Tarea 1
Simulación de Sistemas

Beatriz Alejandra García Ramos

A 14 de Agosto de 2017

1 Movimiento Browniano

El Movimiento Browniano se refiere a una partícula cambiando su posición uniformemente al azar. Los movimientos pueden ser de muchos distintos tipos, pero en esta práctica nos limitamos a un caso sencillo donde la particula se mueve en pasos discretos, es decir, cada paso mide lo mismo, y las únicas posibles direcciones de movimiento son las direcciones paralelas a los ejes cardinales del sistema de coordenadas en el cual se realiza el movimiento. Vamos a utilizar pasos unitarios (es decir, el paso mide uno), teniendo como la posición inicial de la partícula el origen.

2 Tarea

- Examinar de manera sistemática los efectos de la dimensión en el número de veces que la caminata regresa al origen durante una caminata del movimiento Browniano.
- Verificar que el número de pasos de la caminata o el número de repeticiones del experimento no estén causando un efecto significativo
- Estudiar de forma sistemática y automatizada el tiempo de ejecución de una caminata en términos del largo de la caminata y la dimensión.
- Realizar una comparación entre una implementación paralela y otra versión que no aproveche paralelismo.

3 Solución

Al interpretar el código me dí cuenta de que se ejecutan las caminatas, se obtiene la distancia de cada caminata de forma Euclidiana o Manhattan y se grafica la distancia máxima de cada repetición dependiendo de la dimensión del vector inicial que va de 1 a 8. El reto fue averiguar cuántas veces se pasa por el origen de nuevo en la caminata que se realiza, observé que algunas cosas del código no se necesitaban y otras debían incorporarse para que se hiciera el conteo de las ocasiones en que volviera al origen, creé una variable llamada origen y si el vector pos (posición) era igual que el origen entonces se agrega 1 al vector suma que va agregando las veces en que se vuelve a pasar por ahí, al final la suma nos indica el máximo de veces en que se pasa por el origen y se grafica en una *boxplot* la distancia con respecto a la dimensión dada y se generan las repeticiones indicadas.

Para el primer reto estuve investigando sobre las funciones que se pueden utilizar en R para evaluar el tiempo en que se ejecuta un código y lo que encontré fue que Sys.time() cuenta el tiempo desde que se abre el programa R hasta la primera vez que se lee en el código y lo guarda, lo mismo sucede si se lee al final del código, por lo cual al hacer la diferencia del fin y del inicio se puede obtener el tiempo exacto de ejecución del código y se imprime en la consola.

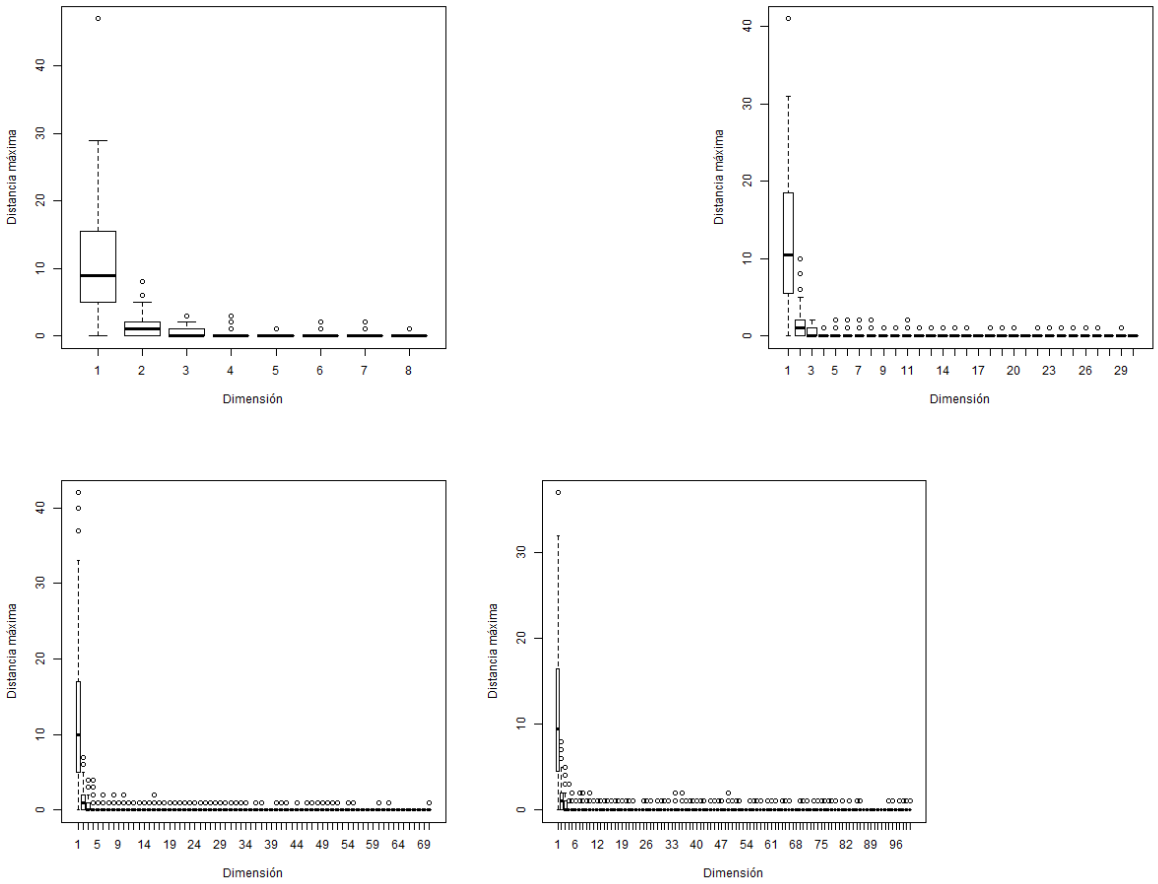

Figure 1: Variación de dimensiones a 8, 30, 70 y 100 respectivamente

Para ver lo que sucede cuando no se utiliza la implementación paralela se necesita cambiar la funcion pasSapply por sapply y no es necesario tener los cluster.

Los códigos que utilicé se encontrarán adjuntos en GitHub.

3.1 Examinación de los efectos de la dimensión.

Cuando en el código se cambia la dimensión que pueden tomar los distintos vectores y se analiza con una gráfica *boxplot* podemos observar que conforme la dimension va aumentando la suma promedio de las veces que pasó por el origen va disminuyendo y aún cuando se corre con distintos tamaños de dimensiones el promedio se mantiene cercano a 10 y va en caída como en el caso antes mencionado, como podemos observar en las gráficas de la Figura 1 cuando la dimensión es de 1 la media se aproxima a 10, cuando la dimensión es 2 la media se aproxima a 1 y se sigue en caída; pero podemos observar también que la cantidad máxima de veces que se pasó por el origen se aproxima a 30 aún cuando las dimensiones son variadas.

3.2 Examinación de los efectos de los pasos

En el caso en el que se cambia la cantidad de pasos que podría llegar a hacer nuestro movimiento Browniano nuestro promedio cambia y se acerca a 15 veces en que pasa por el origen mientras que la

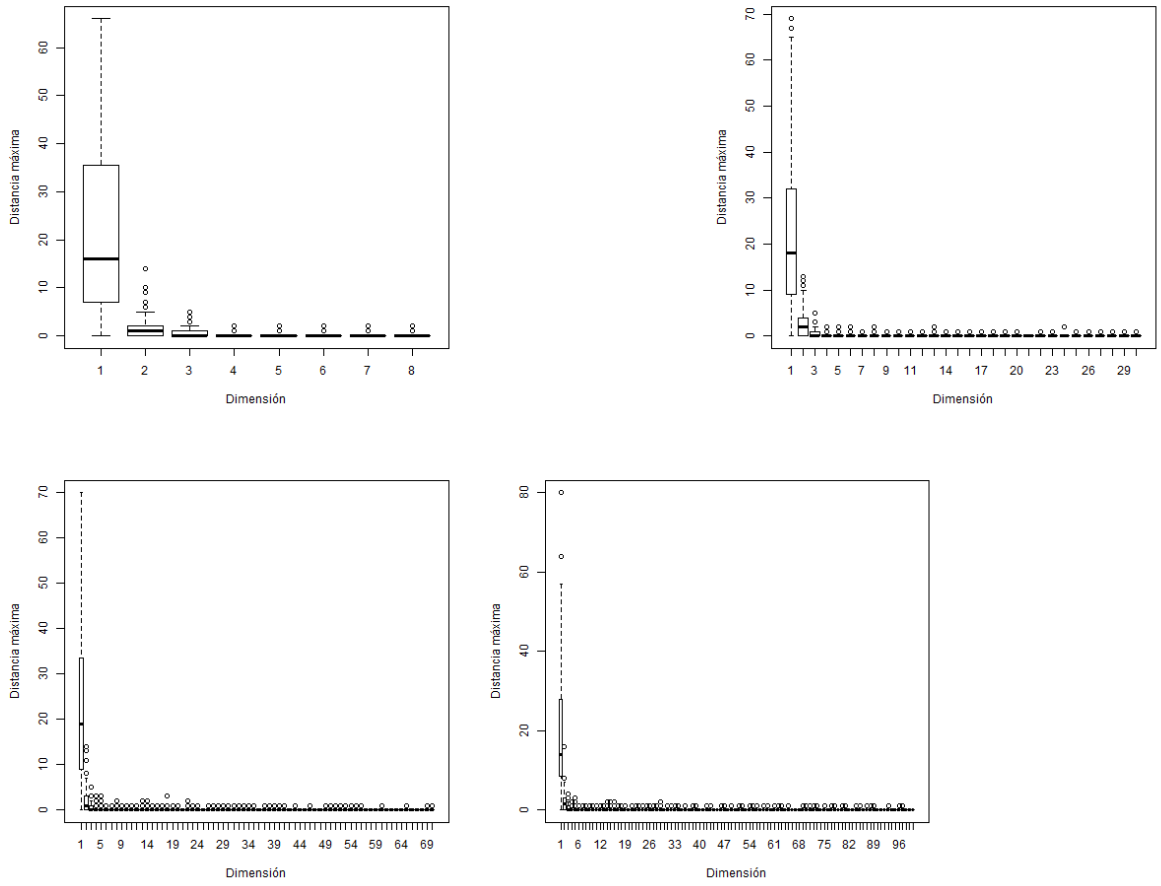

Figure 2: Variación a 800 pasos con dimensiones 8, 30, 70 y 100 respectivamente

cantidad máxima en las que pasa por el origen se encuentra entre 60 y 70 veces. Esto nos dice que entre más pasos pueda dar más veces es posible que pase por el origen, aunque siguen una trayectoria en caída nuestras gráficas como se muestran en la Figura 2, es decir, mienrtras mayor sea nuestra dimensión, menor será nuestra cantidad de veces que se pasa por el origen como sucedió anteriormente.

3.3 Examinación de los efectos de las repeticiones.

En el caso en el que la cantidad de repeticiones cambia podemos observar que varía solamente un poco con el experimento de cambiar las dimensiones, en lo que "afecta" es en que en lugar de que el promedio se mantenga en 10, inicia casi en 10 y va disminuyendo un poco dependiendo de la cantidad de dimensiones que se tengan, mientras que la cantidad máxima de veces en las que se pasa por el origen va de 29 a 35, lo cual nos indica un valor muy similar al cambiar sólo las dimensiones (Sección 3.1).

Conclusiones

Cuando hacen modificaciones con respecto a las dimensiones o a las repeticiones que se indican de entrada en el código no hay diferencia sobre la obtención de la cantidad de veces que se pasa por el origen, caso contrario en el cambio de pasos, dado que mientras más pasos se puedan ingresar mayor es la cantidad de veces en las que se podría pasar por el origen.

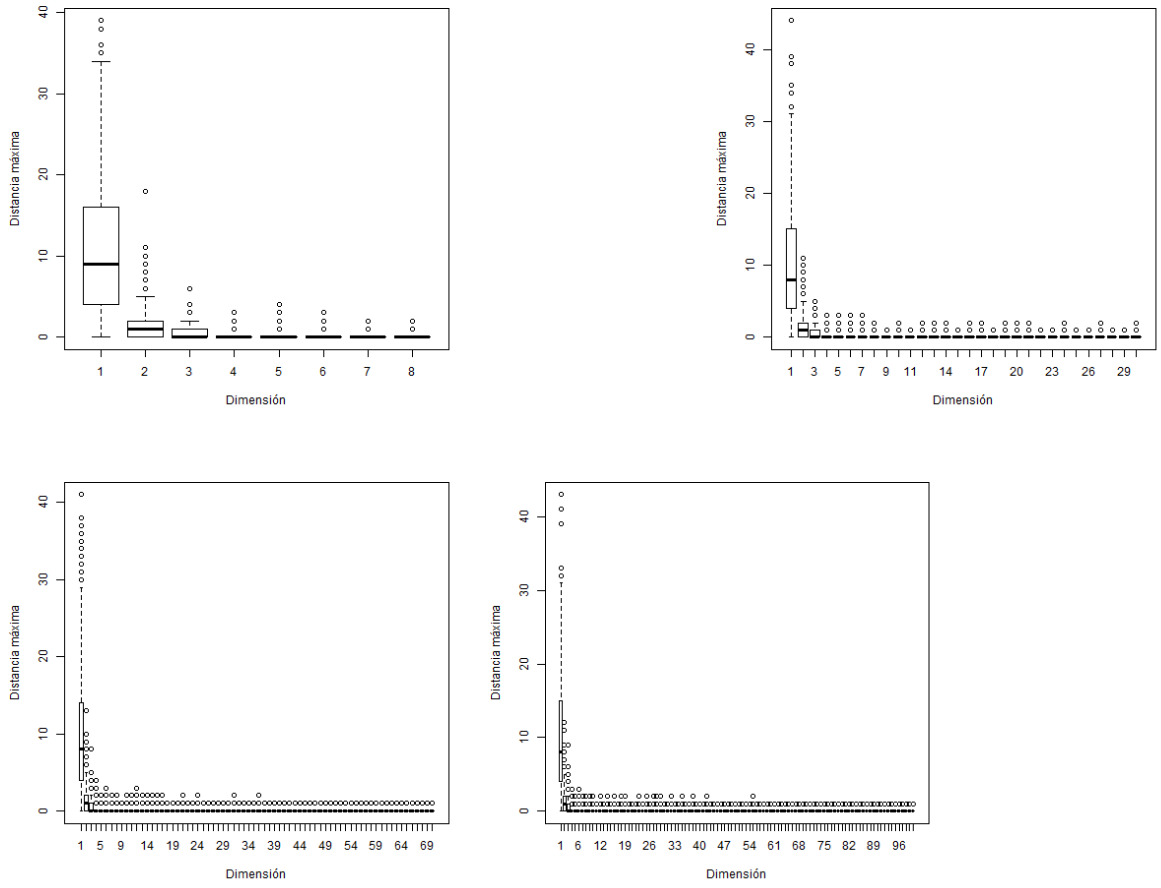

Figure 3: Variación a 800 repeticiones con dimensión 8, 30, 70 y 100 respectivamente

Analizando los datos, conforme modificamos la cantidad de pasos que se pueden dar en nuestro movimiento podremos decir si serán pocas o muchas veces las ocasiones en las que se vuelve al origen, esto se debe a que si la cantidad de pasos que le permitamos dar es pequeña podemos decir que la cantidad de veces en que pasará por el origem también lo será, y de la misma manera ocurrirá cuando la cantidad de pasos que le permitamos dar sea grande, ya que mientras más pasos se pueda dar, podrá avanzar y retroceder con libertad en nuestro plano, lo cual le permitirá al código sin ningún problema volver una y otra vez al origen hasta que la cantidad de pasos que le queden sea pequeña y deba evitar que se acaben antes de llegar a nuestro límite.

3.4 Examinación del tiempo de ejecución. Reto 1.

Para poder comparar el tiempo de ejecución con respecto al largo de la caminata y la dimensión utilicé 4 distintas repeticiones con las 4 distintas dimensiones que había estado manejando para la parte anterior de la Tarea 1. Es evidente que conforme aumentamos la medida de nuestra dimensión aún teniendo las mismas repeticiones el tiempo de ejecución aumentará ya que entre mayor sea la dimensión más caminos distintos se deben verificar.
Sin embargo, conforme las repeticiones van aumentando el tiempo de ejecución también aumenta. Podemos observar en la tabla que no existe una diferencia significativa si observamos en cada cantidad de repeticiones las distintas dimensiones que se manejan, pero si vemos el tiempo de ejecución de cada dimensión con las distintas cantidades de repeticiones podemos notar que existe una gran diferencia entre una y otra, más cuando la dimensión es grande. Puede ser que de 1 a 4 minutos no se "vea" problema, pero tomamos en cuenta que es un código pequeño; podriamos llegar a tener alguno con un tiempo mucho mayor de ejecución y de 20 a 50 minutos por decir un ejemplo se podría generar una gran dificultad.

	Repetición 100	Repetición 800	Repetición 1200	Repetición 2000
Dimensión 8	2.289337secs	10.12318secs	13.57148secs	23.25646secs
Dimensión 30	5.765527secs	32.61009secs	47.7492secs	1.331601 mins
Dimensión 70	11.42293 secs	1.284031 mins	1.886746 mins	3.126801 mins
Dimensión 100	16.01836 secs	1.97959 mins	2.767622 mins	4.698927 mins

3.5 Examinación de implementación paralela y no paralela. Reto 2.

Como se quiere mostrar, la utilización de *programación paralela* nos ayuda a repatir el trabajo entre los núcleos con el propósito de disminuir el tiempo de reloj requerido para llevar a cabo la tarea en cuestión. Claramente en la tabla observamos que es verdad, cuando descartamos la implementación paralela nos encontramos con que el tiempo de ejecución aumenta (hasta diez veces o más en este caso), lo cual es menos conveniente al momento de hacer la ejecución de un código porque en ocasiones se tiene "el tiempo encima" para hacer algún programa.

	Implementación paralela	No implementación paralela
Dimensión 8	2.289337 secs	2.412464 secs
Dimensión 30	5.765527 secs	7.845728 secs
Dimensión 70	11.42293 secs	21.10429 secs
Dimensión 100	16.01836 secs	28.3924 secs