F210 Programación Aplicada a Finanzas
Proyecto de Investigación
Otoño 2021

Los modelos de automatas celulares son una herramienta muy util en distintos campos de las ciencias para estudiar la propagación de shocks o ciertos fenómenos, en particular aquellos que dependen de la interacción entre los individuos que componen una población.

Se usa en modelos de ciencas sociales para analizar el proceso de formación de precios y expectativas, en ciencias naturales para la evolución de procesos epidemiológicos o de cambios en el medio ambiente y en ciencias físicas para estudiar el comportamiento de un conjunto de partículas, entre otros.

En este proyecto particular, utilizaremos el modelo de autómatas celulares para simular la propagación e impacto de un virus de contagio y analizaremos la evolución en el agregado de la población y en sus distintos estratos componentes, bajo diferentes escenarios.

Un mundo de automatas: La sociedad que vamos a construir para nuestro simulador va a asociar un autómata celular a cada individuo. Un autómata, para nuestros fines, será considerado como un conjunto de datos que tiene la posibilidad de evolucionar en el tiempo como consecuencia de la interacción con otros autómatas y con su entorno. En otras palabras, un autómata no es otra cosa más que un conjunto de *atributos* (al conjunto lo llamaremos *estado* del autómata) que pueden variar en el tiempo, dependiendo dicha variación del estado del autómata, de las características del mundo en el que se encuentra y de su interacción con otros autómatas.

1. **Setup Inicial. Mundo estático.** Para comenzar, consideraremos un primer mundo sencillo, formado por una grilla G que contiene G filas y G columnas. Cada fila y columna está numerada de 1 a G. En este primer mundo, el estado de cada autómata estará unicamente dado por un par de coordenadas (x,y), con $1 \leq x \leq G, 1 \leq y \leq G$, correspondientes a su ubicación. Aceptaremos en lo que sigue que dos autómatas puedan compartir una misma posición (x,y).

   Asumiremos que en $t = 0$ el mundo está poblado por N individuos. N puede ser cualquier número no negativo, pudiendo llegar a ser mayor que G^2 (el número total de casillas). La relación entre N y G^2 define la *densidad* poblacional promedio $\alpha = \frac{N}{G^2}$,que da la cantidad promedio de habitantes por casillero.

   El programa o función principal desde donde se correrá este mini proyecto se llamará **Viral_Automata1.R.** Si lo arman como función, no debe aceptar argumentos de entrada.

   Viral_Automata1 debe leer los parámetros del problema (por ahora G y N) de una función `Get_Parms1()` que los devuelve a la salida en una variable `Parametros`. Es libre de usar a la salida la estructura de datos que mas le plazca (vector, lista, matriz, dataframe).

   Una vez seteado `Parametros`, **Viral_Automata1** debe llamar a una función `Start_World1(G,N)`, que reciba como inputs el tamaño de la grilla G y de la población total inicial N y devuelva a la salida una estructura de datos (lista de vectores, matriz o dataframe) de $Nx2$, llamada `Poblacion,`conteniendo en cada fila i el estado inicial del autómata i, con su posición (x_0, y_0).

   Estas coordenadas inciales deben ser asignadas al azar de entre las componentes de un vector entre 1 y G.Para ello, usaremos la función de R `sample(v, size, replace = TRUE, prob = NULL)`que sirve para hacer un muestreo al azar de entre los elementos del vector de entrada `v,` muestreo de tamaño `size,` indicándose en el flag opcional `replace` si el muestreo se hará con o sin reposición. El flag `prob` contiene un vector del mismo tamaño que el vector `v,` indicando con qué probabilidad sale cada elemento v[i]$\in$ `v`. Desde ya que la suma de las probabilidades en prob debe ser 1. Por default prob = NULL y todos los elementos de v serán equiprobables a la hora del muestreo; y así la dejaremos por ahora. Usela en un par de ejemplos sencillos para familiarizarse con ella.

   Obtenida la estructura de datos `Poblacion,` nos interesa poder visualizar como quedo distribuida en la grilla. Para ello dentro de **Viral_Automata1** llamaremos a la función `Plot_Population1(Poblacion,Parametros)`, que no tiene valor de salida, y que en su interior grafique las posiciones iniciales x0 y y0 de los automatas usando la funcion `plot()` con los siguientes flags (averigue y explique que hace cada uno en el video) `pch=19, col= 'blue', xlim = c(1,G), ylim=c(1,G),xlab='x',ylab='y'`.

   Investigue como se utiliza la funciòn `legend()` para incluir una leyenda e incluya una,en el margen superior izquierdo del gráfico, indicando el valor de densidad a partir de los valores de N y G.

   Haga un video en zoom (video1) mostrando como le quedo el codigo y corriendolo para G =1000 y varios valores de densidad, $\alpha = 0.01, 0.1, 0.5, 1$ y 2

2. **Autómatas que se mueven.** En esta parte del proyecto, daremos movimiento a los autómatas para que se desplacen por la grilla. Para ello debemos introducir en el modelo un parámetro adicional: la *movilidad* M. Ella indica el número de casilleros que cada autómata se puede desplazar en un periodo dado en cada dirección. Supondremos por el momento que M es el mismo para todos los autómatas en nuestro mundo.

   El programa a desarrollar lo llamaremos **Viral_Automata2.** Lo primero que debe hacer el programa es llamar a una función `Get_Parms2()`que devuelve una estructura de datos `Parametros` conteniendo los nuevos parametros del modelo, G, N y M. Seguidamente, se debe generar la población inicial con la misma función `Start_World1(G,N)`del punto anterior y se la grafica del mismo modo que en el punto anterior con `Plot_Population1(Poblacion,Parametros).`

   Dada esta configuración inicial, procederemos a actualizar la posición de los autómatas llamando a una funcion `Move_Automata2(Poblacion,Parametros)`que recibe la estructura de datos `Poblacion` inicialmente creada y los parametros del modelo, devolviendo a la salida una estructura de datos igual a `Poblacion`, llamada `Poblacion_update,` pero con los estados actualizados. Para actualizar las posiciones, `Move_Automata2` debe hacer lo siguiente:

   - Usar la función `sample(v, size, replace = TRUE),` esta vez con `v = -M:M` y `size = N` para obtener el desplazamiento Δx de cada automata en la poblacion en la dirección x; y luego repetir el proceso para obtener los desplazamientos Δy en la dirección vertical.
   - Actualizar la posición de los autómatas haciendo $x' = x + \Delta x$ e $y' = y + \Delta y$ para cada uno de ellos, con (x', y') las nuevas coordenadas y (x, y) las anteriores. Al hacer esto, puede pasar que alguna de las coordenadas (x', y') se salgan de la grilla, es decir, que sean mayores que G o menores que 1. En tal caso, el autómata deberá reingresar a la grilla por el otro extremo, es decir: si la coordenada nueva fuese $G + s$, la coordenada a asignar será s; y si la coordenada nueva fuera $s \leq 0$, la coordenada a asignar será $G + s$.

   Obtenida `Poblacion_update,` utilice `Plot_Population1(Poblacion_update,Parametros)`para graficar las nueva distribución de los agentes.

   Arme ahora un loop abierto que al entrar le pregunte si desea recalcular el modelo un periodo mas, y en caso afirmativo actualice `Poblacion_update` haciendo `Poblacion_update <- Move_Automata2(Poblacion,Parametros)` y actualizando el gráfico de salida con `Plot_Population1(Poblacion_update,Parametros).`

   Haga un video en zoom (video2) mostrando como le quedo el codigo y corriendolo para cuatro escenarios con G =500, densidades $\alpha = 0.1$ y 0.5 y $M = 1, 10$.Haga unas 50 iteraciones para el loop en cada caso.

3. **Autómatas con edad y estado de salud** A fin de prepararnos para la simulación de propagación del virus, vamos a construir una estructura de datos más rica. El programa que generaremos en esta instancia es **Viral_Automata3.** El objetivo es introducir varios vectores más en la estructura de datos `Poblacion` (que solamente tiene por ahora las coordenadas x,y) :

   - El primer vector es el de `id,` que como su nombre lo indica está asociado univocamente al automata i. En definitiva, tendremos que hacer `id<-1:N,` tanto en la función de inicialización `Start_World3()`como en la actualización de coordenadas `Move_Automata2().`
   - Otro vector será denominado `tiempo` o `periodo,` que usaremos para poder construir ya no una foto del estado de la población en un instante dado, sino una película que permita estudiar la evolucion del virus en el tiempo. En la función de inicialización `Start_World3(Parametros)`se seteará `tiempo <- rep(0,N)`para todos los individuos y en la actualización de estados `Update_Population3(Poblacion,Parametros)` - ver mas abajo - se deberá leer el máximo tiempo t existente en `Poblacion` e incrementarlo en 1 para el estado de la `Poblacion` en t+1.
   - Un vector `Age_Group,` que indicará el grupo etario al que pertenece cada individuo. Los vamos a agrupar en 4 grupos etarios, a saber

$$\texttt{Age\_group} = \begin{cases} 1 & 0-19\ yr & p=0.4 \\ 2 & 20-49\ yr & p=0.3 \\ 3 & 50-69\ yr & p=0.2 \\ 4 & 70+\ yr & p=0.1 \end{cases}$$

     donde `p` indica la probabilidad de que un individuo en la población pertenezca a dicho grupo etario. Esta información debe ser incorporada a la estructura `Parametros` para luego ser pasada a `Start_World3(Parametros).` Dentro de `Start_World3` se debe crear ahora el vector `Age_Group` de Nx1 que formará parte de la estructura de datos de salida `Poblacion.`Dicho vector se mantendrá inalterado a lo largo de las simulaciones, ya que supondremos despreciable la probabilidad que una persona cambie de grupo etario en el lapso de tiempo en que transcurren las simulaciones.

     Para asignar cada individuo a un grupo etario, usaremos nuestro viejo amigo `sample(1:4,N,replace = TRUE, prob = p)`ya que ahora los resultados no son equiprobables. Este vector nos va a permitir estudiar diferencialmente como el virus actua en una persona que se enferma y eventualmente desarrolla sintomas de gravedad y fallece. Pero eso para la proxima etapa.
   - Un vector `Health,`indicando el estado de salud de cada autómata. Vamos a definir 4 estados de salud posibles:

$$\texttt{Health} = \begin{cases} "S" & \text{Sano} \\ "E" & \text{Enfermo} \\ "G" & \text{Enfermo grave} \\ "R" & \text{Recuperado} \\ "F" & \text{Fallecido} \end{cases}$$

     En esta etapa no simularemos ni contagios ni evolución de la enfermedad, pero sí queremos ya tener armada la estructura de datos para más adelante. Este se inicializará dentro de `Start_World3(Parametros)` y debe salir de la función dentro de la estructura de datos `Poblacion.` La asignación inicial de estados de salud a los agentes por ahora la haremos aleatoria muy sencilla: `sample(c("S","E","G","R","F"),N,replace = TRUE),` con lo cual habra un 20% de la población de cada tipo aproximadamente. A medida que pase el tiempo y los agentes se vayan moviendo, habrá una instancia de actualización del estado de salud de cada individuo, pero no en esta etapa. Los automatas mantendrán, al igual que su grupo etario, su estado de salud inicial en el tiempo.

   Viral_Automata3 debe entonces hacer lo siguiente.

   (a) En primer lugar, llamar a una función `Get_Parms3()`y asignar a la salida la estructura de datos `Parametros` con los parametros del modelo.

   (b) Llamar a `Start_World3(Parametros)`que devuelva a la salida la estructura de información `Poblacion` en t=0, conteniendo las variables `id,x,y,tiempo,age_group,health` de cada autómata.

(c) Con `Poblacion` y `Parametros` asignados, se debe iniciar un lazo cerrado que realizará **K** iteraciones, siendo cada iteración un avance en el tiempo. **K** debe definirse al principio de **Viral_Automata3.** En cada iteración, se debe llamar a una función `Poblacion<-Update_Population3(Poblacion,Parametros)`que debe recibir una `Poblacion` dada en un momento t y un vector de `Parametros` del modelo y actualizarla para el periodo t+1; llamando primero a `aux<-Move_Automata3(Poblacion,Parametros)` que unicamente actualice las coordenadas de los autómatas, y luego usando la salida `aux` e introduciendola en una nueva funcion `salida<-Update_State3(aux,Parametros)` que debe cambiar el `tiempo` t presente en `aux` por el valor siguiente $t+1$.

(d) Las sucesivas salidas de `Update_Population3` que van saliendo en cada iteración deben ser almacenadas en una estructura de datos que contenga la evolucion de las poblaciones en el tiempo. Llamaremos a esa estructura `Historia`.Por ejemplo, si `Historia` y `Poblacion` se los piensa como dataframes, bastaria con hacer `Historia<-rbind(Historia,Poblacion)`para ir recopilando la historia.

(e) Completada la `Historia,` debe introducirla en otra funcion `Animate_History(Historia,Parametros,step),` que no tenga resultado a la salida y que realice lo siguiente: Al entrar debe setear $t = 0$ y disparar un loop abierto que se ejecute mientras $t \leq K$, el máximo tiempo en la simulación. En cada iteración, se debe extraer de `Historia` la `Poblacion` para la cual el `tiempo` $= t$ y hacer un gráfico con la distribución de la `Poblacion` mediante la función `Plot_Population2(Poblacion_update,Parametros).`

Finalizada esta etapa, se debe consultar al usuario si quiere continuar con la animacion o detenerla. En caso de querer continuar, se debe actualizar el tiempo a $t' = t + step$ y volver al inicio del loop.

Para confeccionar `Plot_Population2(Poblacion_update,Parametros),` la estructura del plot debe ser similar a esta para permitir la diferenciacion de los puntos por `health` y `Age_Group`

```
plot(x,y,xlim=c(1,G),ylim=c(1,G),
     col=c('green','orange','red','purple','black')[as.factor(Health)],
     pch=c(15,17,19,23)[Age_Group]))
```

Note que los flags col y pch ahora tienen vectores porque cada estado de salud y grupo etario se graficarán con un color y forma propia para su visualizacion. Realice un video (video 3) mostrando el trabajo realizado y la salida del programa con la animación para $G = 500$, $\alpha = 0.2, M = 5$, $K = 1000$ y $step = 10, 25\ y\ 50$.

4. **Contagio.** Llegado a este punto, estamos en condiciones de modelar la dinámica de contagio y evolución de la enfermedad de los agentes en la grilla. Vamos primero a describir el contexto, el comportamiento de los autómatas y del virus, y luego a plantear la estructura computacional para llevar a cabo la simulación, y los resultados que pretendemos obtener. El contexto de base es el del problema anterior: autómatas caracterizados por su posición en una grilla de tamaño G, con un grado de movilidad M, que se desplazan por la grilla libremente. Tienen asociados además dos atributos, su grupo etario `Age_group` y su estado de salud `health`. A la estructura de datos le agregamos una variable adicional `tiempo` para indicar a que momento del tiempo (periodo) al que corresponde cada observacion de la estructura de datos.

   En este mundo, vamos primero a describir el proceso de contagio. En principio, cualquier autómata enfermo (sea "E" o "G") supondremos que tiene la capacidad de contagiar a los autómatas sanos o recuperados ("S" o "R") que los rodean. Los fallecidos ("F") vamos a suponer que ya no tienen la capacidad de contagiar. Cómo se produce el mecanismo de contagio entre enfermos y no enfermos?

   Los agentes enfermos esparcen en las celdas de la grilla una cierta **carga viral,** que llamaremos $l\,(x, y)$ .Obviamente la carga viral será máxima en la celda en la que está el autómata enfermo, e irá decreciendo a medida que aumenta la distancia.

   Por su parte, un agente sano "S" que se encuentre ocupando la celda (x, y) en un momento dado del tiempo, estará en contacto en dicha posición con cierta **carga viral total** $L\,(x, y)$, que resulta de sumar las contribuciones $l\,(x, y)$ de todos los agentes enfermos que hay en la grilla a la carga viral en la posición (x, y) . La carga viral total estará relacionada a la probabilidad de contagiarse. que para los fines de este proyecto supondremos que tiene la forma

$$P(E \mid S, x, y) = 1 - \exp^{-\beta L(x,y)}$$

   expresando que la probabilidad de pasar de Sano a Enfermo estando en la celda (x, y) depende de la carga viral total en esa celda y de un parámetro $\beta \geq 0$ que indica la contagiosidad de dicha carga viral. Claramente, un β alto implica que un poquito de carga viral alcanza para que quien este en esa celda se enferme casi con certeza,

y $\beta = 0$ implica que el virus no enferma ni aunque se acumule sustancialmente. Por ejemplo, si β fuera una función de (x, y), serviría para modelar la diferencia de contagio en espacios abiertos (β bajo) versus espacios cerrados (β mas alto). Para esta etapa del proyecto lo consideraremos constante a lo largo de la grilla.

Con esto, nos queda por determinar como computar la carga viral individual $l(x, y)$. Vamos a suponer que un agente enfermo parado en la celda (x_0, y_0) tiene un impacto que no va más allá de R casillas (R por radio de acción). En nuestro mundo todavía nadie usa barbijos ni nada que pudiera reducir R. La carga viral entonces que puede propagarse de un enfermo que está en (x_0, y_0) vamos a suponer que sigue la siguiente funcion

$$l(x, y) = \begin{cases} e^{-\frac{\left[(x-x_0)^2+(y-y_0)^2\right]}{\sigma^2}} & \text{si } |x - x_0| \leq R \text{ y } |y - y_0| \leq R \\ 0 & \text{otro caso} \end{cases}$$

con σ un parametro que muestra como decae la carga viral con la distancia dentro del radio de accion R.

Nos resta poder modelar la evolucion de los enfermos. Para ello, vamos a agregar en primer lugar una variable más al estado de los autómatas, a saber:

- **El tiempo transcurrido en cada estado de enfermedad.** Cada automata tendrá asociado un "clock" propio, para cuando potencialmente transite la enfermedad. Llamaremos a dicho clock `timesick.` Los pacientes sanos "S" lo tendrán en todo mometo en 0. Cuando un agente sano se enferme, pasará de estado "S" a "E" y `timesick` se seteará en un valor **TS** > 0. Este número indicará la cantidad de periodos futuros durante los cuales el autómata puede esparcir carga viral a su alrededor y eventualmente pasar a ser un paciente grave "G". Por cada periodo que un enfermo mantenga su estado "E" y no se agrave, disminuirá `timesick` en 1 unidad (le falta un periodo menos para salir de alta). Si para un enfermo "E" se alcanzara `timesick = 0,` éste pasará a ser recuperado "R"; en cuyo caso la variable `timesick` se establecerá en **TR** $>$**0**, el tiempo que un paciente recuperado quedará "inmune" al contagio del virus. Al igual que con los enfermos, a los recuperados también les corre el clock de la inmunidad, disminuyendo hasta llegar a 0, momento a partir del cual pasarán a ser sanos "S", y proclives a enfermarse nuevamente.

  En caso que un paciente enfermo "E" pase a ser grave "G" (a terapia intensiva), se adicionarán **TG** > 0 periodos a `timesick,` y se continuará con la cuenta regresiva. Desde esta perspectiva, un paciente que se agrava no solamente tardará mas tiempo en recuperarse, sino que ahora entrará en riesgo de fallecimiento. Si el paciente grave "G" no se muere a lo largo de su `timesick` pasará a ser un "R" recuperado con `timesick=`**TR** $>$**0** como en el caso de los enfermos que se curan de tipo "E".

Para modelar la evolución de un paciente enfermo "E", vamos a usar una probabilidad condicional. Supondremos que con cierta probabilidad $PEG > 0$ un paciente Enfermo pasa a ser Grave, y su `timesick` se incrementará en **TG**. Obviamente, con probabilidad $1 - PEG$ el paciente permanecerá en estado Enfermo, y su `timesick` se reducirá en una unidad.

Cuando un paciente está grave, en nuestro modelo vamos a suponer que sigue contagiando gente, pero podría no ser el caso si supusieramos que está hospitalizado.

Finalmente, un paciente grave "G", tiene mientras `timesick` sea mayor que cero una cierta probabilidad de morir $PGF > 0$ en cada periodo. Si muere, su estado pasa a ser "F", fallecido, su `timesick` pasa a ser 0 y permanece en ese estado por el resto de la simulación. Como ya dijimos, los fallecidos supondremos que no contagian, con lo que podremos sacarlos del mapa, enviarlos a una ubicacion fija, (1,1) por ejemplo, o dejarlos seguir moviendose por la grilla. A los fines prácticos dara igual.

Con esta estructura ya podemos armar un modelo completo de contagio, evolución de la enfermedad y fallecimientos. Lo llamaremos `Viral_Automata4,y` debe hacer lo siguiente:

(a) Llamar a una función `Get_Parms4()`y asignar a la salida la estructura de datos `Parametros` con todos los parametros del modelo (G, N, M, p, $\beta, R, \sigma, TS, TG, TR, PEG$ y PGF).

(b) Llamar a `Start_World4(Parametros)`que devuelva a la salida la estructura de información `Poblacion` inicial en `tiempo`= 0, conteniendo las variables `id,x,y,tiempo,age_group,health y timesick` de cada autómata. Esta función trabajará igual que `Start_World3,` excepto que devolverá a todos los autómatas de la población en estado "S", sanos, y con `timesick=0`.

(c) Con `Poblacion` y `Parametros` asignados, `Viral_Automata4` seleccionará al azar una determinada cantidad de "pacientes cero", N_0, para iniciar el brote del virus. N_0 debe definirse al principio de **Viral_Automata4.** En principio se puede tomar $N_0 = 1$ o $N_0 = 10$ e ir viendo que pasa. Usualmente será un número pequeño. Para ello se usará `sample(1:N,`N_0`,replace=F)` a fin de seleccionar los autómatas a infectar, y se los pasará de estado "S" a estado "E", con su `timesick` pasando a TS.

(d) Seguidamente, se debe iniciar un lazo cerrado que realizará **K** iteraciones, siendo cada iteración un avance en el tiempo de la simulación. **K** debe definirse al principio de **Viral_Automata4.** En cada iteración, se debe llamar en primer lugar a una función `Poblacion<-Update_Population4(Poblacion,Parametros)`que debe recibir una `Poblacion` dada en un momento t y un vector de `Parametros` del modelo y actualizarla para el periodo t+1; de la siguiente forma

- Llamando primero a `L<-Compute_Load4(Poblacion,Parametros)`que recibe una dada `Poblacion` y `Parametros` y devuelve a la salida una matriz de G filas y G columnas conteniendo la carga viral total que hay en cada celda (x,y) de la grilla. Use la expresion de $l(x, y)$ definida más arriba para calcular el impacto individual, y acumule las contribuciones individuales en la matriz `L` $= L(x, y)$
- Seguidamente se analiza la evolución de los agentes. Llamaremos a una función `Poblacion <-Evolve4(Poblacion,` que primeramente separa los agentes por su estado de salud de entrada.
  A los agentes "E" y "G" de la `Poblacion`. se les va a asignar su nuevo estado de salud `health`.A cada uno de estos agentes se les asignará un número aleatorio entre 0 y 1 haciendo `random <-sample(0:100000,1,replace=T)/100000.`
  Los autómatas que eran tipo "E" , sabremos que pasarán a terapia intensiva "G" si `random` $<$ `PEG` y se mantendrán en estado "E" en caso contrario. Actualice el clock `timesick` de estos enfermos segun corresponda como ya se especificó antes.
  En el caso de los automatas que eran tipo "G", el numero aleatorio random se comparará contra PGF, determinando que si `random` $<$ `PGF,` el autómata fallece en esa fecha y pasa a estado "F" de ahi en mas y su `timesick` pasa a 0. Será un "fallecimiento" para ese dia. Caso contrario, el automata permanece en estado "G" y su `timesick` se reducirá en 1.
  En todos los casos en que un enfermo "E" o "G" llegue a `timesick` 0, se lo debe pasar a "R", recuperado, con timesick TR>0.
  Los automatas que ya eran de tipo "R" deben disminuir su `timesick` en 1, ya que les queda menos tiempo de inmunidad residual.
  Finalmente, nos queda ver quienes de los inicialmente sanos se contagian. Al igual que a los enfermos, a cada agente originalmente sano, "S", se le asigna un numero aleatorio entre 0 y 1 `random <-sample(1:100000,1,replace=T)/100000.` Sabemos que una persona en una celda se enferma con cierta probabilidad PSE $= P(E \mid S, x, y) = 1 - \exp^{-\beta L(x,y)}$. Computaremos entonces, dada la matriz `L`, una matriz asociada `PSE` conteniendo las probabilidades de contagio en cada posición de la grilla. Para cada agente sano en una celda (x,y) cualquiera, buscaremos la PSE correspondiente a dicha celda y diremos que un agente se infecta si `random` $<$ `PSE` y se mantendrá sano en caso contrario. De enfermarse, su estado pasará a ser "E" y será un caso de contagio. Debe adecuarse el clock `timesick` al valor TS en ese caso.
  `Evolve4` devuelve entonces a la salida una `Poblacion` actualizada en su estado de salud.
- Con el estado de salud de la población actualizada, se llama ahora a `Poblacion<-Move_Automata4(Poblacion,Para` que actualiza las coordenadas de los autómatas, al igual que en los puntos anteriores.
- Finalmente, se cambia el valor de la variable `tiempo t` de la Poblacion por su valor siguiente $t + 1$.

(e) Las sucesivas fotos de `Poblacion` que salen de `Update_Population4` en cada iteración deben ser almacenadas en una estructura de datos que contenga la evolución de las poblaciones en el tiempo. Llamaremos a esa estructura `Historia`.Por ejemplo, si `Historia` y `Poblacion` se los piensa como dataframes, bastaria con hacer `Historia<-rbind(Historia,Poblacion)`para ir recopilando la historia total de la pandemia.

(f) Completada la `Historia, Viral_Automata4` debe introducirla en otra funcion `Animate_History(Historia,Parametr` similar a la del punto 3, que permita ver la propagación del virus en un gráfico.

(g) Asimismo, a partir de `Historia,` se le pide que construya una serie de reportes gráficos que muestren:

- Cantidad total de casos a lo largo del tiempo. Un individuo que se enfermó y se recupero o se murio, cuenta como un caso. Un reincidente debería contarse mas de una vez.
- Evolucion de camas UTI empleadas en cada momento del tiempo (enfermos graves).

- Cantidad de fallecidos por día y total acumulado en la pandemia.
- Cantidad de contagios (gente que pasa de "S" a "E") para cada momento del tiempo.

Realice un video (video 4) mostrando el trabajo realizado y la salida del programa con la animación para $G = 200$, $N = 8000, M = 10$, $R = 5$, $\beta = 2$, $\sigma = 3$, $TS = 14$, $TG = 7$, $TR = 90$, $PEG = 0.01$, $PGF = 0.01, K = 1000$ y $step = 10$.

Elija dos parámetros de su interés que tengan que ver con el modelo de contagio, y estudie cómo cambian los puntos f y g cuando los modifica individualmente. Muestre sus resultados en otro video (video 5) Explique conceptualmente en el video que significarían las modificaciones que está haciendo.

Por último, en el mismo video 5 explique en base a lo aprendido del modelo porqué se suele decir que "el ebola, que es igual de contagioso que el COVID es mucho más fácil de contener porque la letalidad es rápida y muy alta". Qué parámetros se verían afectados? Qué implica eso en términos de la propagación del virus?

5. Suponga que tiene el modelo del punto anterior y puede elegir una de las siguientes medidas de política sanitaria:

   - Hacer obligatorio el uso de tapabocas.
   - Obligar a los agentes a una cuarentena general.
   - Vacunar al 100% de los autómatas con una vacuna china. (La vacuna no previene el contagio pero reduce los casos de gravedad en un 50%)
   - Hacer testeos aleatorios a la gente, y en caso de detectar un enfermo "E" aislarlo en Tecnópolis. (Los "G" claramente se supone que serían detectables y por lo tanto pasibles de ser aislados)

   Elija una, explique en un último video (video6) qué modificaciones debería hacer al código para simular este tipo de política, y corra el código, mostrando la misma salida que se le pidio en el punto 4.f y 4.g.