

# Práctica 1: movimiento Browniano

Examina 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. Recuerda verificar que el número de pasos de la caminata o el número de repeticiones del experimento no estén causando un efecto significativo.

Lo primero que hay que realizar es comprender el programa para realizar el movimiento browniano, como como el primer ejemplo que nos da el problema es realizar una variable que estará iniciándose en una duración de tiempo prevista y cambiando con una rutina en un número aleatorio, incrementándose o decrementándose dependiendo de la condición.

```
pos <- 0
runif(1)
dur <- 10
for (t in 1:dur) {
  if (runif(1) < 0.5) {
    pos <- pos + 1
  } else {
    pos <- pos - 1
  }
  print(pos)
}
```

Tomando como punto de partida el ejemplo que nos proporcionan lo realizaremos primeramente en una dimensión, esta vez no se realizara la distancia mayor si no que observaremos cuantas veces pasa la variable que creamos por el origen, esto es posible creándole una igualdad con la variables ya dicha y el origen previamente denominado en un inicio todo esa dentro de un condicional (es posible dejarle los valores de mayor y las instrucciones no es necesario borrarlas, yo las borre para no generar confusión)

```
pos <- 0
salida<-0
origen <- c(0)
dur <- 100
for (t in 1:dur) {
  if (runif(1) < 0.5) {
    pos <- pos + 1
  } else {
    pos <- pos - 1
  }
  if(pos == origen){
    salida <- salida + 1
    print(pos)
  }
}
print(salida)
}
```

Como se puede observar en seguida, los resultados del programa que pasa por el origen en unadimension, se agregaron dos print; uno para imprimir la cantidad de ceros (print(pos)) y el otro para imprimir la cantidad de veces que pasa por el origen (print(salida)) .

```
R Console

> pos <- 0
> salida<-0
> origen <- c(0)
> dur <- 100
> for (t in 1:dur) {
+   if (runif(1) < 0.5) {
+     pos <- pos + 1
+   } else {
+     pos <- pos - 1
+   }
+   if(pos == origen){
+     salida <- salida + 1
+     print(pos)
+   }
+ }
[1] 0
[1] 0
[1] 0
[1] 0
[1] 0
[1] 0
[1] 0
[1] 0
[1] 0
[1] 0
[1] 0
[1] 0
[1] 0
[1] 0
[1] 0
[1] 0
[1] 0
[1] 0
> print(salida)
[1] 15
> |

<
```

Aquí se puede observar el resultado en dos dimensiones tomando como partida dimensión de 2 y duración de 20

```
> dim <- 2
> dur <- 20
> pos <- rep(0, dim)
> origen <- c(0,0)
> contador <- 0
> for (t in 1:dur) {
+   cambiar <- sample(1:dim, 1)
+   cambio <- 1
+   if (runif(1) < 0.5) {
+     cambio <- -1
+   }
+   pos[cambiar] <- pos[cambiar] + cambio
+   d <- dist(pos)
+   print(pos)
+   if (pos[1] == origen[1] & pos[2] == origen[2]) {
+     contador <- contador + 1
+   }
+ }
[1] 1 0
[1] 0 0
[1] 1 0
[1] 1 1
[1] 1 2
[1] 1 3
[1] 1 2
[1] 1 1
[1] 2 1
```

```

[1] 3 1
[1] 2 1
[1] 2 2
[1] 2 1
[1] 3 1
[1] 3 2
[1] 2 2
[1] 3 2
[1] 3 3
[1] 3 4
[1] 4 4
>
>
> print(contador)
[1] 1
>

```

Enseguida realizaremos en dos dimensiones, en este apartado necesitaremos mandar un archivo llamado distancia.R proporcionado por el problema, este nos ayudara saber el tipo de movimiento que realizara en dos dimensiones

```

euclidean <- function(p1, p2) {
  return(sqrt(sum((p1 - p2)**2)))
}

manhattan <- function(p1, p2) {
  return(sum(abs(p1 - p2)))
}

ed.orig <- function(p) {
  dimension <- length(p)
  origen <- rep(0, dimension)
  return(euclidean(p, origen))
}

md.orig <- function(p) {
  dimension <- length(p)
  origen <- rep(0, dimension)
  return(manhattan(p, origen))
}

```

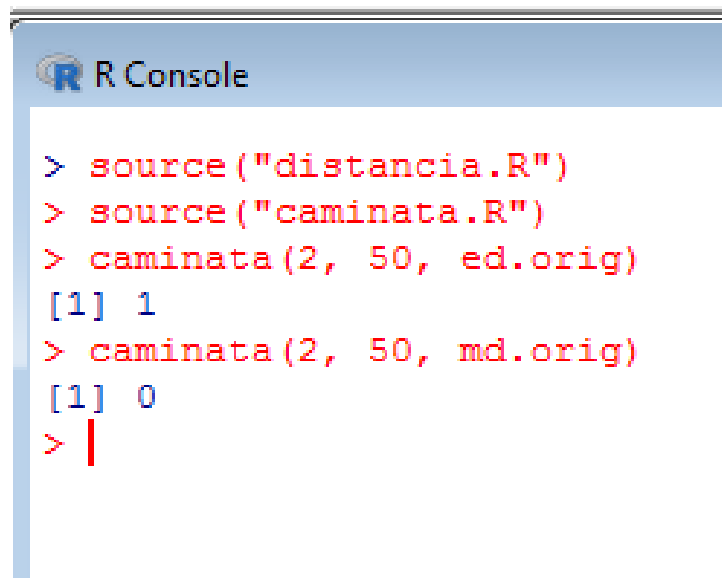
```
}
```

Tomando en cuenta el archivo del ejemplo tomaremos la estructura y lo llamaremos caminata.R, se le realizaron las modificaciones para saber las veces que pasa por el origen que coincidan en ambas dimensiones

```
caminata <- function(dim, dur, dist) {  
  pos <- rep(0, dim)  
  origen <- rep(0, dim)  
  contador <- 0  
  for (t in 1:dur) {  
    cambiar <- sample(1:dim, 1)  
    cambio <- 1  
    if (runif(1) < 0.5) {  
      cambio <- -1  
    }  
    pos[cambiar] <- pos[cambiar] + cambio  
    d <- dist(pos)  
    if (pos[1] == origen[1] & pos[2] == origen[2]) {  
      contador <- contador + 1  
    }  
  }  
  return(contador)  
}
```

Probaremos el programa como se muestra en el ejemplo usando las mismas instrucciones

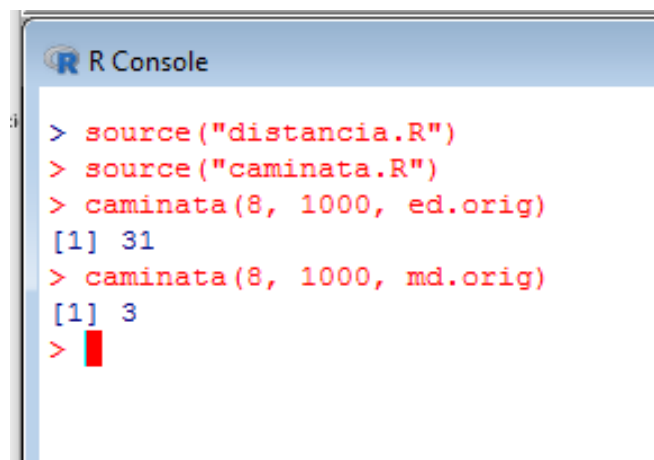
```
source("distancia.R")
source("caminata.R")
caminata(2, 50, ed.orig)
caminata(2, 50, md.orig)
```



```
R Console

> source("distancia.R")
> source("caminata.R")
> caminata(2, 50, ed.orig)
[1] 1
> caminata(2, 50, md.orig)
[1] 0
> |
```

También podemos realizarse con más dimensiones y mayor número de duración para comprobar que lo realice en mayor número de dimensiones



```
R Console

> source("distancia.R")
> source("caminata.R")
> caminata(8, 1000, ed.orig)
[1] 31
> caminata(8, 1000, md.orig)
[1] 3
> |
```