Métodos de simulación por ordenador

Modelo de propagación de una enfermedad

Alberto Cabello Álvarez
Mario Valdemaro García Roque

Para llevar a cabo este proyecto, seguimos las diretrices sobre simulación del artículo 'Introduction to Modeling and Simulation'. Llevamos a cabo las distintas etapas para este estudio que indicamos a continuación.

Paso 1, identificar el problema:

Imaginemos que somos una institución de de control sanitario como la OMS, entonces nos llega información de un nuevo virus que se esta extendiendo por una región y nuestro objetivo es averiguar como se extiende por distintas zonas.

Paso 2, formular el problema:

Dado que sabemos con que probabilidad se extiende el virus debido al contacto y como de mortal es, queremos saber como afectaría este virus a distintas poblaciones. Por tanto estudiaremos a que tipos de población afecta más el virus, y que tipos de funciones de contagio afectan de una determinada forma a la población, todo ello en función del tiempo.

Paso 3, recoger y procesar información del sistema real:

Toda la información en la que se basa este sistema ha sido generada por nosotros, no esta obtenida de sucesos reales. Por tanto no vamos a representar ningún modelo real. Sin embargo si se hiciesen los estudios suficientes sobre como de contagiosa es una enfermedad, por ejemplo el ebola, y se obtuviesen datos de como esta distribuida una población real, entonces si se estaría representando un modelo real.

Paso 4, formular y desarrollar un modelo:

Para este modelo vamos a considerar que tenemos varios individuos, esta individuos tienen un estado que va a indicar su salud, siendo los tres estados que hemos considerado sano, enfermo y muerto.

Básicamente las funciones de cambio de estado son 2, la de cambio de sano a enfermo, que dado un individuo por su posición en la tabla de población da una probabilidad de que se ponga enfermo. Esta función pude estar desarrollada de muchas formas, un ejemplo seria ver de toda la población

de infectados quien es el mas cercano y dependiendo de la distancia hasta ese infectado dar una probabilidad de infección.

La otra función de cambio de estado seria la que nos lleva del estado enfermo a muerto. Básicamente los que consideramos aquí es que dado un individuo de la población ver cuanto tiempo ha transcurrido desde que se infecto y calcular en función de este tiempo la probabilidad de que muera.

Sería posible ampliar el modelo y añadir mas estados, como por ejemplo inmune o vacunado, sin embargo no hemos desarrollado esta funcionalidad debido a que los estados que tenemos nos parecen suficientes para hacer un estudio interesante.

Las funciones de cambio de estado se producen en los siguientes casos. El cambio de estado de sano a enfermo se produce para todos los individuos sanos en cada etapa de la simulación, para ver si un individuo se ha puesto enfermo lo que hacemos es que la probabilidad que nos devuelve la función de contagio la comparamos con un numero generado aleatoriamente entre 0 y 1, si este numero es menor que la probabilidad de contagio que hemos obtenido entonces el individuo cambio de estado a enfermo guardando la información de en que etapa se ha puesto enfermo. El cambio de estado de enfermo ha muerto se produce dependiendo de cuanto tiempo lleve el individuo enfermo. Para todos los individuos enfermos en cada etapa de la simulación, la función de muerte nos dará una probabilidad de que el individuo muera que depende de cuanto tiempo lleve infectado una vez obtenida esta probabilidad hacemos lo mismo que en el caso de infección.

Paso 5, validar el modelo:

Al no usar información real no es posible que comparemos nuestras simulaciones con un sistema real. Pero si contásemos con ella podríamos comprobar si la funciones de propagación de la enfermedad son adecuadas o no.

Paso 6, documentar el modelo para uso futuro:
Como ya habíamos comentado, el objetivo es conseguir un modelo adecuado para la representación de la propagación de una enfermedad infecciosa por una población.
Consideraremos la población como una serie de puntos en un plano euclídeo, cada punto representando a un habitante. Este habitante tiene como parámetros su posición x-y en el plano y una variable indicando su estado.
Esta simulación se ejecutará a lo largo del tiempo, en cada etapa se harán los cálculos necesarios, se actualizarán las tablas y se pasará a la siguiente etapa. Estas etapas pueden corresponder a días o semanas en una situación real. Como parámetro en la ejecución se indicará el tiempo máximo sobre el que debe realizarse la simulación.

Paso 7: Elegir un diseño experimental apropiado:

Como primer paso, llevamos a cabo un generador de poblaciones adecuado. Sobre esta población se extenderá el virus, así que las componentes de densidad de población y distribución serán muy importantes. Para conseguir este objetivo, lo afrontamos de dos maneras. Primero creamos una serie de núcleos de población, que representan centros de ciudades, donde la densidad de población es mayor y puede haber más contacto por donde la enfermedad se extienda El porcentaje de habitantes en núcleos se indica como parámetro en el generador de poblaciones. El porcentaje restante no en núcleos se reparte en el resto del plano euclídeo.

Gracias a este tipo de distinciones podemos obtener información de la expansión del virus en función de la distribución espacial de la población. Veremos como la expansión resulta más rápida en las zonas más densas, además de la manera en la que la enfermedad se translada de un núcleo a otro cercano a través de los puntos intermedios.

Seguidamente desarrollaremos las funciones de contagio y muerte. Como ya comentamos, estas funciones devuelven un número entre 0 y 1, que se compara con otro número aleatorio durante la simulación. Si el resultado de la función es superior al numero aleatorio, la infección se efectúa. Realizamos varias de estas funciones según diferentes criterios, principalmente teniendo en cuenta las distancias con vecinos próximos, de acuerdo a que el objetivo es encontrar una correlación espacial y temporal.

Paso 8: Establecer condiciones experimentales para las pruebas:

Para comprobar si obtenemos la información experimental deseada, la guardamos de distintas maneras para poder visualizarlas. Guardamos tablas con la posición de un habitante y el instante de tiempo en el que queda infectado o muere. De esta manera, con una herramienta del paquete STPP animation, podemos representar como un pequeño gif animado la evolución de la población. Esto nos sirve para ver muy claramente la expansión de la enfermedad y poder buscar la función de infección más adecuada. Por otro lado, disponemos de múltiples funciones de representación gráfica para ver no solo el mapa con los puntos que representan a la población, si no para ver además los datos cumulativos.

Además, para facilitar las ejecuciones tenemos varias funciones que actúan como drivers para una batería de pruebas, realizando varias (3-5) simulaciones con los mismos parámetros y posteriormente guardando todas las tablas y datos en ficheros, de tal manera que obtenemos una gran cantidad de datos que poder comparar.

Los parámetros de entrada serán variables, pero de magnitudes en torno a 20-50 iteraciones de tiempo y poblaciones entre 2000-10000 habitantes.

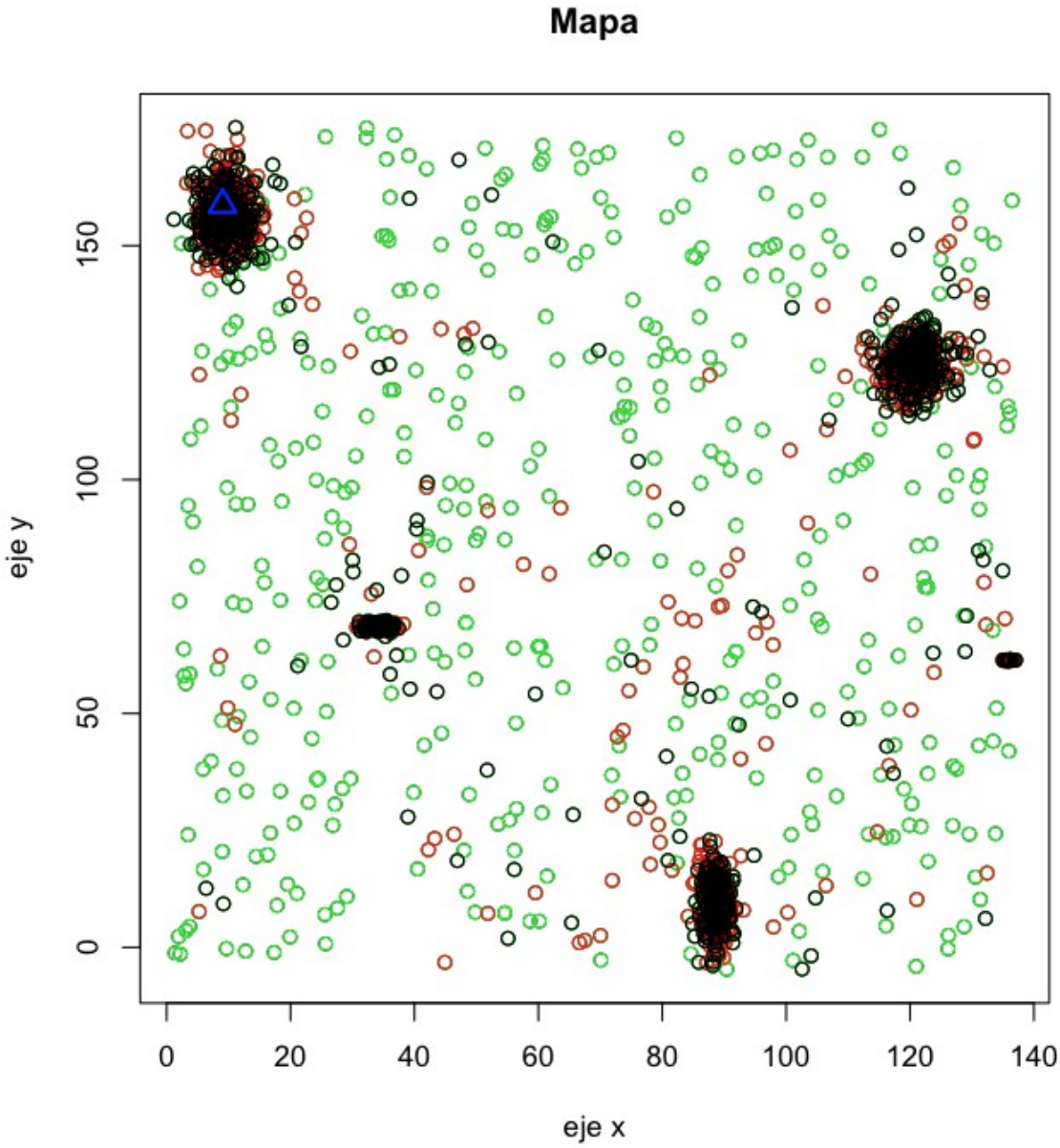
El proceso de la simulación consistirá en establecer un paciente inicial en la población, de manera aleatoria. A partir de ahí, en cada etapa evaluar las condiciones de infección y muerte para la población, cambiando los estados que correspondan.

Paso 9: Realizar la simulación:

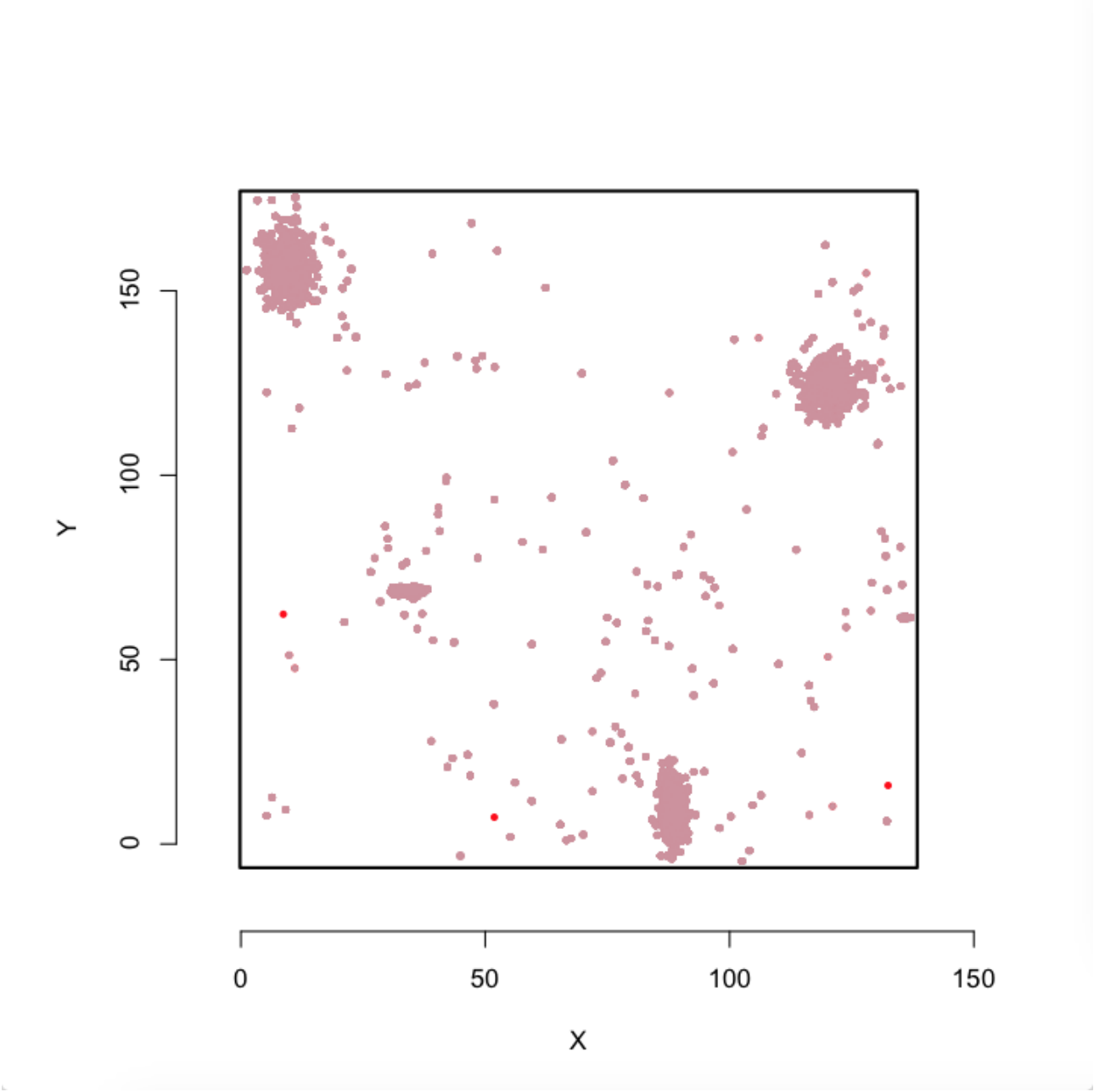
Realizamos una serie de simulaciones de acuerdo con lo indicado en el punto anterior.

Paso 10: Interpretar los resultados:

Mostramos ahora, en primer lugar, los resultados de una simulación, para mostrar el tipo de datos que obtenemos. Se trata de una población de tamaño 2000, con 5 núcleos de población, ejecutada durante 50 etapas de la simulación, utilizando una función de contagio que usa la función de densidad libre de escala siendo el valor de la alpha 1.1 y utilizando la función de densidad normal para ver si un individuo muere con una valor de la media de 1 y una varianza también de 1 . Obtenemos los siguientes resultados:

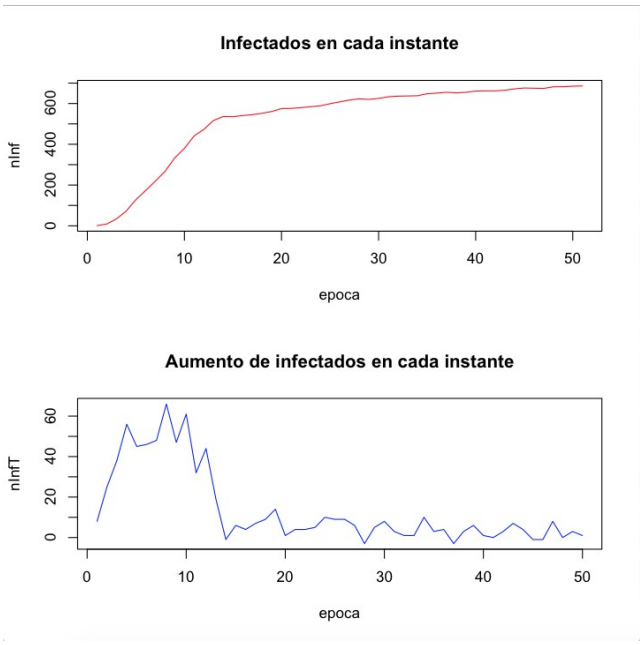


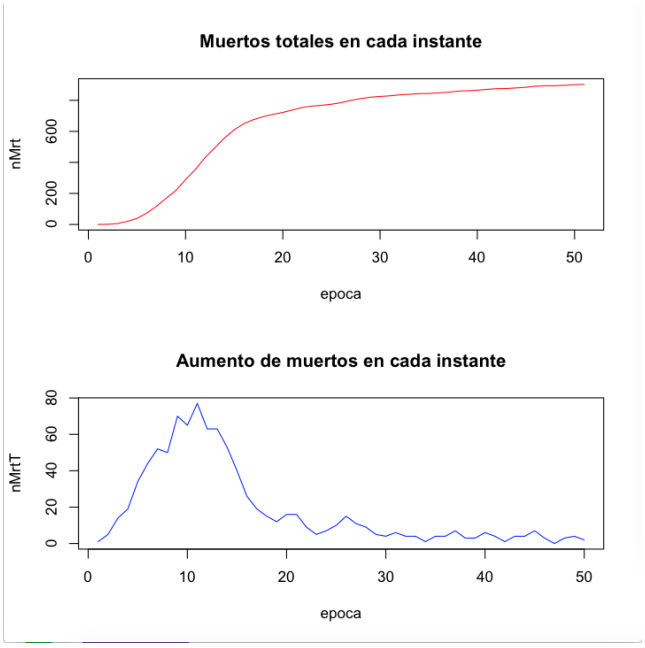
Este es el plano euclídeo que representa la región de estudio. Los puntos verdes son habitantes sanos, los rojos enfermos, y los negros, las muertes. El triángulo azul en la esquina superior izquierda representa el paciente cero, establecido de forma arbitraria. Se pude observar como los individuos que se encuentran cerca de los núcleos son mas propensos a morir o a infectarse que los que viven mas alejados de estos . Este comportamiento de transmisión se puede ver claramente en la animación. Mostramos a continuación el paso final de la animación.



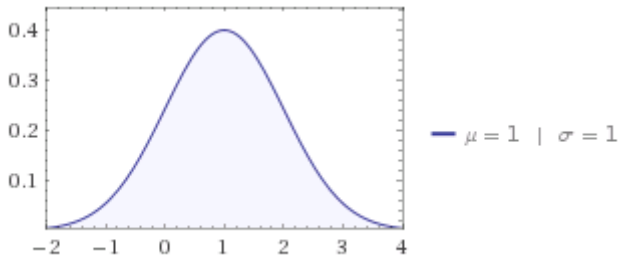
Seguidamente observamos la evolución del número de infectados y muertes. Siguen un comportamiento similar, tomando la forma de S peculiar de este tipo de procesos. Al principio evoluciona lentamente, hasta que alcanza una pendiente más pronunciada, punto de crecimiento más rápido,

como podemos observar en la gráfica inferior. El punto de más rápida infección corresponde a la etapa 7, aproximadamente, para este ejemplo, la 11 para la evolución de muertes. Estos dos procesos siguen una función similar, pero desplazada a la derecha. Al final, cuando los núcleos han sido infectados, la infección se ralentiza, ya que principalmente quedan los puntos más aislados, como comprobamos en el mapa.

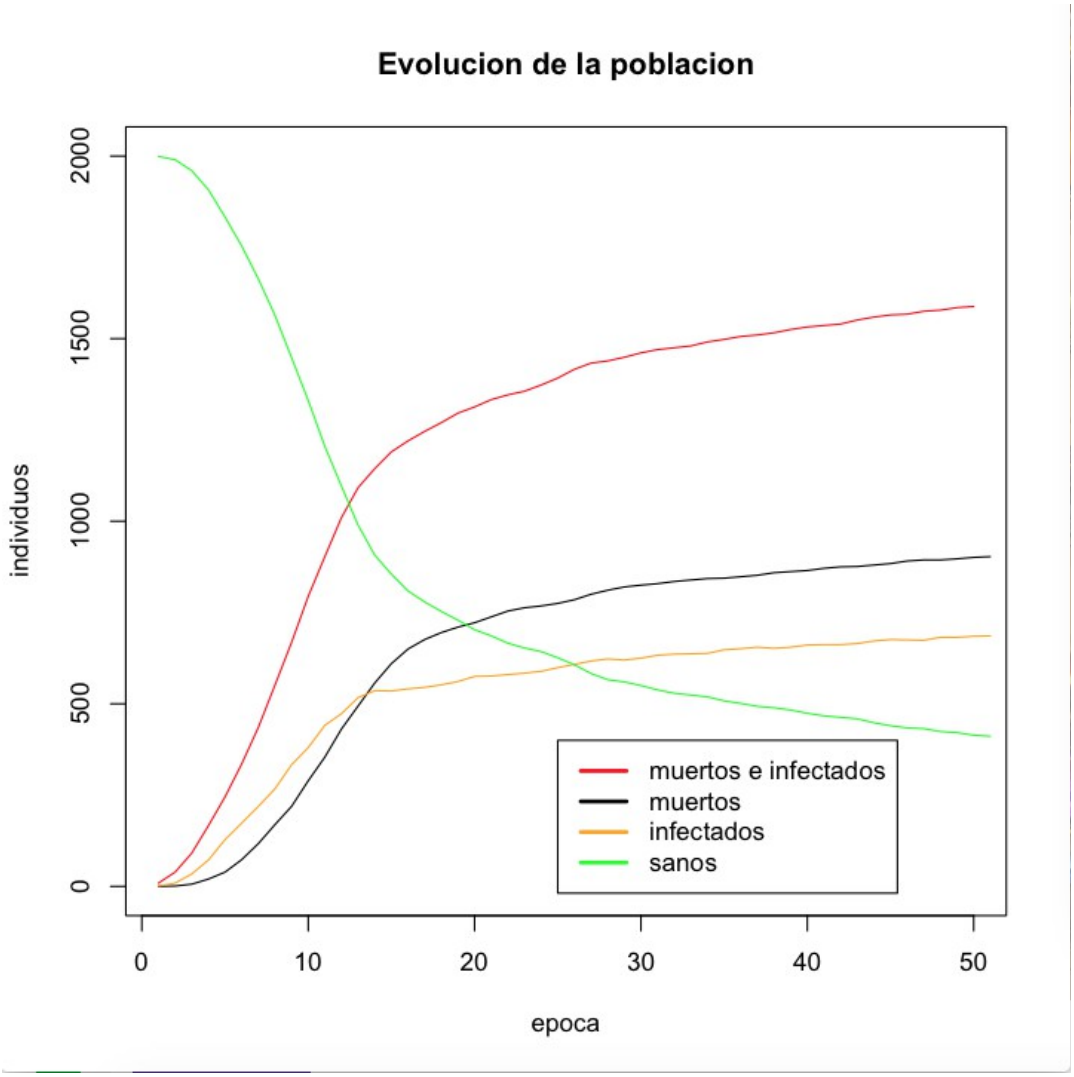




Ambas gráficas se parecen esencialmente por 2 motivos. El primero es que las muertes dependen del numero de infectados y mientras haya un numero elevado de infectados habrá un numero elevado de muertes en sucesivas iteraciones. El segundo motivo es el tipo de función que hemos usado para cambiar del estado infectado a muerto, esta función es la de densidad normal centrada en 1 y con una varianza de 1, es decir, el momento en el que mas probablemente un individuo infectado se morirá sera en la etapa siguiente, ya que si observamos esta función:



vemos que el valor con mas probabilidad es 1, es decir la etapa siguiente. Y las siguientes etapas tendrán una probabilidad menor, siendo así menos probable que un individuo que lleve mucho tiempo infectado muera. La idea que se esconde detrás de esto es representar que en cierto modo el individuo se ha “inmunizado” ante el virus pero sigue siendo un portador.



Finalmente, agrupamos los datos obtenidos en una gráfica comparativa. Vemos como desciende la población sana en función a como se extiende la enfermedad.

Lo que nos queda preguntarnos ahora es si la cantidad de muertos que observamos o la cantidad de infectados tiene que ver con alguna distribución, para ello realizamos varios test de bondad de ajuste:

La primera prueba que se nos ocurre es si el numero acumulativo de muertos tiene que ver con la distribución normal o la exponencial ya que ambas tienen un aspecto similar al que podemos observar en la gráfica superior. Por tanto realizamos la prueba de Kolmogorov sobre el porcentaje de muertos de la población total para una distribución normal y una exponencial, obteniendo los siguientes valores:

Dpnorm=0,5 Dpexp= 0,6376

Como vemos esta prueba nos indica que se parecen mas bien poco y que es bastante idóneo descartar que la muestra de muertos proceda de cualquiera de estas dos distribuciones. Sin embargo si volvemos a realizar la prueba pero esta vez en vez de contar a los muertos como el tanto por ciento de la población sino de los propios muertos obtenemos resultado diferentes.

Dpnorm=0,5 Dpexp=0,3679

Esta vez observamos que mientras que podemos seguir descartando que la distribución de muertos se parezca a una normal, el que los muertos sigan una distribución exponencial no es algo descabellado ya que se ajusta bastante.

Si realizamos el mismo test para el numero de infectados o el numero de afectados por la enfermedad obtenemos los mismo valores si comparamos con una función de distribución exponencial y valores ligeramente distintos si lo estamos comparando con una normal. Esto se debe a que si comparamos los muertos con la infectados obtenemos una valor de:

D=0,098

y si lo comparamos con los afectados obtenemos:

D=0,0651

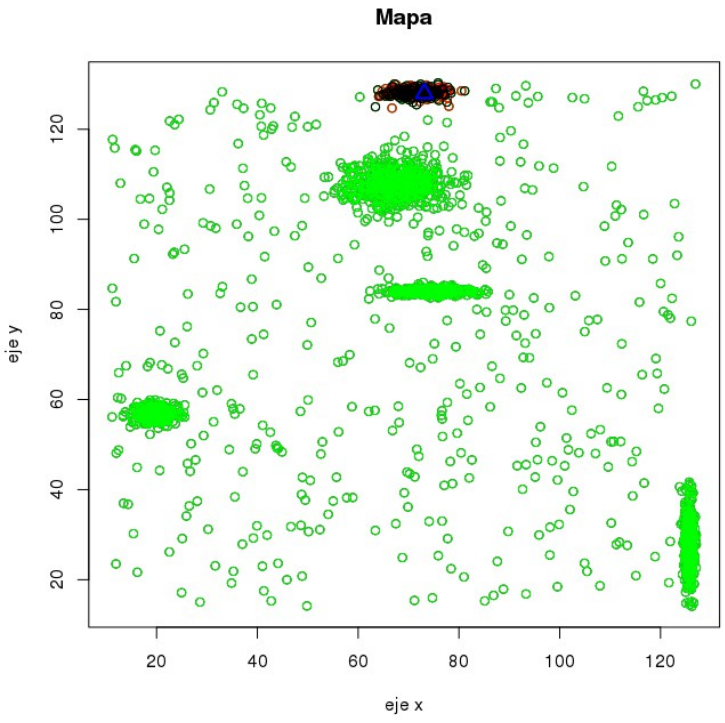
Paso 11: Recomendar pasos siguientes a tomar:

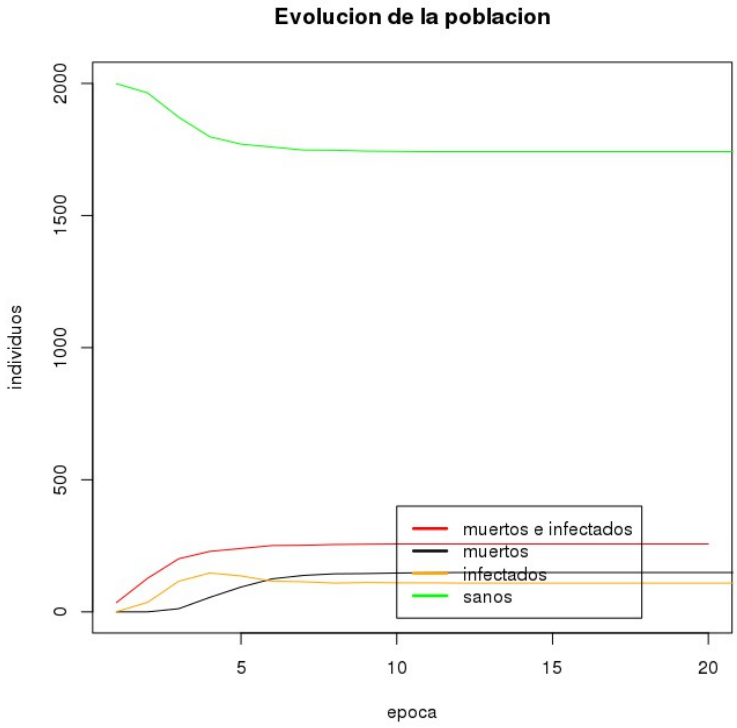
Las recomendaciones que se podrían hacer a un grupo de población que se pueda verse afectado por algunas de estas enfermedades serian básicamente minimizar el contacto físico con otras personas, mejorar las condiciones de aseo, etc. Esto se debe principalmente a que si se siguen este tipo de medidas concretas la probabilidad de infección baja ya que este suele ser el aspecto que mas condiciona la afección de un virus a una población.

Otro motivo importante para que se siguieran las recomendaciones de aseo es que estas son mas "baratas" que intentar reducir otros aspectos. Es decir es mas barato, mas sencillo y mas rápido que la gente se asee mas cuando existe un riego de contagio elevado en vez de que que se investiguen métodos para curar a los enfermos.

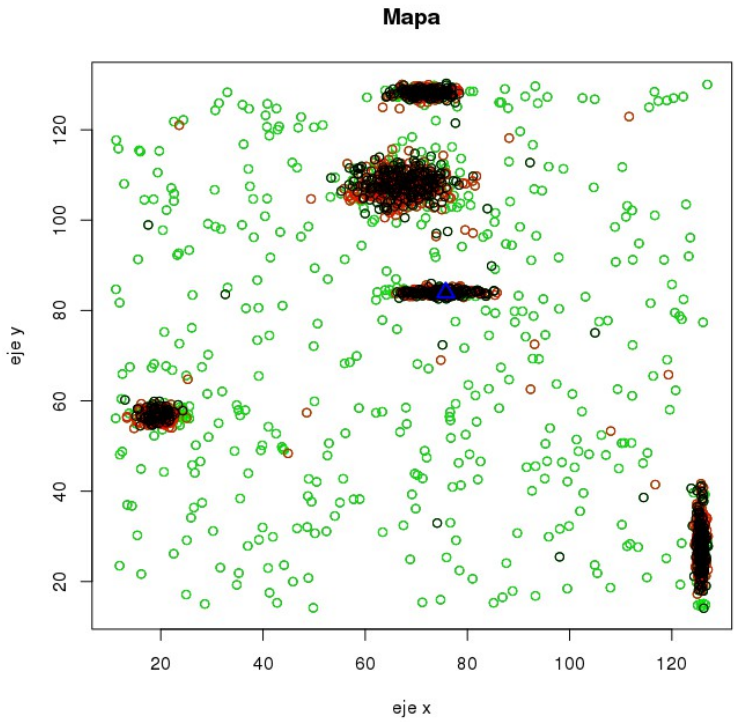
Por tanto concluimos que en un periodo inicial de la enfermedad las medias de prevención son fundamentales para reducir el impacto del virus.

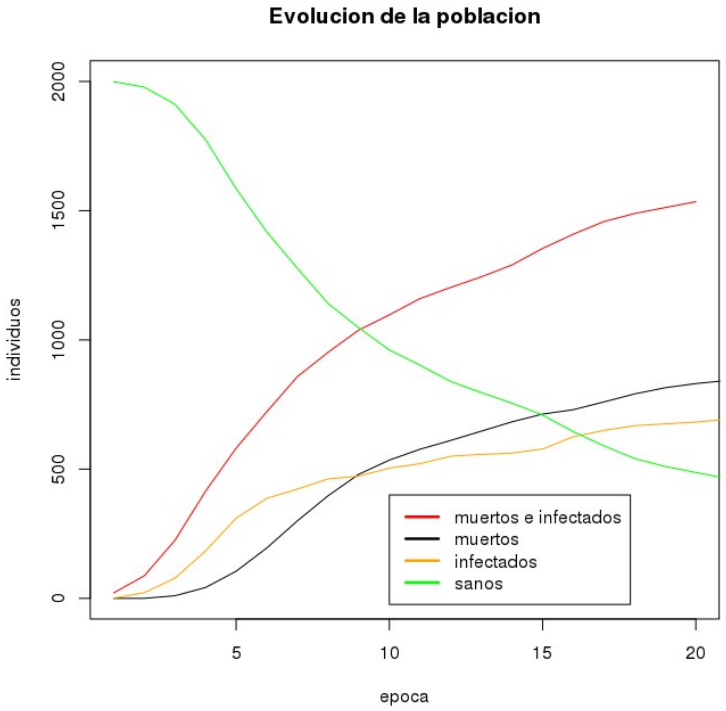
Esto lo podemos observar en los siguiente caso:



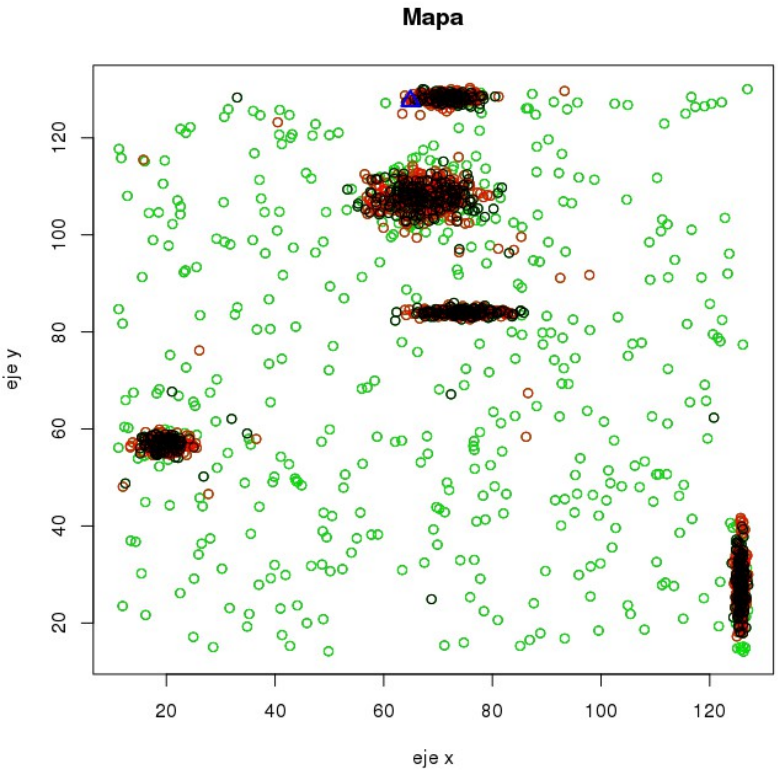


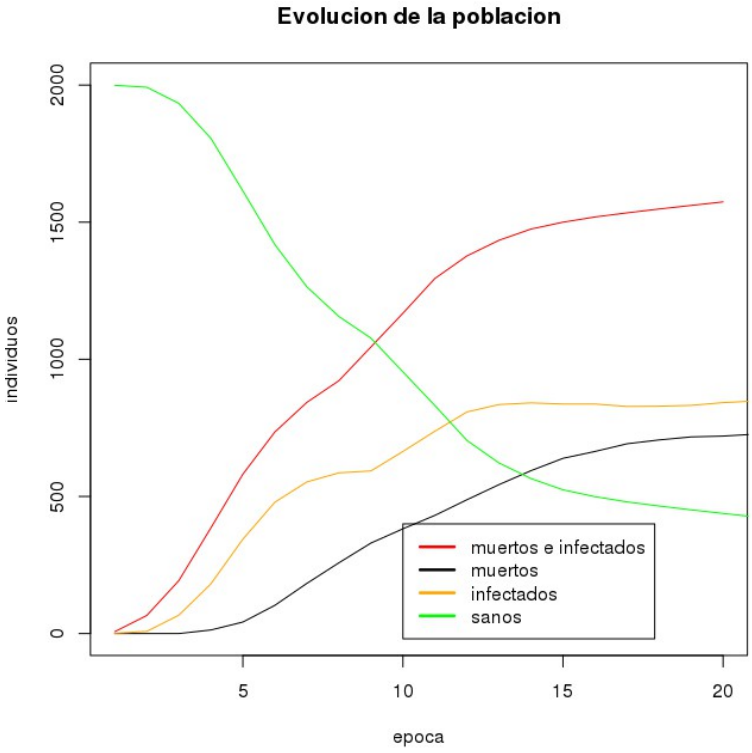
Esta prueba esta realizada con una función de infección menos agresiva.





Esta prueba es el valor de referencia.





Esta prueba usa una función de muerte menos agresiva.

Como se observa la mejora mas significativa la encontramos si usamos una función de infección menos agresiva ya que menos gente se pone enferma.

Modo de ejecución.

El código precisa de unas librerías de R para su ejecución y para mostrar los resultados. En el fichero principal se cargan las librerías con los comandos correspondientes. Estas librerías son stpp y poweRlaw, más dependencias, que deben ser instaladas.

Por otro lado, debe cargarse el fichero shared object dist.so. Debe ser compilado con el comando R CMD SHLIB dist.c, para más detalles, consultar el anexo II.

Para la lectura y escritura de resultados en ficheros, debe cargarse ficheros.r, dónde se encuentran las funciones de entrada/salida.

La función principal para la ejecución de la simulación es testpob(t, funcionInfeccion, funcionMuerte, pobl), donde se ejecuta la simulación según el proceso infeccioso y de muerte, sobre una población determinada y sobre el tiempo indicado. Se pueden introducir datos directamente ahí, obteniendo la población del generador de poblaciones, o se pueden utilizar las funciones de prueba que hemos incluido, como testPrueba() y bateriaPruebas().

Finalmente, los resultados se pueden observar con las funciones indicadas para ello, en la sección de funciones graficas, como plotGraficas y animacionInfectados.

Anexo I: Algunas cuestiones concretas sobre el generador de población:

El generador de poblaciones que hemos desarrollado funciona de la siguiente forma. Le pasamos unos datos sobre cual es la distancia máxima sobre la que pude generar individuos o núcleos de población, le decimos cuantos núcleos queremos que genere, que cantidad de individuos se encuentran en núcleos y la cantidad de población de individuos que queremos que genere.

Para generar los núcleos elegimos un punto "x" y un punto "y" aleatoriamente que sera el centro del núcleo, una desviación típica del eje "x" y otra del eje "y" para cada núcleo que generaremos mediante una distribución uniforme que tendrá como máximo el valor del logaritmo natural de la máxima distancia del eje "x" y el "y". Y una vez obtenidos estos valores asignamos una cantidad de población a cada núcleo en función de cual sea el que tiene la mayor desviación típica. A continuación generamos los individuos de cada población usando dos distribuciones normales, una para el eje "x" y otra para el "y", que nos indicara la posición de cada individuo en el mapa.

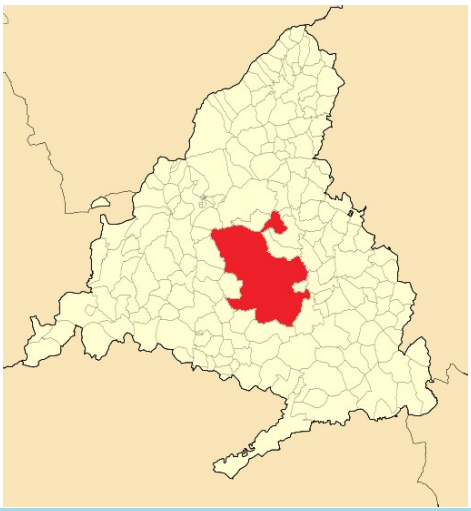
El resto de la población que no se encuentra en núcleos la generamos aleatoriamente entre los puntos máximo y mínimo de los individuos presentes en los núcleos, es decir rellenamos el espacio entre los núcleos con individuos separados aleatoriamente.

Una cuestión importante es si este tipo de generación de individuos se acerca a la realidad y a pesar de que al principio no tuvimos ningún tipo de estudio sobre la distribución de la población como referencia, pasado cierto tiempo alguno de estos estudios se nos puso delante y no se aleja demasiado de lo que hemos desarrollado.

"*El 54 por ciento de la población mundial actual reside en áreas urbanas y se prevé que para 2050 llegará al 66 por ciento, según datos de un informe de la ONU difundido este jueves.*"
"*Esas megaciudades acogen en conjunto a 453 millones de personas, o un 12% de la población urbana mundial. "*
https://www.un.org/development/desa/es/news/population/world-urbanization-prospects-2014.html
Si nos vamos concretamente al caso de la población española observamos lo siguiente:

"La zona roja representa la mitad de la población de España, igual que la zona azul. La mitad de la población española vive en 125 de los más de ocho mil municipios constituidos. Dato a tener en cuenta: Casi cinco mil ayuntamientos (60% del total) cuentan con

menos de mil habitantes, y nada menos que mil (13% del total) gobiernan a menos de cien personas.“



“La mitad de la población de Madrid vive en Madrid”

“El 50% de la población reside en el área roja, cuyos extremos son Tarrasa, Castelldefels y Mataró. Fuera de esa zona sólo hay cinco localidades con más de 50.000 habitantes: las otras tres capitales de provincia, Reus y Manresa. No es la única región española con una distribución demográfica muy centralizada, claro.”

http://fronterasblog.com/2016/01/14/veinte-mapas-sobre-la-distribucion-de-la-poblacion-en-espana/

Anexo II: Uso de C en R.

Al realizar este trabajo, nos encontramos con un problema de eficiencia. Al manejar poblaciones relativamente grandes y realizar cálculos sobre ellas, decidimos hacer un cálculo previo de las distancias entre cada habitante antes de la simulación en sí, para ahorrar tiempo a la larga.
Inicialmente realizando esa tarea en R, resultaba glacialmente lento, así que decidimos implementar parte del código en C. Según ciertos benchmarks que consultamos, en el ejemplo de resolver cierto número de sudokus, C es del orden de 1000x más rapido que R en este problema. La ventaja de R, no reside en su eficacia, claramente, si no en la multitud de herramientas y paquetes disponibles.

Para este fin, en primer lugar, se crean las funciones de C, que serán invocadas desde R. Estas funciones siguen algunas particularidades en cuanto a argumentos y operaciones permitidas. Los argumentos deben ser punteros simples, de tipos compatibles con R, como int* y double*. Las matrices de dos dimensiones no son compatibles de manera directa, pero se puede utilizar indexación de 1 dimensión como equivalente a 2 con las operaciones correspondientes.

Una vez escrito el código de la función en C, es recomendable crear un wrapper en R que invoque a la función con los argumentos correspondientes, para facilitar el uso.

Para compilar y utilizarlo desde R, se utiliza el comando

```
R CMD SHLIB dist.c
```

Que crea un archivo .so (shared object), que se puede cargar a R con

```
dyn.load("dist.so")
```

El uso de código C en la parte de precálculo de la tabla de distancias salva una cantidad de tiempo sustancial en esta tarea, así como en general al poder usar esta tabla ya calculada en el resto de la simulación.