Práctica 1
Movimiento Browniano
Delavy Duque Martínez
13 de agosto de 2017

Introducción:

El movimiento browniano se refiere a una partícula que cambia su posición uniformemente al azar. Los movimientos que esta partícula puede realizar. En esta práctica, se simuló el movimiento browniano en pasos discretos, o sea, que se incrementaban uniformemente, y se decidió incrementarlo en una unidad, teniendo como posición inicial el origen.

Hipótesis

Entre más dimensiones se mueva la partícula, será más difícil que regrese al origen.

Simulación

Para realizar esta simulación, se tomó en cuenta una función en el software R para hacer valores aleatorios *runif(),* con pasos aleatorios unitarios incrementados uniformemente. Los números aleatorios que se generen tendrán valores entre el 0 y el 1.

La partícula avanza o retrocede dependiendo del valor de runif, si es más de 0.5 avanza 1 unidad, y si es menor de 0.5, retrocede una unidad. Pimero se simuló en una dimensión, con pocos pasos y pocas repeticiones de este experimento, dando resultados diferentes cada vez que se ejecutaba. Al comprobar que funcionaba, se agregaron hasta 8 dimensiones para simular cuantas veces regresa esa partícula al origen.

Utilizamos un ciclo para ir recorriendo las pruebas en pasos y las repeticiones ya establecidas, y al final, las graficamos para ver mejor los cambios.

Resultados

Los resultados se muestran en las gráficas siguientes.

En la primer gráfica, se muestra una simulación de movimiento browniano, con 100 repeticiones y 200 pasos, para probar diferentes repeticiones y pasos, verificando que no haya diferencia significativa si se cambian los números de repeticiones y de pasos.

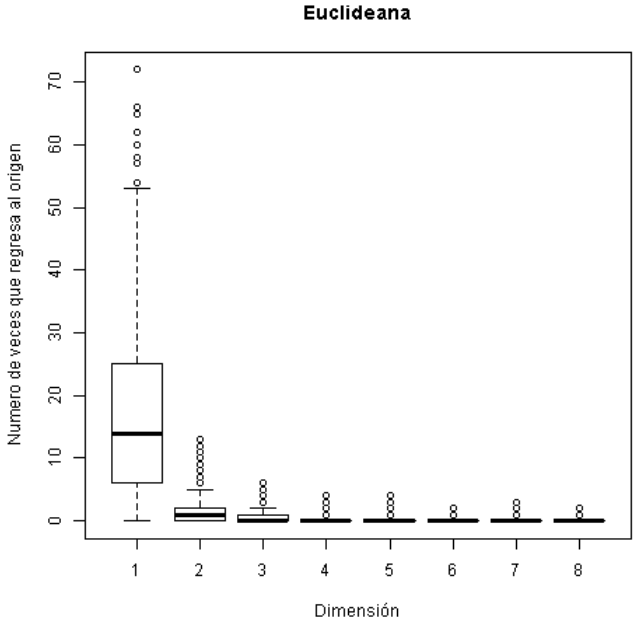

Ilustración 1 - Experimento repetido 100 veces en 200 pasos

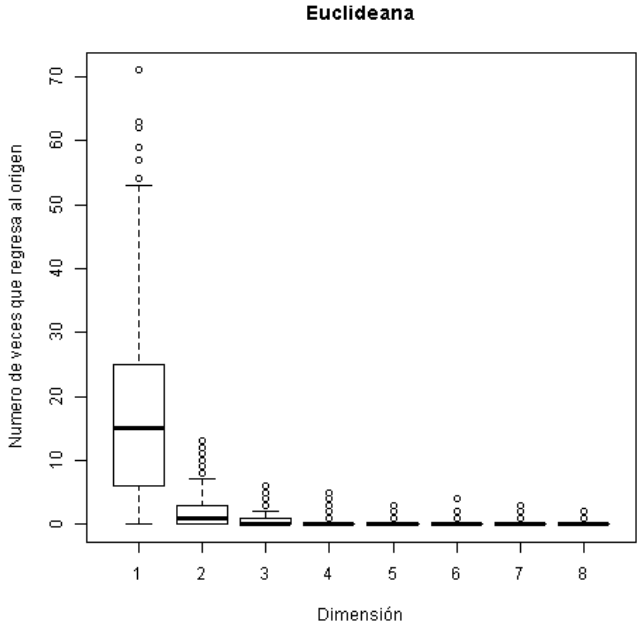

Ilustración 2 - Experimento repetido 1000 veces en 500 pasos

En otra parte de la simulación, el enfoque era del tiempo de ejecución realizado por el procesador para ejecutar la simulación en todas las dimensiones. Los resultados se pueden ver en las gráficas siguientes, el proceso en paralelo y cuando no es en paralelo. Se cambió la función de *parSapply* que ejecuta procesos en paralelo, por *sapply,* que no ejecuta procesos en paralelo.

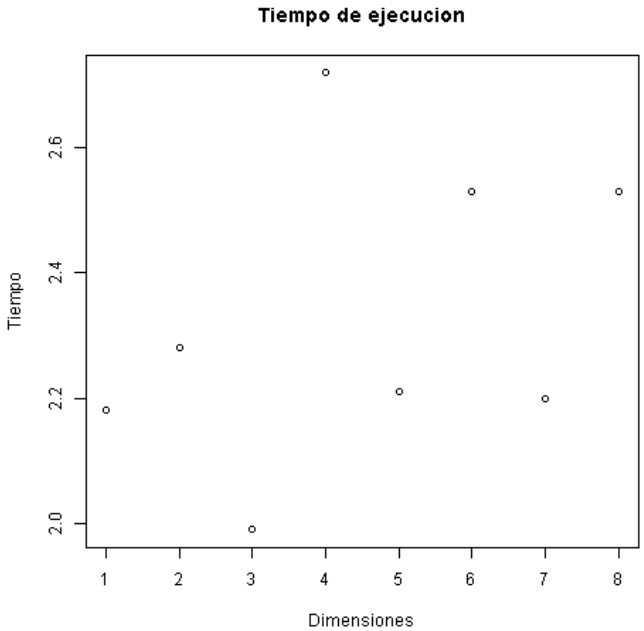

Ilustración 4 - Tiempo de ejecución con paralelismo (parSapply)

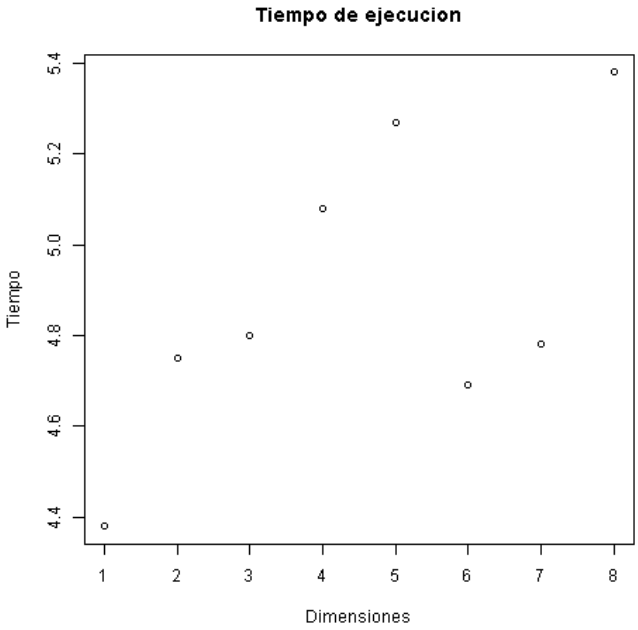

Ilustración 3 - Tiempo de ejecución sin Paralelismo (sapply)

Conclusiones

Es más probable que la partícula regrese al origen si tiene menos dimensiones, comprobando la hipótesis propuesta, sin gran diferencia si se repite esta simulación más veces o se ponen más pasos. El tiempo de ejecución paralela es mucho menor que cuando no se hace en paralelo, ya que cuando no se hace en paralelo, invierte hasta casi el doble de tiempo de lo que le cuesta al proceso hacerse en paralelo.

Referencias

http://elisa.dyndns-web.com/teaching/comp/par/p1.html

http://gforge.se/2015/02/how-to-go-parallel-in-r-basics-tips/#Using_parSapply

Código

```
library(parallel)
repetir <- 1000
pasos <- 500
datos <-  data.frame()
datosPD <- data.frame() #datos de Pasos y Dimensiones
datosNP <- data.frame() #datos sin Paralelismo

experimento <- function(replica) {
  pos <- rep(0, dimension)
  origen <- rep(0, dimension)
  times <-0
  for (t in 1:pasos) {
    cambiar <- sample(1:dimension, 1)
    cambio <- 1
    if (runif(1) < 0.5) {
        cambio <- -1
    }
    pos[cambiar] <- pos[cambiar] + cambio
```

```
    if (all(pos == origen)) {
      times <- times + 1
    }
  }
  return(times)
}

cluster <- makeCluster(detectCores() - 1)
clusterExport(cluster, "pasos")

for (dimension in 1:8) {
    clusterExport(cluster, "dimension")
    resultado <- parSapply(cluster, 1:repetir, experimento)
    M <-system.time(parSapply(cluster, 1:repetir,
experimento))[3] #Con Paralelismo y revisando el tiempo
    N <-system.time(sapply(1:repetir, experimento))[3] #Sin
paralelismo
    datos <- rbind(datos, resultado)
    datosPD <- rbind(datosPD, M)
    datosNP <- rbind(datosNP, N)
}

stopCluster(cluster)
png("p1er.png")
boxplot(data.matrix(datos), use.cols=FALSE,
xlab="Dimensi\u{F3}n", ylab="Numero de veces que regresa al
origen", main="Euclideana")
graphics.off()
```

```
png("systemTime.png")

plot(data.matrix(datosPD),

xlab="Dimensiones", ylab="Tiempo", main="Tiempo de
ejecucion")

graphics.off()


png("notParalel.png")

plot(data.matrix(datosNP),

xlab="Dimensiones", ylab="Tiempo", main="Tiempo de
ejecucion")

graphics.off()
```