] <- matriz_datos
  contador <- 0
  copia_matriz <- matriz_datos
  celdas_vivas <- 0
  #   Mientras contador hechas sea menor o igual al numero de iteracio
nes deseadas:
  while (contador < iteraciones){
    # Para cada valor de fila:
    for (i in c(1:lados)){
      # Para cada valor de columna:
      for (j in c(1:lados)){
        celdas_vivas <- celdas_vivas + copia_matriz[indiceseguro(i-1,l
ados),indiceseguro(j-1,lados)]
```

```
        celdas_vivas <- celdas_vivas + copia_matriz[indiceseguro(i-1,l
ados),j]
        celdas_vivas <- celdas_vivas + copia_matriz[indiceseguro(i-1,l
ados),indiceseguro(j+1,lados)]
        celdas_vivas <- celdas_vivas + copia_matriz[i,indiceseguro(j-1
,lados)]
        celdas_vivas <- celdas_vivas + copia_matriz[i,indiceseguro(j+1
,lados)]
        celdas_vivas <- celdas_vivas + copia_matriz[indiceseguro(i+1,l
ados),indiceseguro(j-1,lados)]
        celdas_vivas <- celdas_vivas + copia_matriz[indiceseguro(i+1,l
ados),j]
        celdas_vivas <- celdas_vivas + copia_matriz[indiceseguro(i+1,l
ados),indiceseguro(j+1,lados)]
        matriz_datos[i,j] <- vida_muerte(copia_matriz[i,j], celdas_viv
as)
        celdas_vivas <- 0
        }
    }
    contador <- contador + 1
    lista_matrices[,,contador+1] <- matriz_datos
    copia_matriz <- matriz_datos
  }
  return(lista_matrices)
}

# Funcion para calcular indice seguro y que no de problemas de lectura
:

indiceseguro <- function(j, lados){
  if (j < 1){
    return(lados)
  }
  else if (j > lados){
    return(1)
  }
  else {
    return(j)
  }
}

# Funcion para ver si una celda muere o vive
vida_muerte <- function(celda, celdas_vivas){
  valor <- celda
  if (celda == 1){
    if(celdas_vivas < 2){
      valor <- 0
    }
    else if(celdas_vivas > 3){
      valor <- 0
    }
  }
  else {
    if(celdas_vivas == 3){
```

```
      valor <- 1
    }
  }
  return(valor)
}
# Funcion para crear gifs
crear_gif <- function(lista_matrices, filename, delay){
  return(write.gif(lista_matrices, filename=filename, delay=delay, col
= "rainbow", scale = "smart"))
}
```

En primer lugar, podemos destacar cuatro funciones diferentes: una de ellas se encarga de realizar, en base a los inputs dados, algunos de los cálculos necesarios para crear las matrices de ceros y unos y simular el juego de la vida; la segunda de las funciones se encarga de solucionar el conocido problema de los bordes; la tercera función se encarga de determinar la vida o la muerte de una celda; y la cuarta es capaz de crear un gif en base a los datos proporcionados. La segunda y la tercera función proporcionan a la primera algunos de los outputs que ésta necesita.

El mundo ha sido creado de modo que éste es "abierto", es decir: para solucionar el problema de los bordes se ha optado porque éstos lean los de la correspondiente parte simétrica de la matriz (e.g. para calcular los valores circundantes a la celda (1,5), se calculan los valores (1,4), (1,6), (2,4), (2,5), (2,6) y (lados, 4), (lados,5) y (lados, 6)).

El código no contiene tildes debidos a errores posteriores en el entorno de desarrollo una vez se vuelve a abrir el archivo.

Estabiliza el juego hasta que aparezcan los patrones esperables: osciladores, matusalenes, etc

La estabilización del juego varía en función de la probabilidad inicial de que una celda esté o no inicialmente viva, como podremos ver a continuación. Es por ello que se ha simulado el efecto de la estabilización del juego (entendido como que se llegue a alcanzar durante muchas simulaciones un número constante de células vivas) en función de la probabilidad inicial de que una célula se halle viva.

El código propuesto para la realización del ejercicio es el siguiente:

```
#Ejercicio 2
celdas_vivas <- c()
for (i in seq(0.1, 0.4, by=0.1)){
  filename <- sprintf("Juego de la vida%1.1f.gif", i)
  crear_gif(funcion_juego_vida(30,i,1000), filename, 50)
  celdas_vivas <-c()
  matrices <- replicate(100, funcion_juego_vida(30,i,1000))
  for (k in c(1:1000)){
    celdas_vivas<- c(celdas_vivas, sum(matrices[,,k,])/100)
  }
  print(celdas_vivas)
  plot(celdas_vivas, type="l", main = sprintf("Celdas vivas en funcion
de las iteraciones para una probabilidad de %1.2f", i))
  celdas_vivas <-c()
```

```
  a <- c()
}
```

Para realizar esta tarea, se han simulado un total de 100 simulaciones, contando cada una de ellas con 1000 iteraciones del algoritmo, del cual extraemos el número medio de células vivas para las 100 matrices replicadas en esa iteración determinada. En este caso, se ha simulado una matriz de 30 x 30, y se han representado los valores medios de células vivas para una probabilidad inicial de vida de 0.1, 0.2, 0.3 y 0.4. El motivo de elegir estas cifras radica que, en base al ejercicio realizado posteriormente en el apartado 3, se estima que el número máximo de células vivas para esta probabilidad ronda alrededor de esos valores para una matriz de 30 x 30.

Los resultados obtenidos se presentan a continuación. En primer lugar, se han añadido algunos gifs obtenidos a partir de este código, para comprobar que dan como resultado algunos de los patrones requeridos en este ejercicio: en especial, hablamos de vidas estáticas (sobre todo colmenas y bloques, aunque se pueden llegar a vislumbrar a veces "loafs"), planeadores (un tipo de nave espacial) y osciladores (especialmente, "blinkers"). Para poder vislumbrar los archivos .gif, se recomienda abrirlos con el explorador y ampliar al máximo la vista.

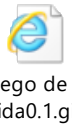


Figura 1. Gif realizado con una probabilidad de aparición inicial de una célula viva de 0.1

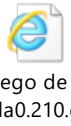


Figura 2. Gif realizado con una probabilidad de aparición inicial de una célula viva de 0.2

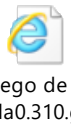


Figura 3. Gif realizado con una probabilidad de aparición inicial de una célula viva de 0.3

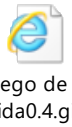
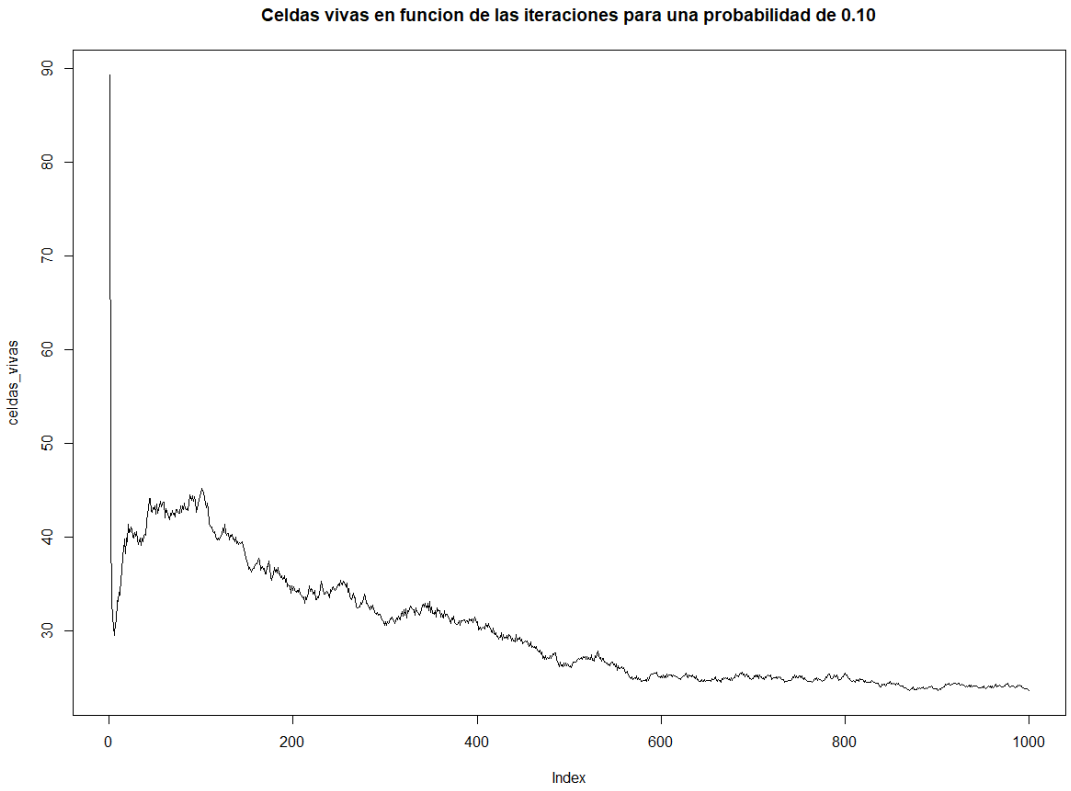
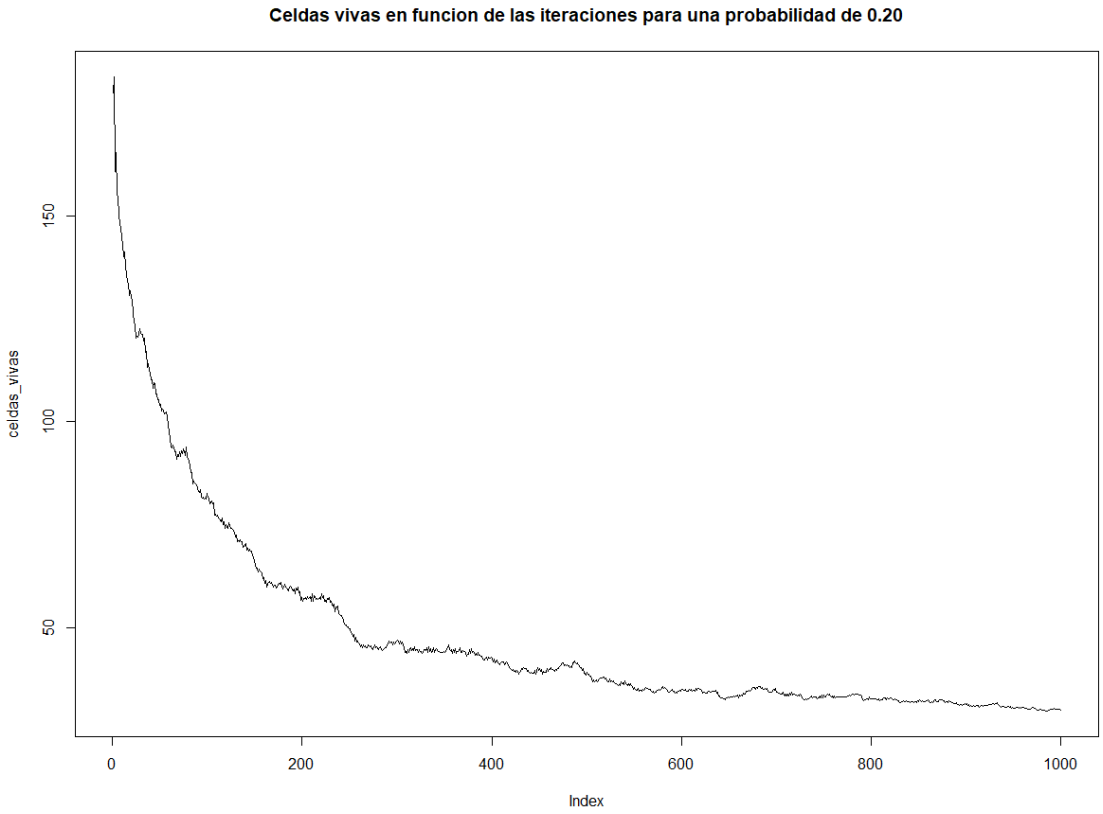
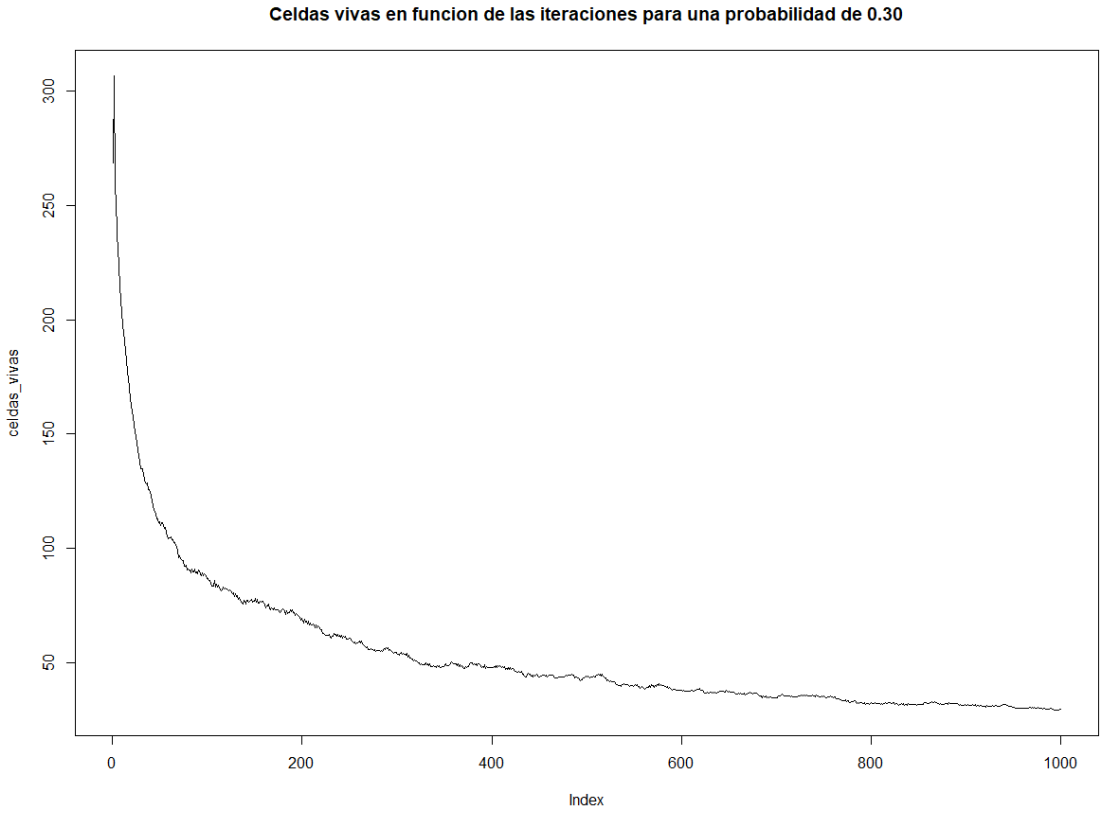


Figura 4. Gif realizado con una probabilidad de aparición inicial de una célula viva de 0.4

Una vez estudiados los diferentes gifs, se presentan a continuación algunas gráficas que reflejan la evolución en el número de celdas medias vivas.


Celdas vivas en funcion de las iteraciones para una probabilidad de 0.10
celdas_vivas
30
40
50
60
70
80
90
0
200
400
600
800
1000
Index


Celdas vivas en funcion de las iteraciones para una probabilidad de 0.20
celdas_vivas
50
100
150
0
200
400
600
800
1000
Index


Celdas vivas en funcion de las iteraciones para una probabilidad de 0.30
celdas_vivas
50
100
150
200
250
300
0
200
400
600
800
1000
Index

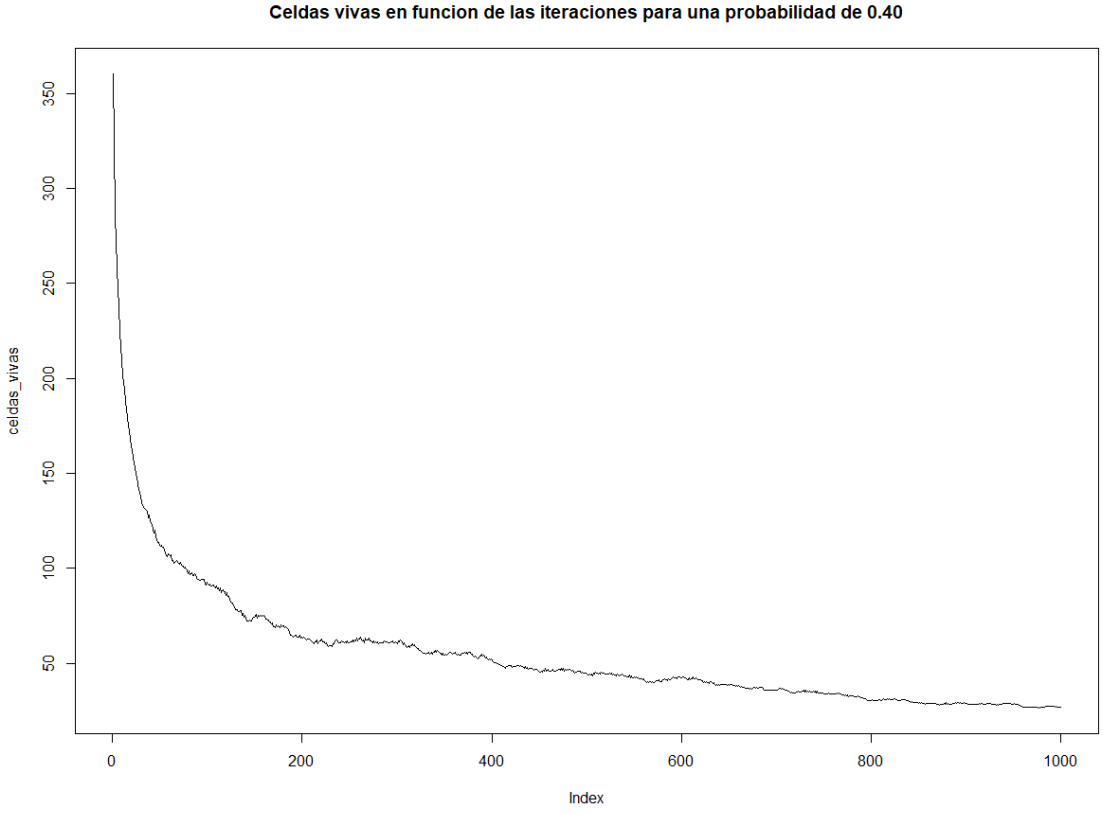

Celdas vivas en funcion de las iteraciones para una probabilidad de 0.40
celdas_vivas
50
100
150
200
250
300
350
0
200
400
600
800
1000
Index

Tabla 1. Número medio de células vivas en función de las iteraciones y de la probabilidad de aparición inicial de una celda viva

Algunas de las conclusiones que se desprenden al mirar la tabla anterior y las distintas gráficas son las siguientes: en primer lugar, podemos observar que se sigue un decrecimiento típico de las funciones racionales (aquéllas de la forma $\frac{1}{x}$). Otro de los aspectos relevantes a señalar es el siguiente: tal y como se puede comprobar, las diferencias iniciales en cuanto al número medio de células vivas varían enormemente en función de la probabilidad. Estas diferencias se van haciendo cada vez más pequeñas a medida que aumenta el número de iteraciones, hasta que a partir de la iteración 987, aunque las diferencias son pequeñas, se observa en la tabla que a menor probabilidad inicial de celdas vivas al inicio, mayor es el número de celdas que consiguen vivir (a excepción de una probabilidad de aparición de una celda viva al inicio de 0.1).

Aun así, podemos concluir que la probabilidad inicial tiene un efecto cada vez menor en el número de celdas vivas a medida que las iteraciones aumentan, de modo que las reglas aquí aplicadas y descritas inicialmente por Conway para la condición de células muertas o vivas "diluyen" los efectos de la probabilidad a medida que se aumenta el número de iteraciones del algoritmo.

Analiza la probabilidad al iniciar autómatas con la cantidad de vida después de 50 iteraciones

Para realizar este apartado, se optó por informar acerca de la media de células vivas en 100 repeticiones del algoritmo variando el tamaño de los lados que conforman la matriz cuadrada. De esta manera, podemos observar las gráficas obtenidas, que muestran el número de celdas vivas en función de la probabilidad de que aparezcan o no vivas, además de las tablas donde se recogen estos valores, donde se varía la longitud de los lados de la matriz.

El código empleado para realizar este ejercicio es el siguiente:

```
# Ejercicio 3
# Para realizarlo, se ha hecho una modificación de la funcion por la c
ual la misma devuelve en la iteracion nº 50 el numero de celdas vivas

celdas_vivas <- c()
probabilidades <- seq(0,1,by=0.01)
for(k in probabilidades) {
  celdas_vivas<- c(celdas_vivas, mean(replicate(100,sum(funcion_juego_
vida(100,k,50)))))
}
plot(celdas_vivas, xlab='Probabilidad', ylab="Media de celulas vivas",
type="l", xaxt = "n")
axis(1, at=c(1:101), labels = probabilidades)
dataset <- as.data.frame(probabilidades)
dataset[2] <- celdas_vivas
print(dataset)
```

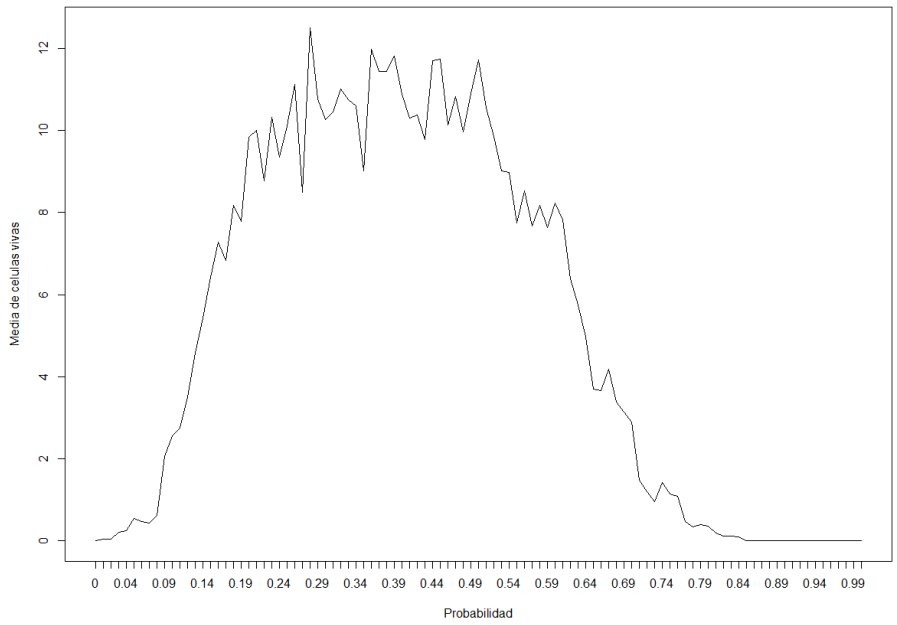


Figura 5. Número de células vivas en función de la probabilidad inicial de que una celda contenga o no vida para una matriz de 10 x 10 tras 50 iteraciones

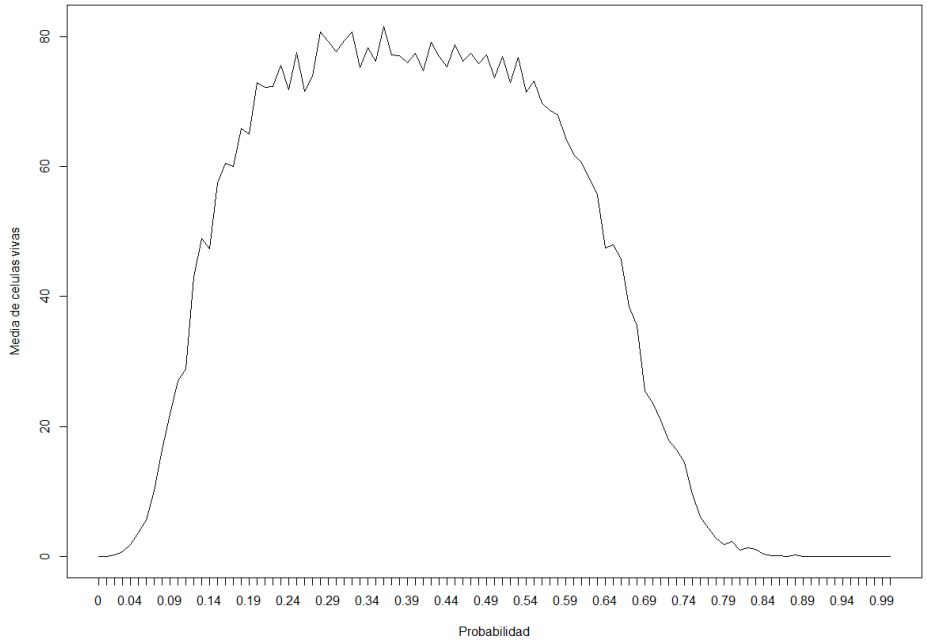


Figura 6. Número de células vivas en función de la probabilidad inicial de que una celda contenga o no vida para una matriz de 25 x 25 tras 50 iteraciones

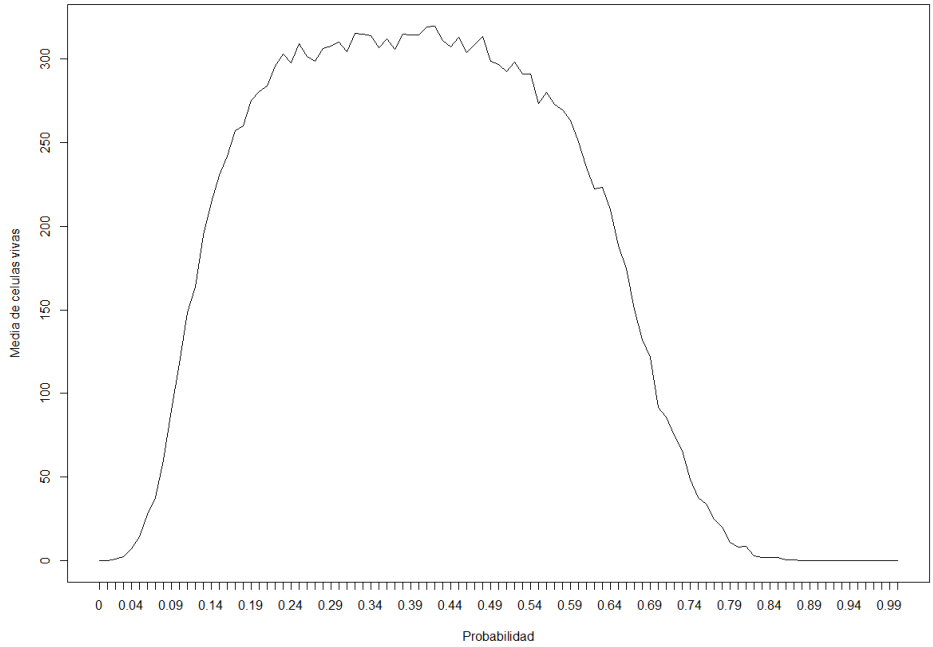


Figura 7. Número de células vivas en función de la probabilidad inicial de que una celda contenga o no vida para una matriz de 50 x 50 tras 50 iteraciones

Tabla 2. Media en 100 repeticiones del número de celdas vivas en función de la probabilidad inicial de estar vivas y del número de lados en 50 iteraciones del algoritmo. En amarillo se han marcado los valores máximos de celdas vivas alcanzados.

Probabilidades	**Número de lados**		
	10	**25**	**50**
0	0	0	0
0.01	0.04	0.03	0.11
0.02	0.03	0.17	0.73
0.03	0.2	0.68	2.23
0.04	0.23	1.75	7.07
0.05	0.53	3.65	14.42
0.06	0.46	5.52	27.57
0.07	0.43	10.1	37.72
0.08	0.62	16.36	60.09
0.09	2.05	21.92	90.33
0.1	2.56	26.96	117.87
0.11	2.75	28.86	148.45
0.12	3.5	43.01	163.59
0.13	4.58	48.88	194.92
0.14	5.43	47.41	214.2
0.15	6.42	57.47	230.68
0.16	7.28	60.5	242.21
0.17	6.83	60.01	257.14
0.18	8.17	65.89	259.93

0.19	7.78	64.96	275.49
0.2	9.84	72.93	280.72
0.21	10	72.22	284.14
0.22	8.77	72.3	296.11
0.23	10.32	75.59	303.08
0.24	9.35	71.77	297.83
0.25	10.12	77.6	309.53
0.26	11.12	71.53	301.87
0.27	8.5	73.86	298.67
0.28	12.51	80.69	306.24
0.29	10.78	79.28	307.99
0.3	10.26	77.73	310.45
0.31	10.45	79.32	304.37
0.32	11.02	80.75	315.59
0.33	10.75	75.26	314.86
0.34	10.6	78.23	314.06
0.35	9.01	76.22	307.14
0.36	11.99	81.57	311.98
0.37	11.44	77.17	305.75
0.38	11.44	77.02	314.84
0.39	11.82	75.91	314.46
0.4	10.89	77.39	314.49
0.41	10.3	74.78	319.27
0.42	10.39	79.11	319.8
0.43	9.79	76.95	310.98
0.44	11.7	75.32	307.62
0.45	11.75	78.75	313.34
0.46	10.13	76.23	303.87
0.47	10.82	77.42	308.67
0.48	9.97	75.87	313.67
0.49	10.91	77.17	298.57
0.5	11.72	73.67	296.93
0.51	10.55	76.95	292.39
0.52	9.86	72.94	298.25
0.53	9.02	76.8	291.25
0.54	8.98	71.41	291.08
0.55	7.75	73.18	273.32
0.56	8.54	69.81	280.22
0.57	7.67	68.71	272.99
0.58	8.18	67.98	269.74
0.59	7.63	64.42	263.25
0.6	8.23	61.89	250.78
0.61	7.82	60.65	235.35
0.62	6.39	58.19	222.61
0.63	5.74	55.68	223.32
0.64	4.96	47.52	210.19

0.65	3.7	47.9	188.44
0.66	3.65	45.77	174.5
0.67	4.18	38.41	150.23
0.68	3.37	35.52	132.02
0.69	3.13	25.45	121.78
0.7	2.89	23.6	91.72
0.71	1.48	20.97	86.01
0.72	1.19	17.85	74.74
0.73	0.95	16.42	65.56
0.74	1.41	14.44	48.71
0.75	1.14	9.59	37.54
0.76	1.08	6.02	33.89
0.77	0.47	4.4	24.75
0.78	0.34	2.78	20.14
0.79	0.39	1.81	10.94
0.8	0.35	2.33	8.15
0.81	0.19	1	8.41
0.82	0.1	1.26	2.62
0.83	0.1	1.09	1.88
0.84	0.09	0.39	1.93
0.85	0	0.14	1.73
0.86	0	0.04	0.21
0.87	0	0	0.31
0.88	0	0.19	0.13
0.89	0	0	0.06
0.9	0	0	0.04
0.91	0	0	0.08
0.92	0	0	0
0.93	0	0	0
0.94	0	0	0
0.95	0	0	0
0.96	0	0	0
0.97	0	0	0
0.98	0	0	0
0.99	0	0	0
1	0	0	0

Como era de esperar, el tamaño de los lados de la matriz influye en el número medio de celdas vivas que se encuentran en la misma. Además de esa propiedad, podemos extraer una conclusión común a partir de la simulación realizada: el rango donde se alcanza la máxima población de celdas vivas varía entre una probabilidad de aparición de una celda viva entre [0.28, 0.42]. El número medio de celdas vivas crece a medida que lo hace la probabilidad de encontrar una celda viva hasta el punto máximo indicado anteriormente, que varía en función del número de lados. Luego, por las propias reglas que rigen el juego de la vida, decrece hasta ser “0” cuando se ronda una probabilidad entre [0.85, 1].

Analiza si hay algún efecto producido por las dimensiones del tablero o es solo la densidad

Para estudiar el enunciado de la pregunta, proponemos que la forma de estudiar los dos fenómenos sea la siguiente: se realizarán 100 repeticiones junto con 500 iteraciones del algoritmo para cada uno de los dos estudios. Sin embargo, en el primero de los casos, variaremos las dimensiones del tablero y mantendremos constante la probabilidad de aparición de una celda viva al inicio del juego, siendo ésta de 0.3. En comparación, para ver únicamente los efectos de la densidad (dado por la probabilidad de aparición al inicio del juego) en una matriz de 30x30, se tomarán como referencia los datos obtenidos en el apartado anterior de este informe.

El código empleado y los resultados gráficos se presentan a continuación.

```
# Ejercicio 3-4
for(lados in c(10, 25, 50)) {
  matrices <- replicate(100,funcion_juego_vida(lados,0.3,500))
  for(k in c(1:500)){
    celdas_vivas<-c(celdas_vivas, sum(matrices[,,k,]/500))
  }
  plot(celdas_vivas, ylab="Media de celulas vivas", type="l")
  celdas_vivas <- c()
}
```

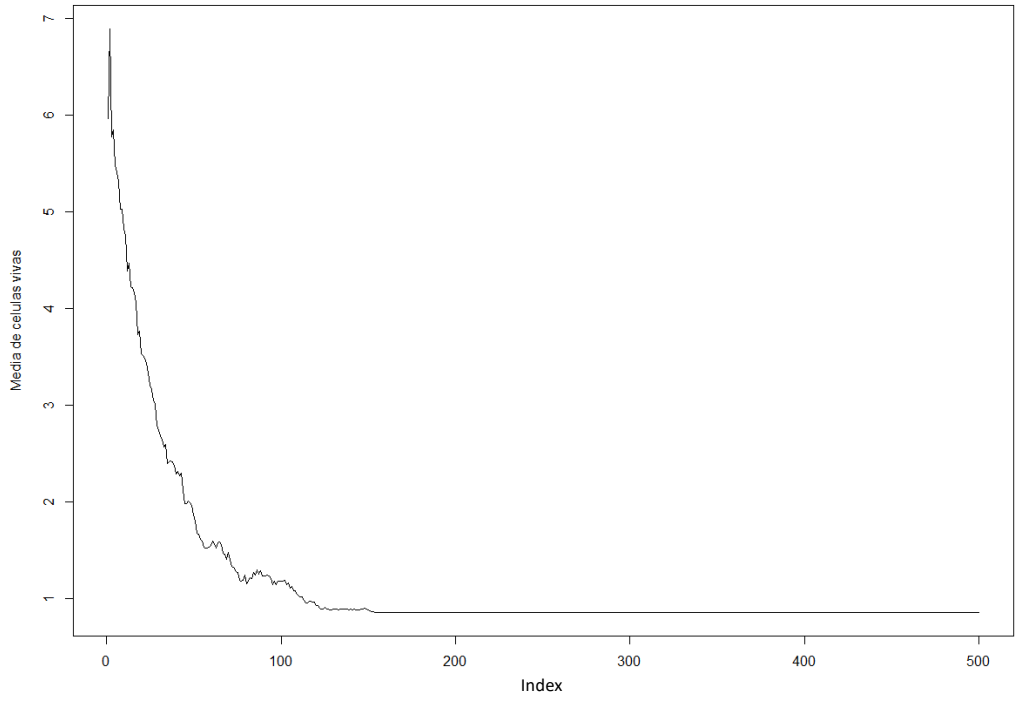


Figura 8. Número medio de células vivas en función de las iteraciones para matriz de tamaño 10x10

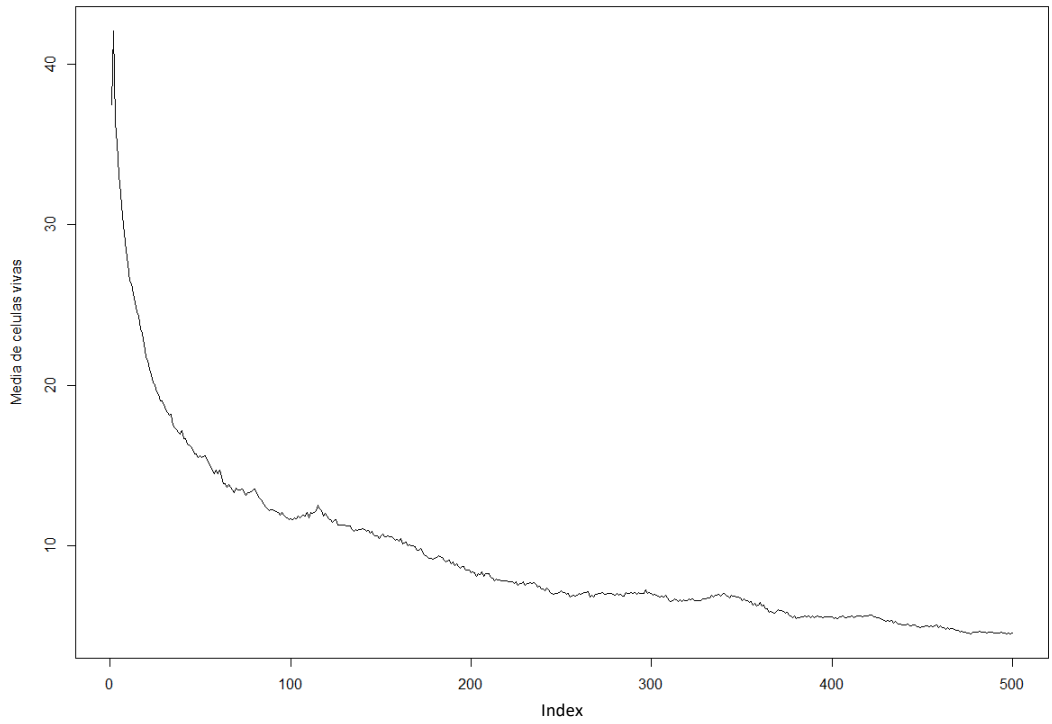


Figura 9. Número medio de células vivas en función de las iteraciones para matriz de tamaño 25x25

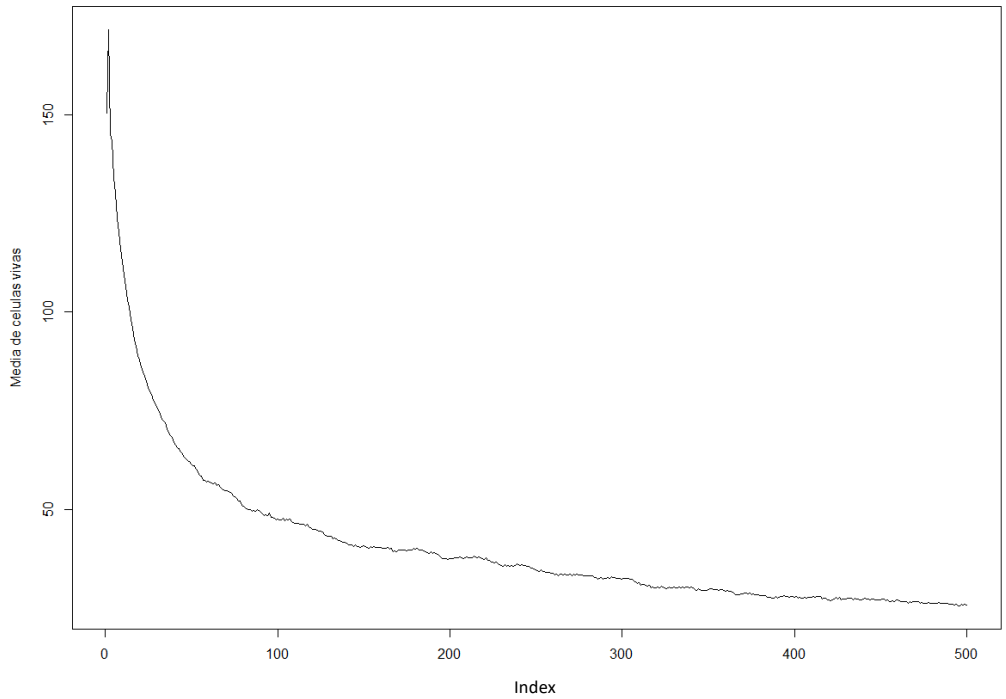


Figura 10. Número medio de células vivas en función de las iteraciones para matriz de tamaño 50x50

En este caso, podemos comprobar que el fenómeno simula el comportamiento de una función de tipo racional, mas podemos comprobar que el efecto de los bordes es, en efecto, importante: al menos hasta el número de iteraciones realizadas, se puede comprobar gráficamente la notoria diferencia en cuanto a la cantidad inicial y final de células vivas en función de cada una de las iteraciones. Aun así, es importante remarcar el importante efecto de las reglas del juego establecidas: no importa de nuevo el número de celdas vivas inicialmente consideradas (logradas aumentando el número de lados o aumentando la probabilidad inicial de encontrar celdas vivas), dado que éstas tienden a regular el número de ellas a medida que se suceden las iteraciones. Esto explica por ejemplo, que las notorias diferencias iniciales de nuevo no vuelven a ser tan importantes a medida que realizamos más iteraciones.

Prueba con reglas diferentes a la (2,3)/3 y comprueba el resultado

A continuación, estudiaremos en este apartado todo lo descrito anteriormente cambiando las reglas para una mejor comparativa.

Las reglas se cambiaban directamente en el código de la función vida_muerte, tal y como aparece en el extracto del código que aparece a continuación.

```
vida_muerte <- function(celda, celdas_vivas){
  valor <- celda
  if (celda == 1){
    if(celdas_vivas < 4){
      valor <- 0
    }
    else if(celdas_vivas > 6){
      valor <- 0
    }
  }
  else {
    if(celdas_vivas == 6){
      valor <- 1
    }
  }
  return(valor)
}
```

```
# Ejercicio 3-4
for(lados in c(10, 25, 50)) {
  matrices <- replicate(100,funcion_juego_vida(lados,0.3,500))
  for(k in c(1:500)){
    celdas_vivas<-c(celdas_vivas, sum(matrices[,,k,]/500))
  }
  plot(celdas_vivas, ylab="Media de celulas vivas", type="l")
  celdas_vivas <- c()
}

for(p in seq(0.1, 0.4, by=0.1)){
  matrices <- replicate(100,funcion_juego_vida(30,p,500))
  for(k in c(1:500)){
    celdas_vivas<-c(celdas_vivas, sum(matrices[,,k,]/500))
```

```
  }
  plot(celdas_vivas, ylab="Media de celulas vivas", type="l")
  celdas_vivas <- c()
}
```

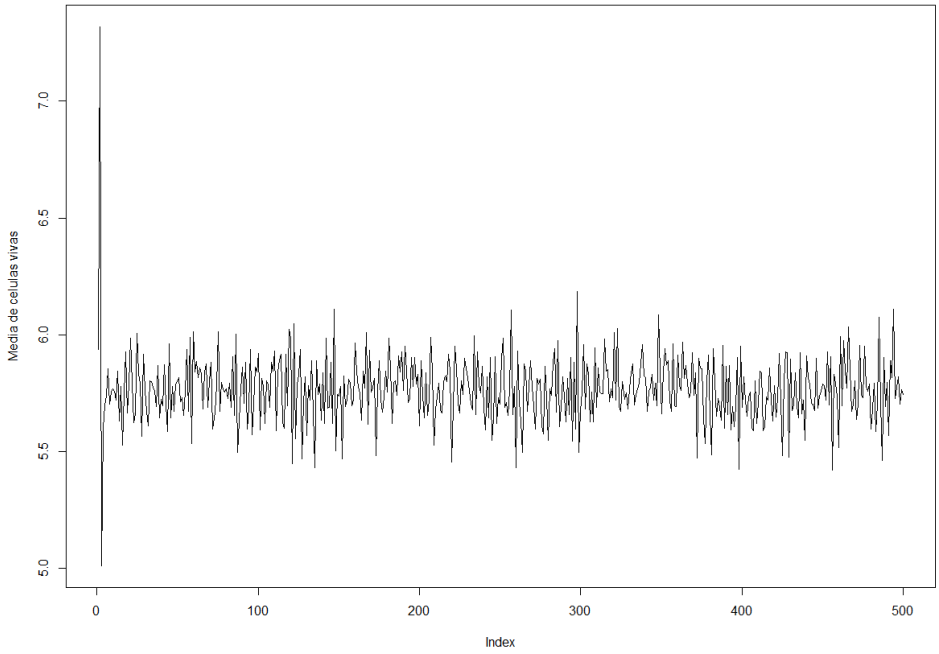


Figura 11. Número medio de células vivas en función al número de iteraciones para las reglas (1,2|2) en una matriz de 10x10

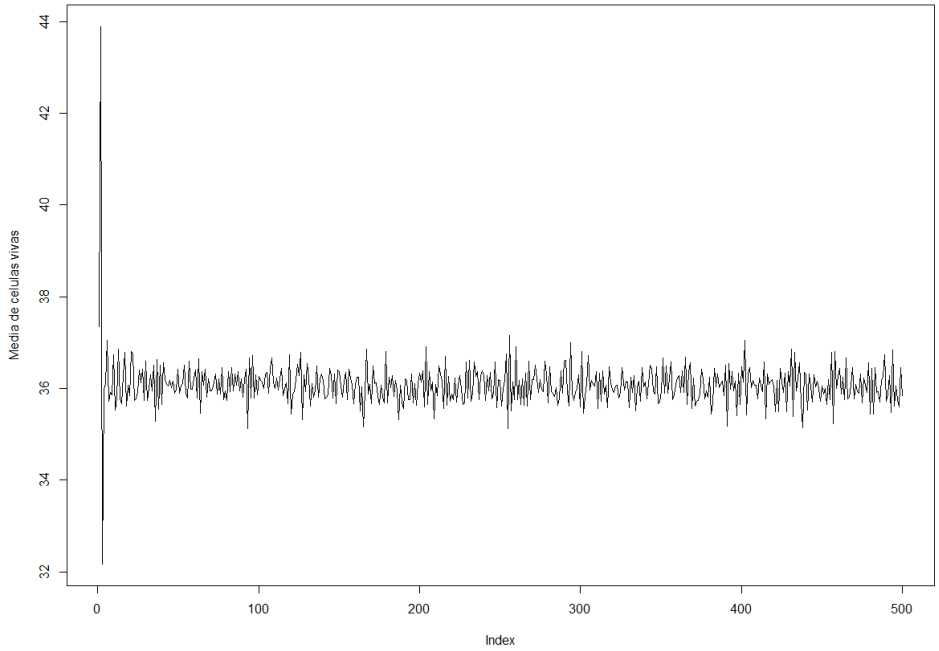


Figura 12. Número medio de células vivas en función al número de iteraciones para las reglas (1,2|2) en una matriz de 25x25

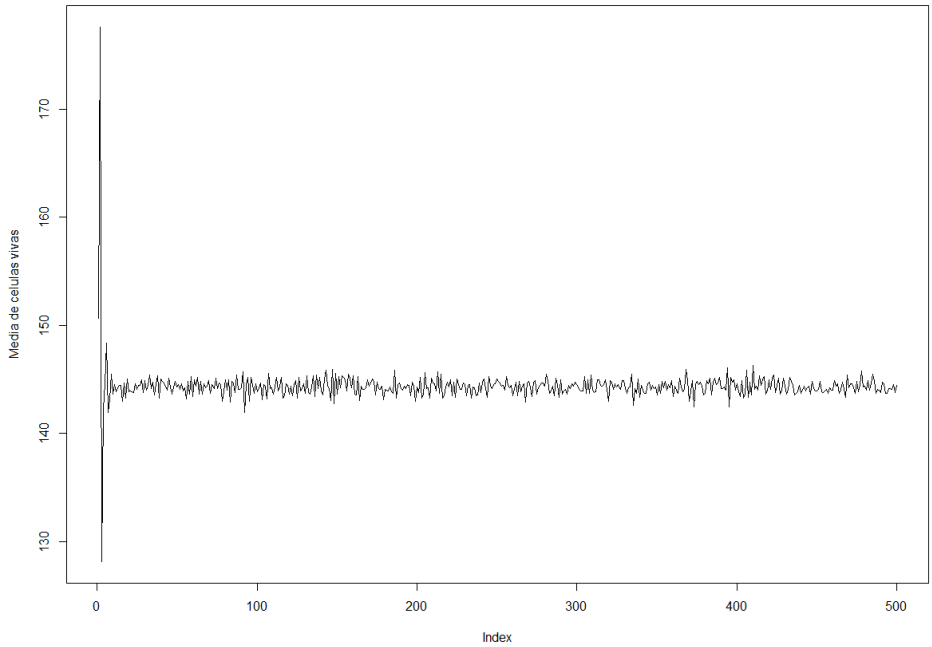


Figura 13. Número medio de células vivas en función al número de iteraciones para las reglas (1,2|2) en una matriz de 50x50

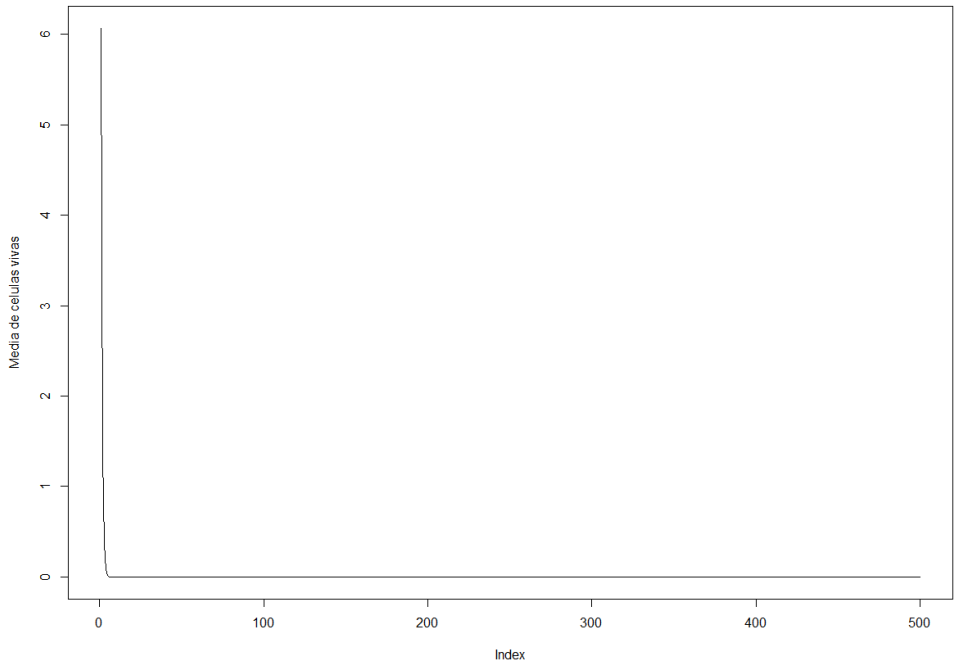


Figura 14. Número medio de células vivas en función al número de iteraciones para las reglas (4,6|6) en una matriz de 10x10

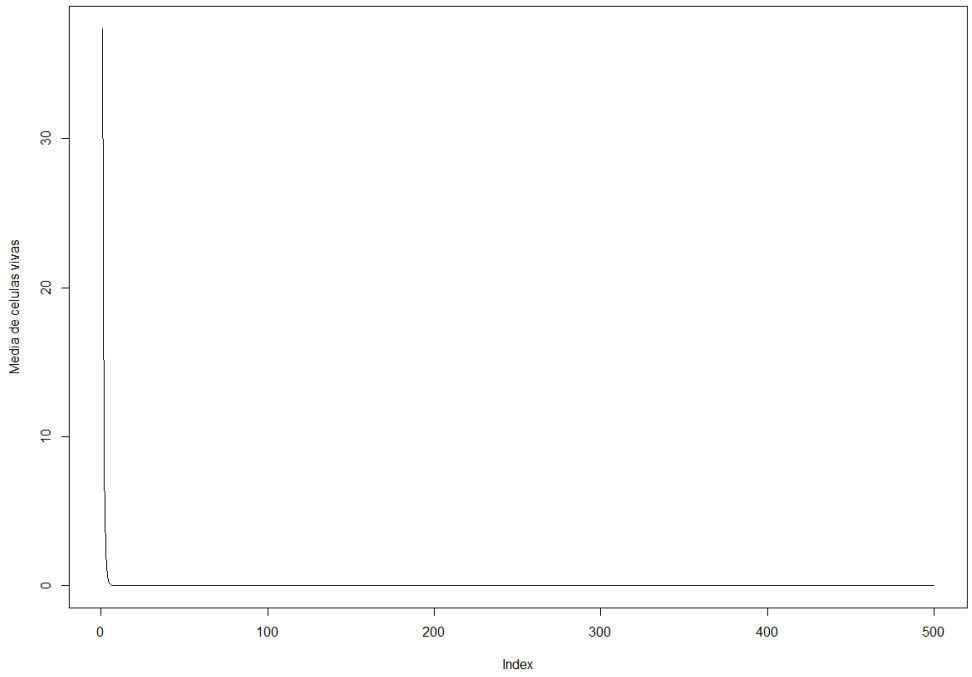


Figura 15. Número medio de células vivas en función al número de iteraciones para las reglas (4, 6|6) en una matriz de 25x25

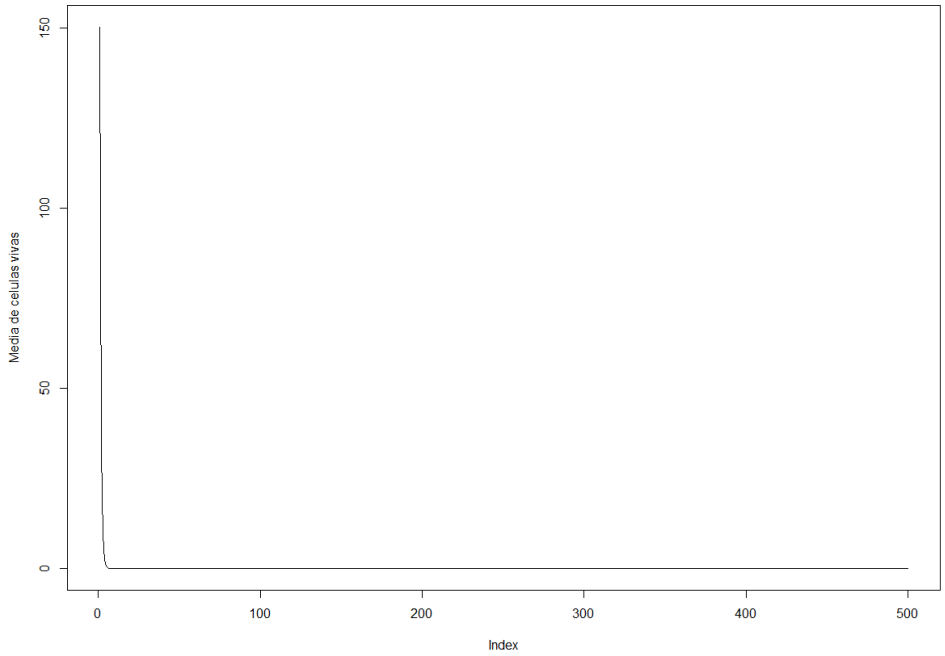


Figura 16. Número medio de células vivas en función al número de iteraciones para las reglas (4, 6|6) en una matriz de 50x50

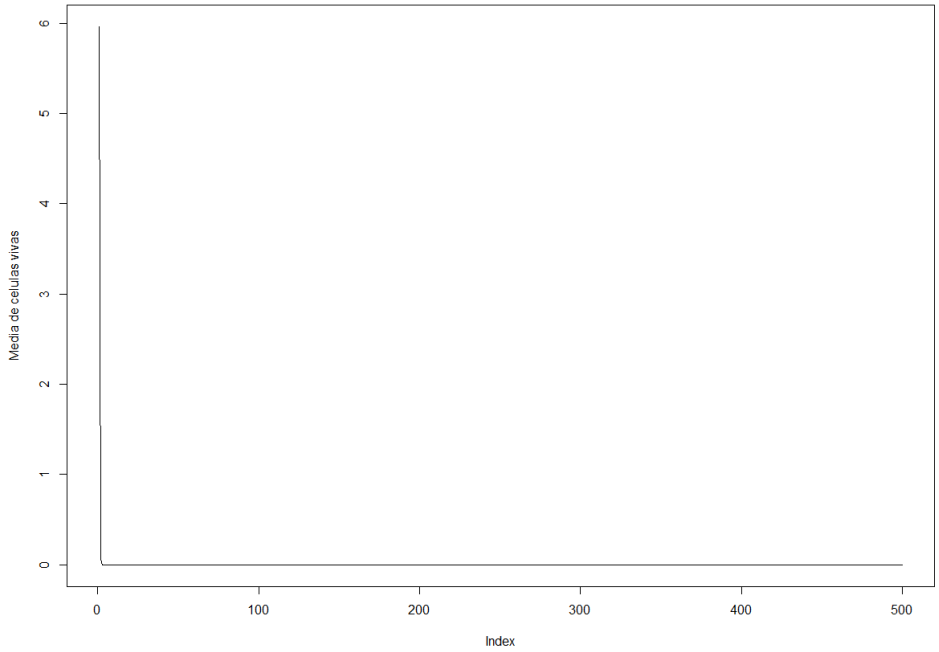


Figura 17. Número medio de células vivas en función al número de iteraciones para las reglas (6, 9|9) en una matriz de 10x10

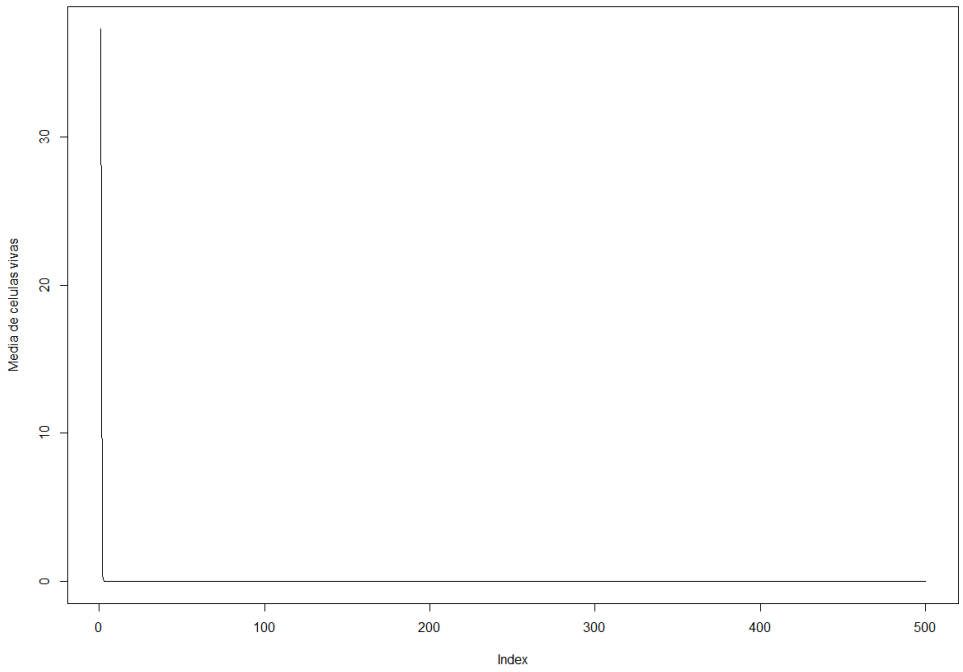


Figura 18. Número medio de células vivas en función al número de iteraciones para las reglas (6, 9|9) en una matriz de 25x25

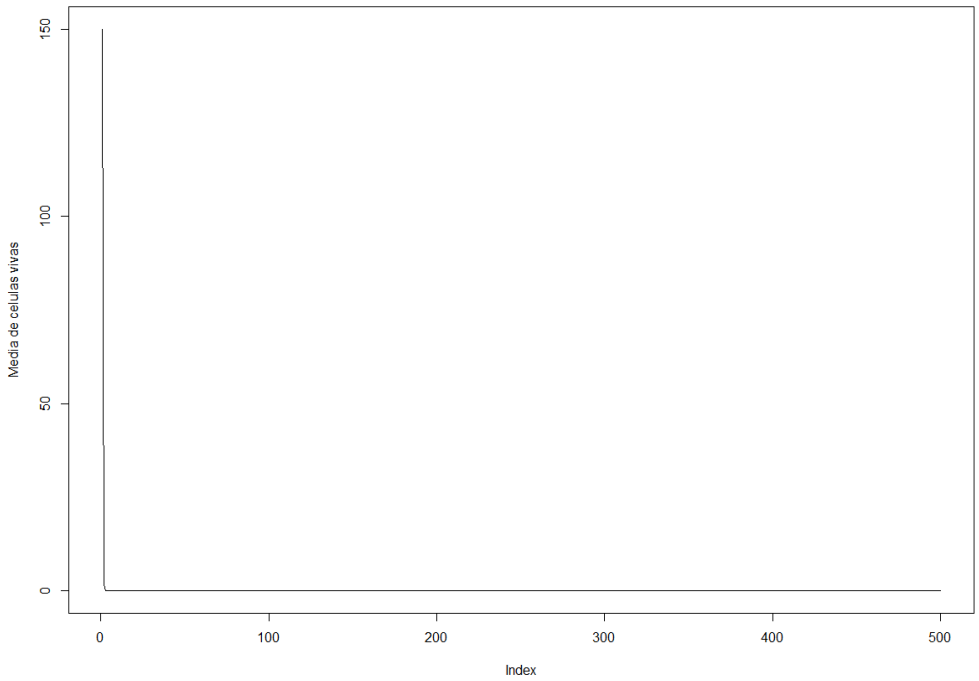


Figura 19. Número medio de células vivas en función al número de iteraciones para las reglas (6, 9|9) en una matriz de 50x50

Tal y como puede concluirse del estudio de las gráficas presentadas en este apartado para el estudio del efecto de las dimensiones de la matriz, las reglas que implican mayor número de celdas para estabilizarse y/o crear vida en celdas adyacentes tienden a hacer que el efecto

creado por el incremento de celdas vivas mediante el incremento de las dimensiones de la matriz desaparezca mucho más rápido. Este efecto es especialmente llamativo para las reglas (4,6|6) y (6,9|9).

Examinemos a continuación el efecto en la probabilidad de aparición inicialmente de una célula viva. Para ello, se representa la media de células vivas en 500 iteraciones del algoritmo para 100 matrices diferentes simuladas de dimensiones 30x30 cada una.

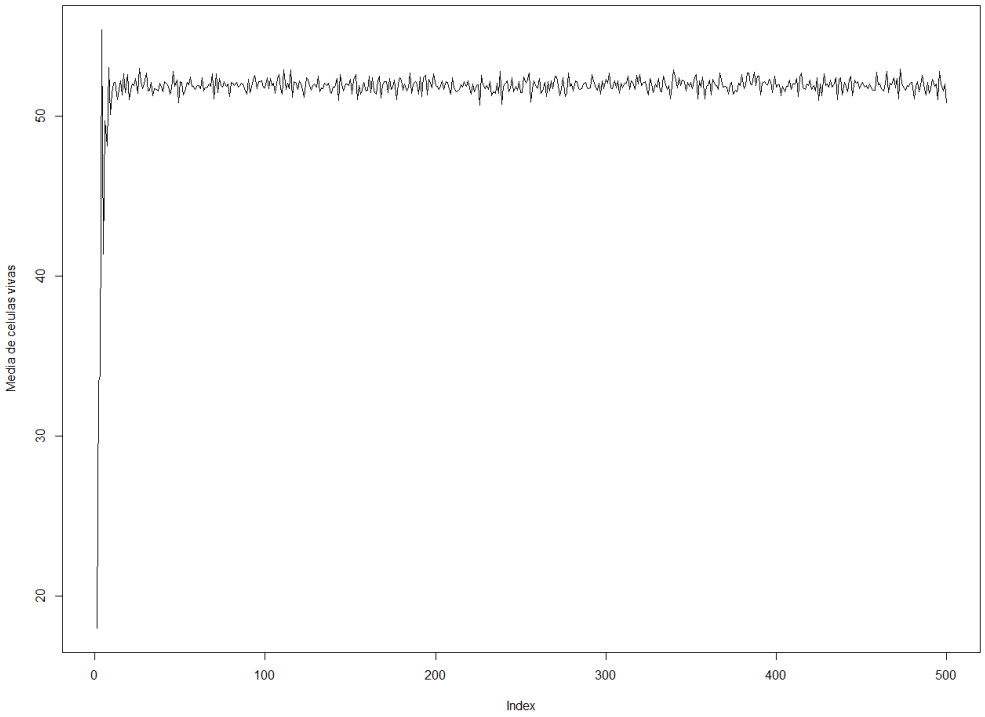


Figura 20. Número medio de células vivas en función al número de iteraciones para las reglas (1, 2|2) para una probabilidad inicial de vida de 0.1

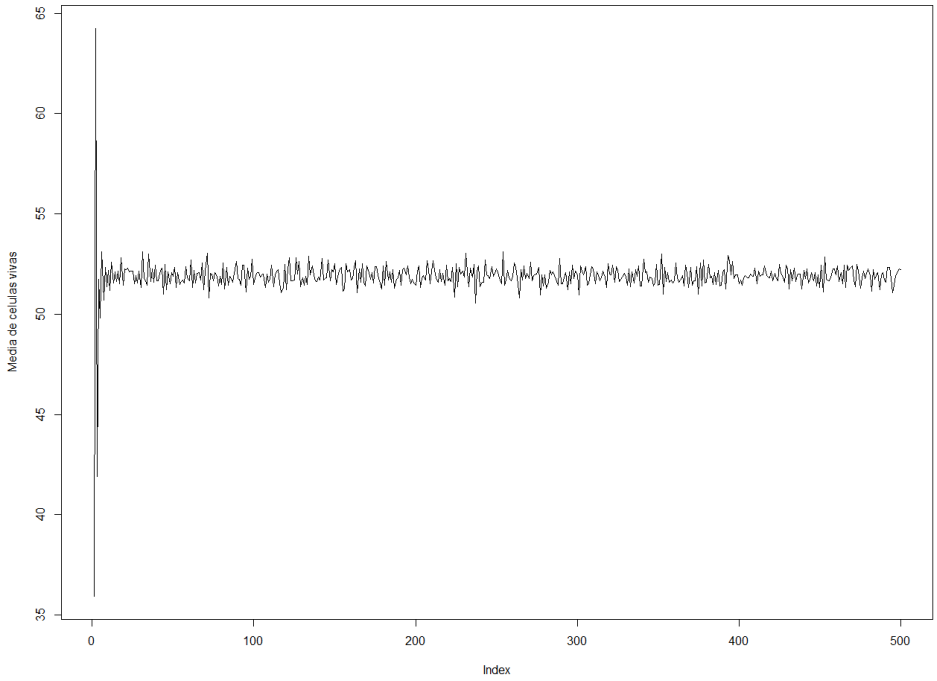


Figura 21. Número medio de células vivas en función al número de iteraciones para las reglas (1, 2|2) para una probabilidad inicial de vida de 0.2

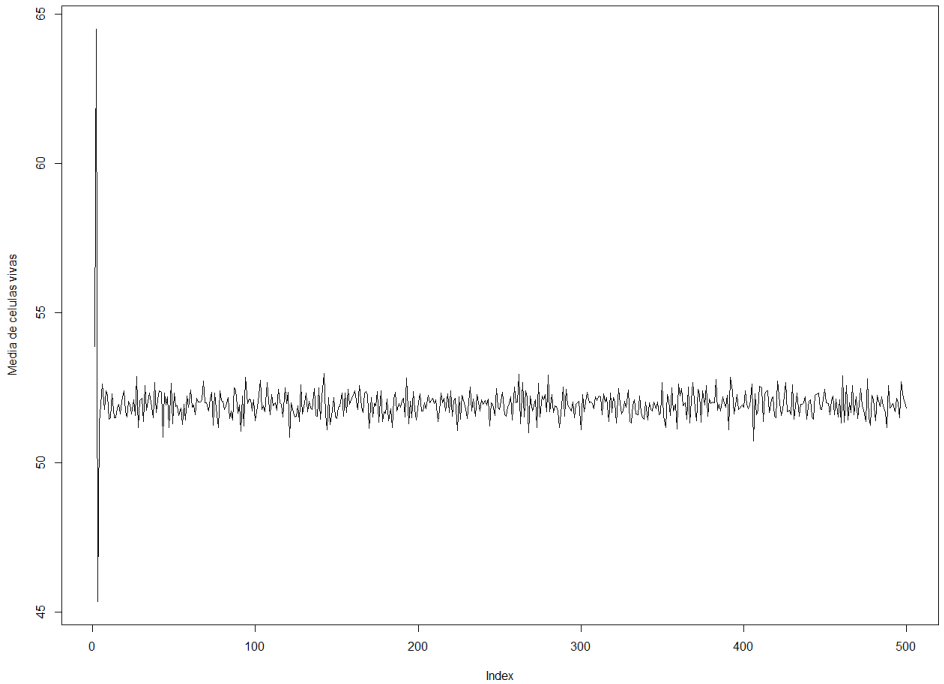


Figura 22. Número medio de células vivas en función al número de iteraciones para las reglas (1, 2|2) para una probabilidad inicial de vida de 0.3

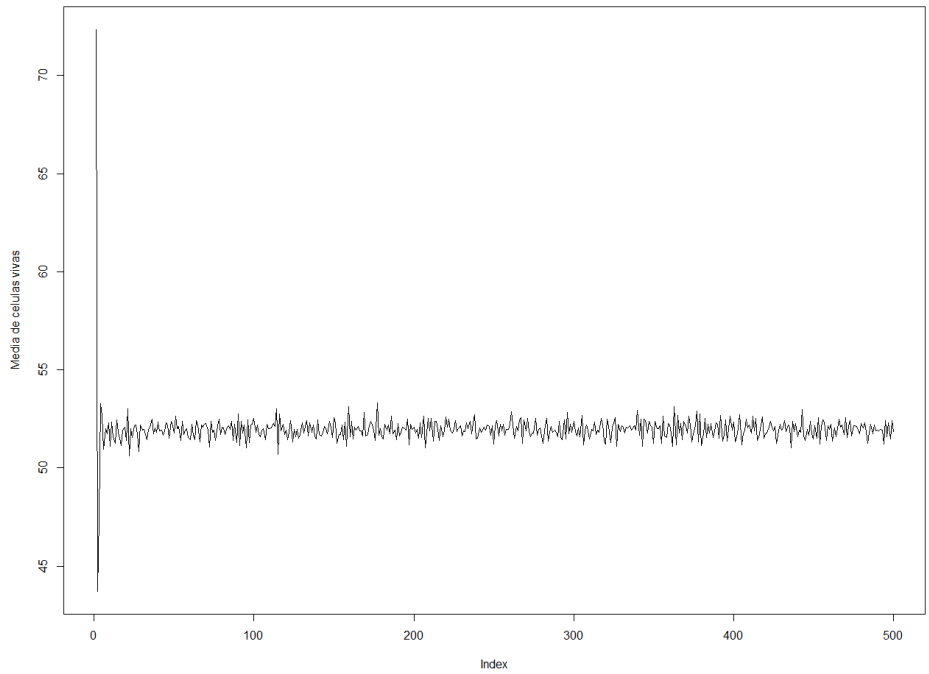


Figura 23. Número medio de células vivas en función al número de iteraciones para las reglas (1, 2|2) para una probabilidad inicial de vida de 0.4

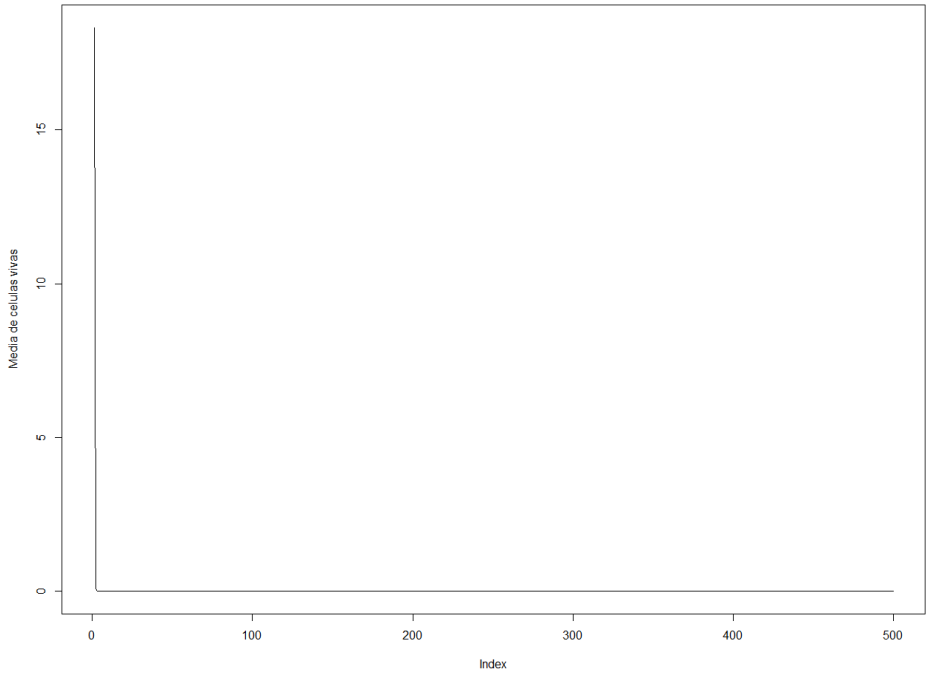


Figura 24. Número medio de células vivas en función al número de iteraciones para las reglas (4, 6|6) para una probabilidad inicial de vida de 0.1

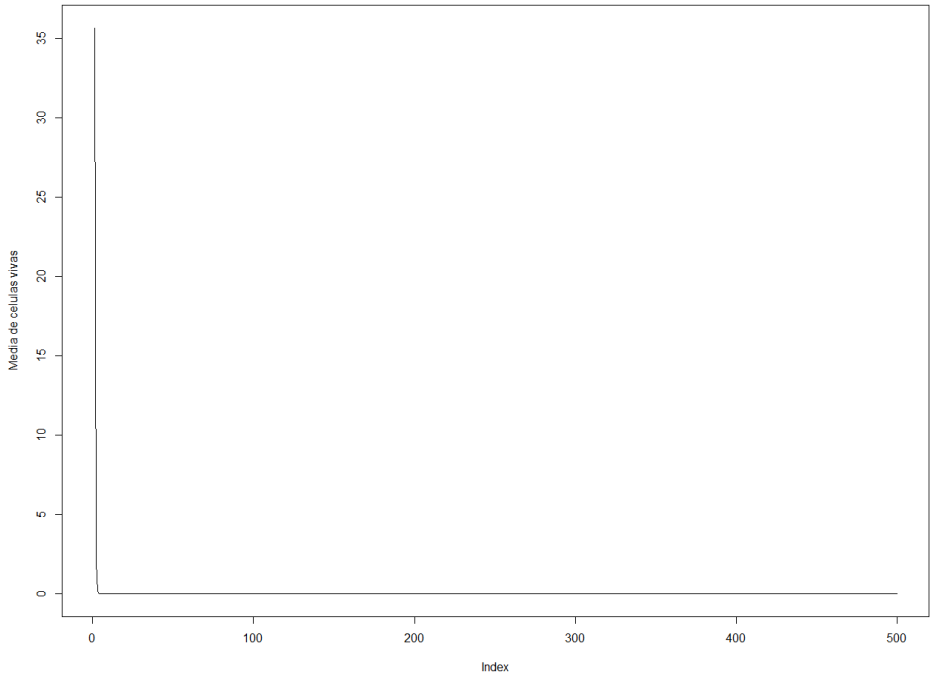


Figura 25. Número medio de células vivas en función al número de iteraciones para las reglas (4, 6|6) para una probabilidad inicial de vida de 0.2

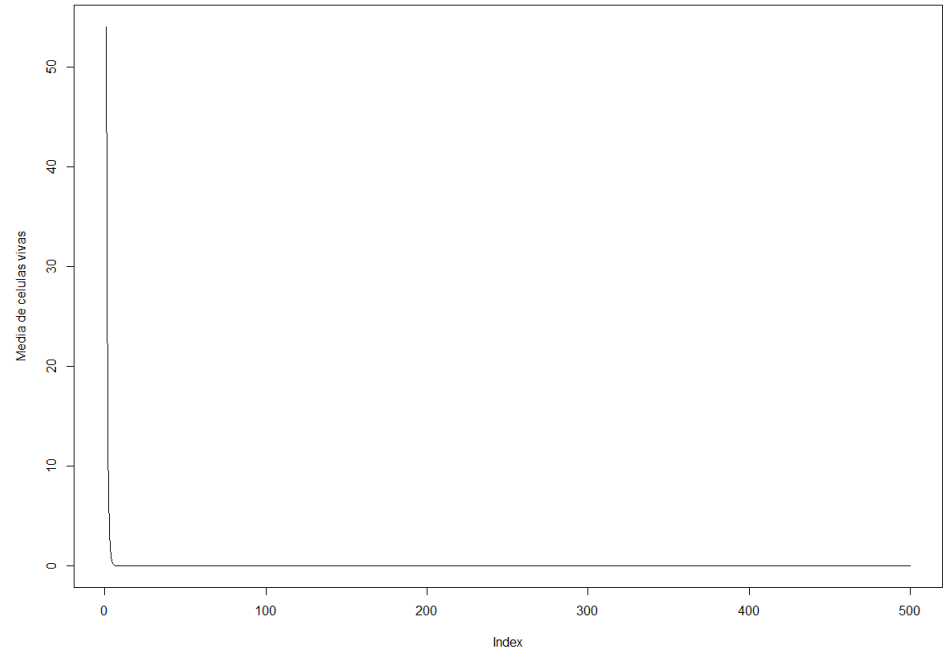


Figura 26. Número medio de células vivas en función al número de iteraciones para las reglas (4, 6|6) para una probabilidad inicial de vida de 0.3

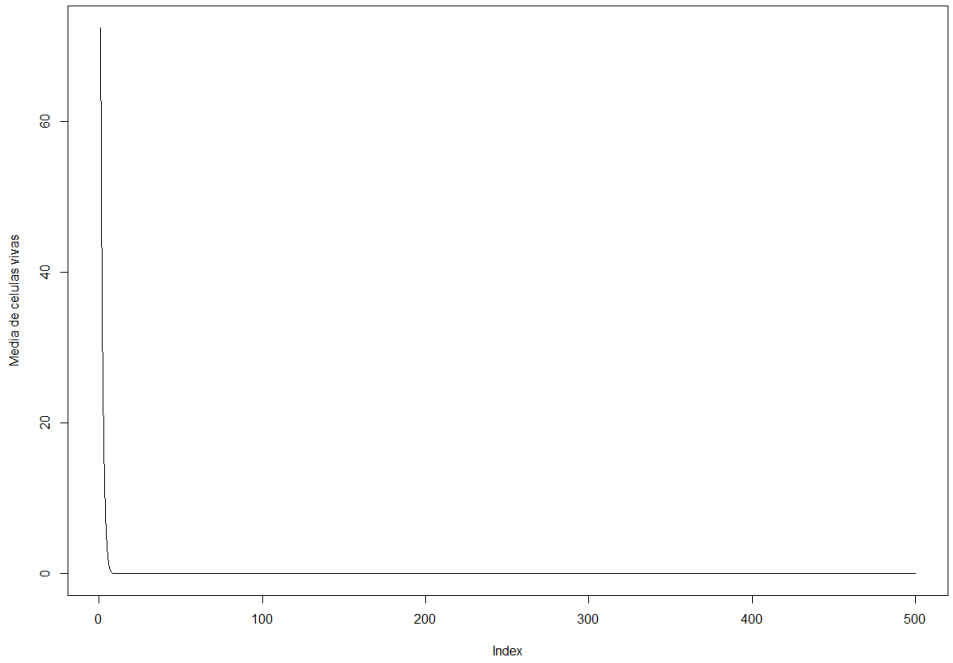


Figura 27. Número medio de células vivas en función al número de iteraciones para las reglas (4, 6|6) para una probabilidad inicial de vida de 0.4

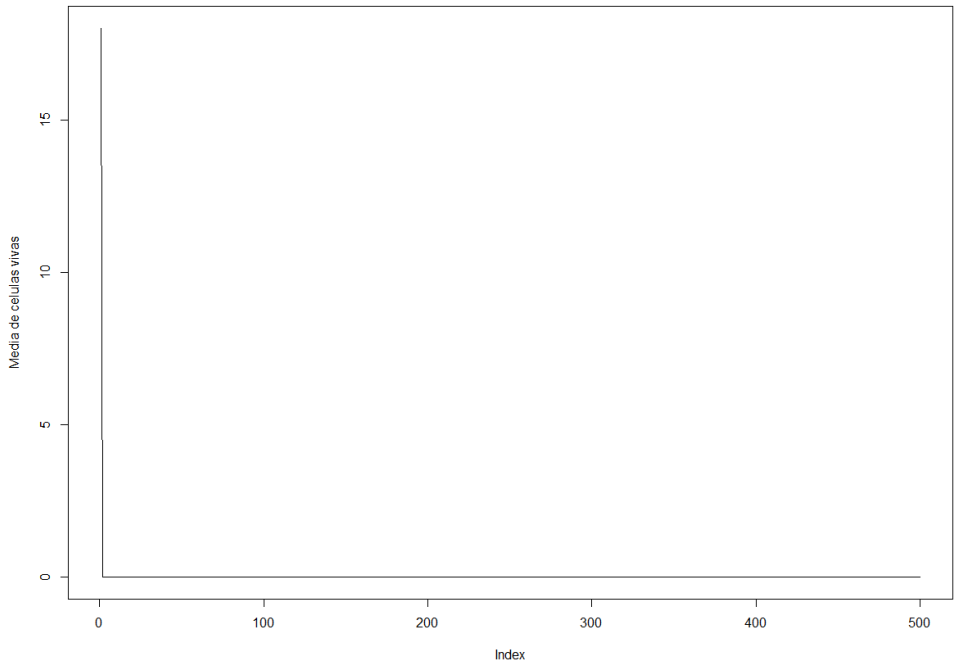


Figura 28. Número medio de células vivas en función al número de iteraciones para las reglas (6, 9|9) para una probabilidad inicial de vida de 0.1

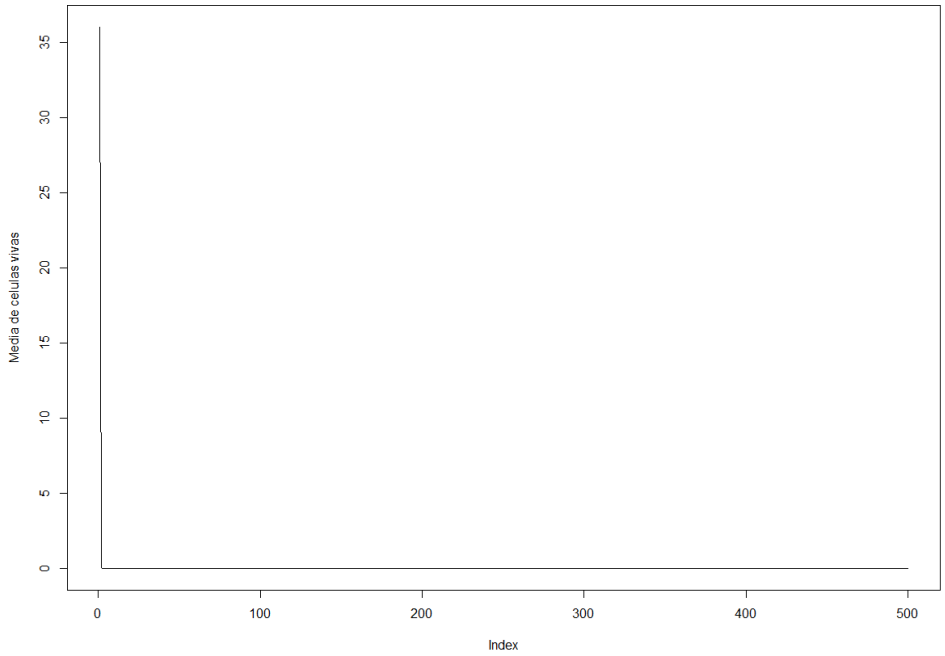


Figura 29. Número medio de células vivas en función al número de iteraciones para las reglas (6, 9|9) para una probabilidad inicial de vida de 0.2

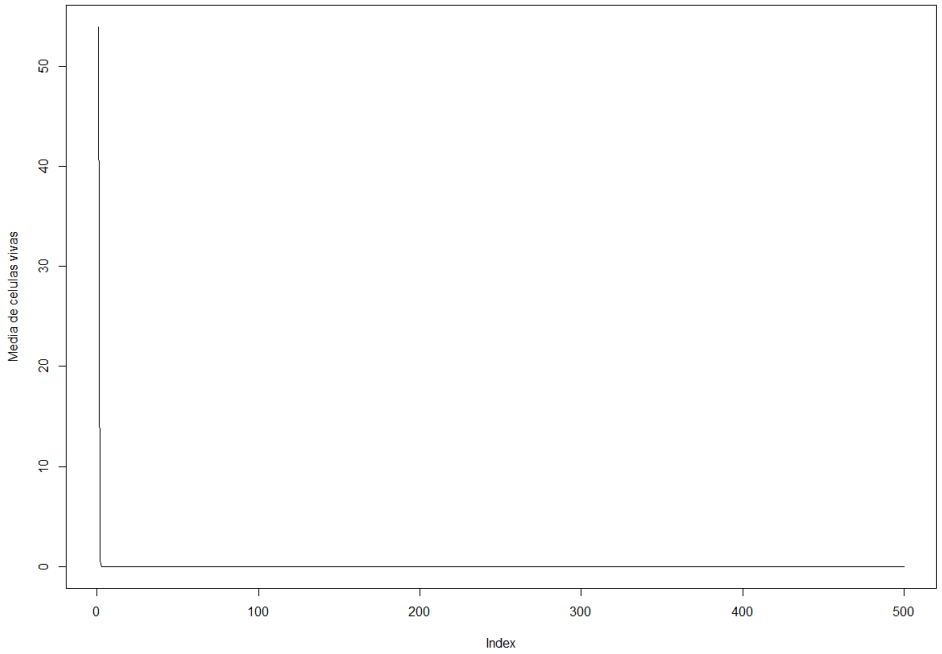


Figura 30. Número medio de células vivas en función al número de iteraciones para las reglas (6, 9|9) para una probabilidad inicial de vida de 0.3

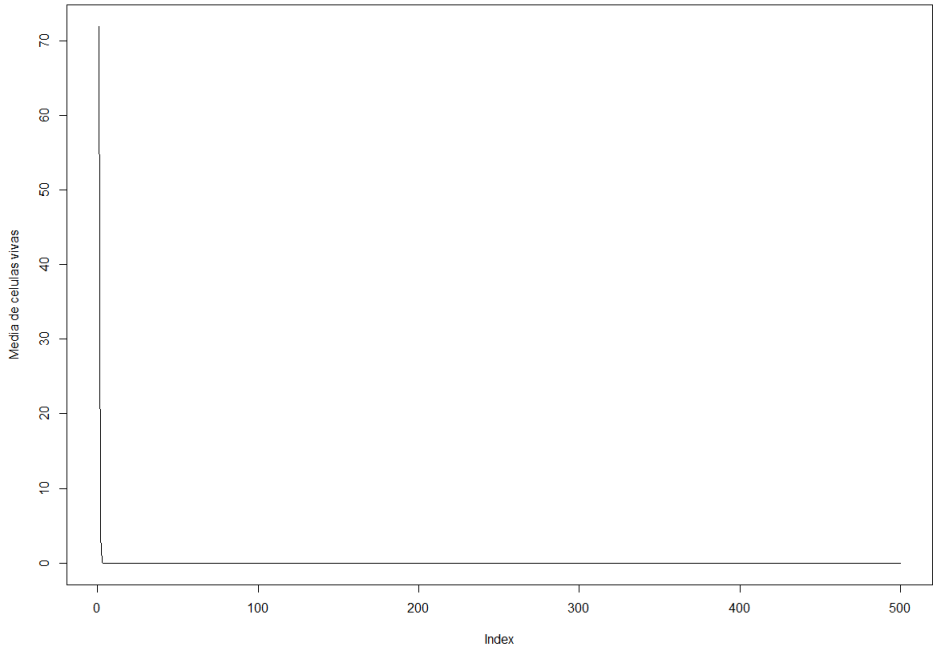


Figura 31. Número medio de células vivas en función al número de iteraciones para las reglas (6, 9|9) para una probabilidad inicial de vida de 0.4

Finalmente, examinemos el efecto de la probabilidad de aparición inicial de celdas vivas ante distintas reglas. Tal y como puede concluirse del estudio de las gráficas presentadas, las reglas que implican mayor número de celdas para estabilizarse y/o crear vida en celdas adyacentes tienden de nuevo a hacer que el efecto creado por el incremento de celdas vivas mediante el incremento de las dimensiones de la matriz desaparezca mucho más rápido. Este efecto es especialmente llamativo para las reglas (4,6|6) y (6,9|9), donde las estrictas condiciones de vida y mantenimiento hacen que en pocas iteraciones la media de células vivas sea "0". Esto no ocurre en las reglas (1,2|2), donde las reglas favorecen en pocas iteraciones del algoritmo un número medio de células vivas que oscila alrededor de 51 aproximadamente.

Del estudio de los efectos de las reglas en cuanto a tamaño de lado y de las probabilidades, podemos extraer la conclusión general de que las reglas que implican condiciones más laxas para la creación o mantenimiento de vida favorecen a determinadas reglas que implican mayor número de vida en matrices mayores. Por el contrario, las reglas más estrictas para las condiciones de vida se ven desfavorecidas ante este fenómeno, en especial su la probabilidad inicial de encontrar celdas vivas es baja (0.3). En cuanto al estudio de la probabilidad, todas las reglas estudiadas tienden a estabilizar en pocas iteraciones el número de celdas vivas; sin embargo, las reglas más laxas tienden a estabilizar el número de celdas vivas en valores positivos, mientras que las más estrictas tienden a hacerlo en "0".

Como mejora para esta sección, se propone estudiar estos dos fenómenos ante distintos tamaños de matrices, distintos valores de la probabilidad y distinto número de iteraciones, no contemplados en este estudio, para comprobar si las conclusiones aquí extraídas son generalizables.

Por último, y en referencia al código ejecutado en el mismo y teniendo en cuenta los resultados en los ejercicios 2, 3 y 4, puede comprobarse que el número de repeticiones realizadas de la función para recrear el juego de la vida resulta ser bastante pequeño, con tan

solo 100 iteraciones. Esto se debe a que conlleva un tiempo considerable realizar todos los cálculos que se proponen para la misma, siendo inviable realizar más repeticiones por cuestiones de tiempo. De este modo, una propuesta de mejora sería, por un lado, intentar optimizar la función presentada en el primer ejercicio de este informe. Igualmente, quizás la forma de resolver el conocido “problema de los bordes” puede no ser la óptima, por lo que se propone de cara a futuras mejoras intentar ver si con otras posibles soluciones a este problema, mejora el tiempo de ejecución del algoritmo.