

TAREA # 1

Movimiento Browniano

Natalia Berenice Pérez López

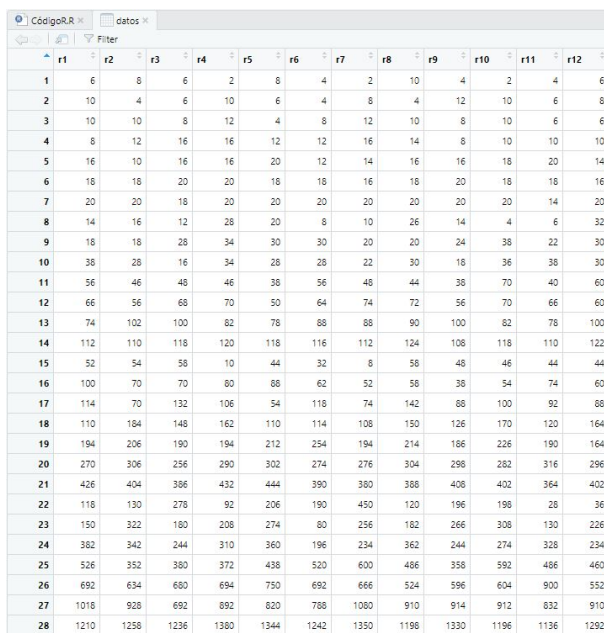
1 de septiembre de 2021

1. Objetivo

El objetivo de esta práctica es examinar los efectos de la dimensión en la distancia Manhattan del Movimiento Browniano de una partícula para varias dimensiones (en incrementos multiplicativos), variando el número de pasos de la caminata en cuatro niveles (Nivel 1= 20 pasos, Nivel 2= 200 pasos, Nivel 3= 2000 pasos, Nivel 4= 20000 pasos), con 12 repeticiones del experimento para cada combinación. Los resultados se observan en una sola gráfica con diagramas de caja-bigote.

2. Desarrollo

Se realizaron diversos intentos para generar el código objetivo de la práctica, dichos intentos se encuentran en [mi repositorio](#) en GitHub. Inicie tomando como base el [código de Movimiento Browniano](#) revisado en clase [8], a este código le quité el paso de evaluar la distancia euclidiana, y para variar la caminata mi primera opción fue agregar cuatro 'for', para así ejecutar los cuatro niveles de caminata. Así mismo los resultados de cada experimento se almacenaban en un 'data.frame' (Ver Figura 1 1) vacío que se colocó al inicio del programa. Con este [nuevo código](#) al final se generó una gráfica con la información del 'data.frame' (Ver Figura 2 2).



	r1	r2	r3	r4	r5	r6	r7	r8	r9	r10	r11	r12
1	6	8	6	2	8	4	2	10	4	2	4	6
2	10	4	6	10	6	4	8	4	12	10	6	8
3	10	10	8	12	4	8	12	10	8	10	6	6
4	8	12	16	16	12	12	16	14	8	10	10	10
5	16	10	16	16	20	12	14	16	16	18	20	14
6	18	18	20	20	18	18	16	18	20	18	18	16
7	20	20	18	20	20	20	20	20	20	20	14	20
8	14	16	12	28	20	8	10	26	14	4	6	32
9	18	18	28	34	30	30	20	20	24	38	22	30
10	38	28	16	34	28	28	22	30	18	36	38	30
11	56	46	48	46	38	56	48	44	38	70	40	60
12	66	56	68	70	50	64	74	72	56	70	66	60
13	74	102	100	82	78	88	88	90	100	82	78	100
14	112	110	118	120	118	116	112	124	108	118	110	122
15	52	54	58	10	44	32	8	58	48	46	44	44
16	100	70	70	80	68	62	52	58	38	54	74	60
17	114	70	132	106	54	118	74	142	88	100	92	88
18	110	184	148	162	110	114	108	150	126	170	120	164
19	194	206	190	194	212	254	194	214	186	226	190	164
20	270	306	256	290	302	274	276	304	298	282	316	296
21	426	404	386	432	444	390	380	388	408	402	364	402
22	118	130	278	92	206	190	450	120	196	198	28	36
23	150	322	180	208	274	80	256	182	266	308	130	226
24	362	342	244	310	360	196	234	362	244	274	328	234
25	526	352	380	372	438	520	600	486	358	592	486	460
26	692	634	680	694	750	692	666	524	596	604	900	552
27	1016	928	692	892	820	788	1080	910	914	912	832	910
28	1210	1258	1236	1380	1344	1242	1350	1198	1330	1196	1136	1292

Figura 1: 'Data.frame' que se obtiene al ejecutar el programa.

En la Figura 1 1 podemos ver que las primeras 7 filas corresponden a las distancias de la caminata Nivel 1, de la fila 8 a la 14 corresponden a la caminata Nivel 2, de la 15 a la 21 corresponden a la caminata Nivel 3 y de la 22 a la 28 al Nivel 4. En la Figura 2 2 podemos ver que los 4 niveles de caminata se ordenan de manera consecutiva en el 'eje x' debido al ordenamiento de los datos en el 'data.frame'. Ya que la gráfica generada no permite visualizar los datos correctamente, se utilizó la librería 'ggplot2', la cual permite generar diagramas de caja-bigote agrupados.

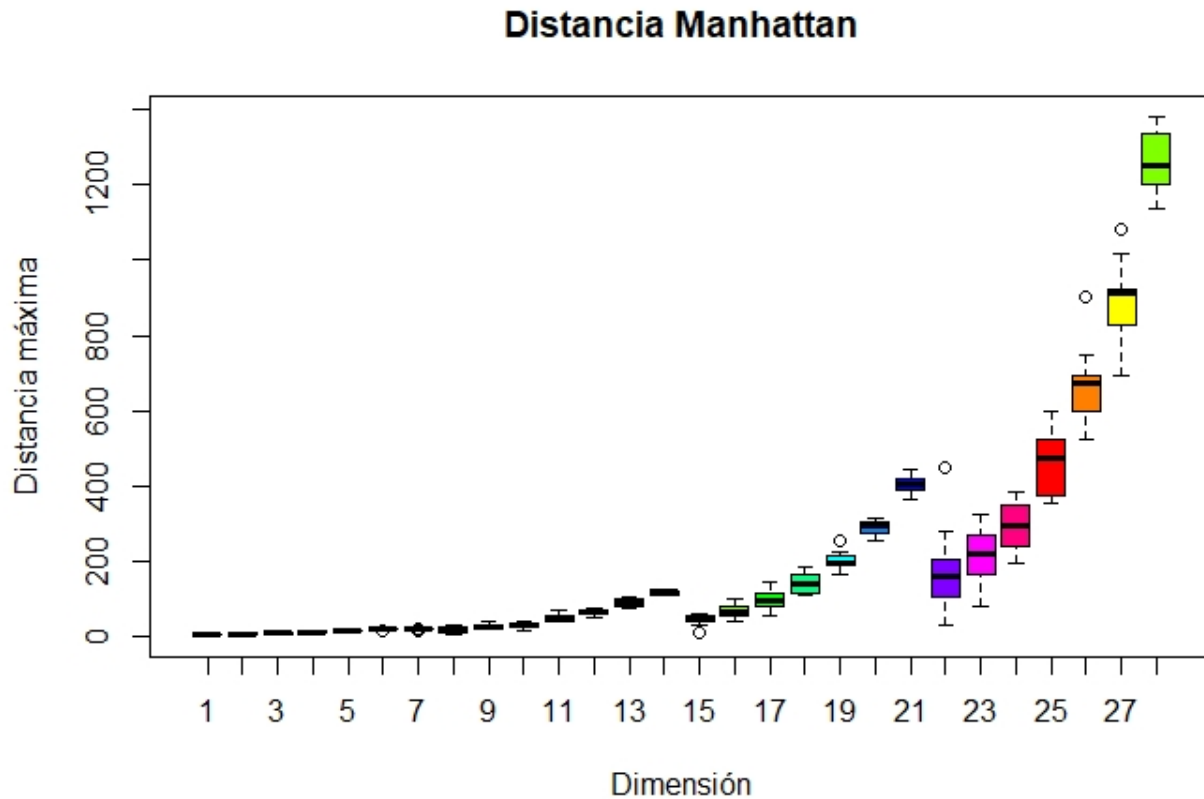


Figura 2: Primer intento de diagrama caja-bigote para los 4 niveles de caminata.

Se revisaron diferentes páginas en internet sobre documentación [12], [5] y video tutoriales para aprender a realizar diagramas caja-bigote o boxplot con la librería 'ggplot2'[6], [3], [4], sin embargo debido al ordenamiento que tenía mi 'data.frame' se me dificultó acomodar las variables para generar mi boxplot, ya que en éste se deben indicar los datos para el 'eje x' y los datos para el 'eje y'. Se revisó documentación para tratar ordenar adecuadamente los datos en el 'data.frame' [1], [2], [7], [11].

El siguiente paso que realicé fue preguntar mis dudas a mis compañeros, y mi compañera de clase Claudia Hernández me explicó una metodología para realizar una gráfica en 'ggplot2', la cual consistía en importar un archivo de Excel con los datos debidamente ordenados y a partir de éste generar el diagrama caja-bigote. De esta forma con mi código obtuve las distancias para los 4 niveles de caminata y los acomode en un archivo de Excel, el cual posteriormente importé y después generé otro código para realizar el diagrama caja-bigote (Ver Figura 3 3). En esta grafica podemos observar que las 4 caminatas se encuentran debidamente ordenadas.

Debido a que aún presentaba dudas en la manera de realizar mi código, presenté mis dudas en el canal de Discord de la clase de Simulación, en donde la Dra. Elisa me brindó retroalimentación, sugirió documentación para revisar [10] y compartió códigos ejemplo [9].

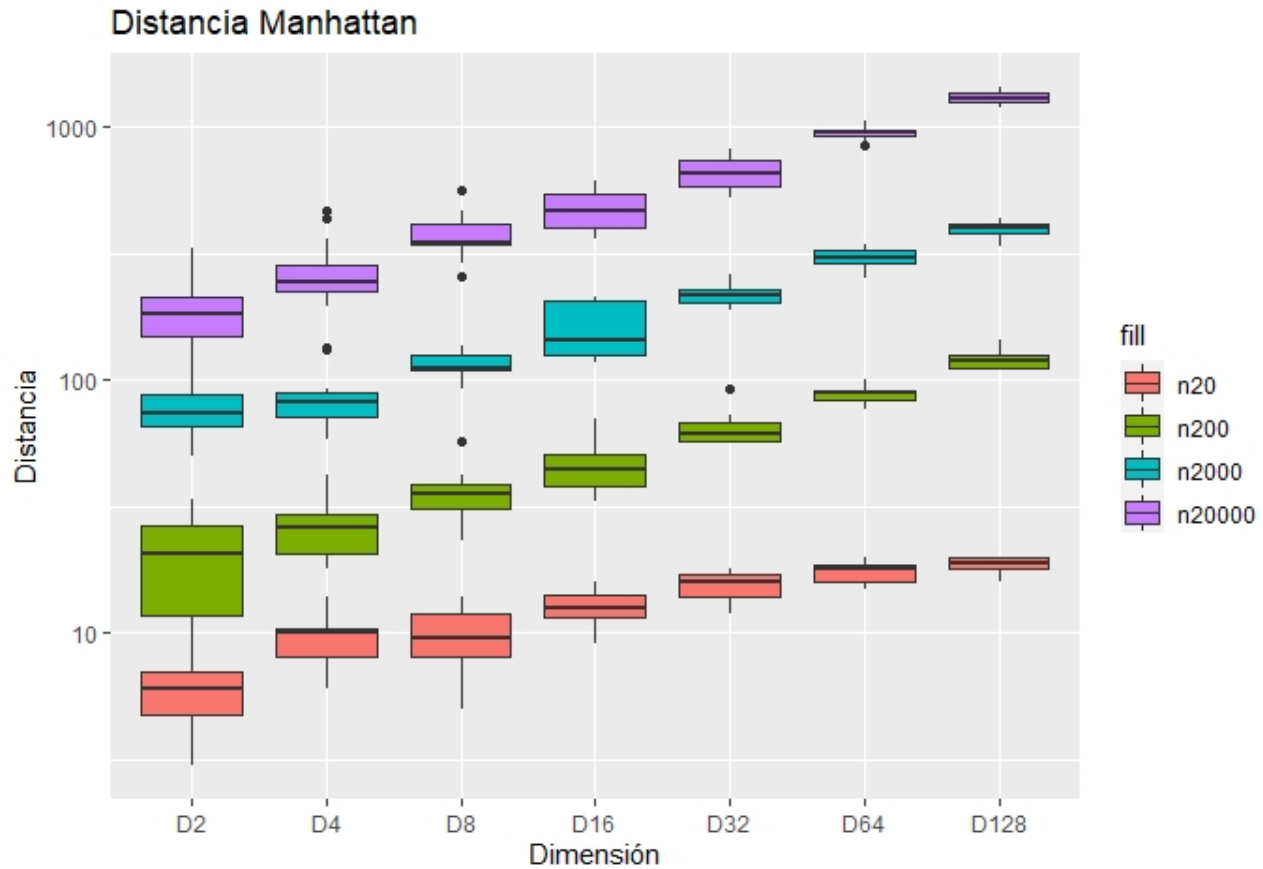


Figura 3: Segundo intento de diagrama caja-bigote importando datos desde un archivo de Excel.

Finalmente me base en el código ejemplo de la Dra. Elisa para realizar un data.frame debidamente ordenado, además entendí como realizar la secuencia de las 4 caminatas sin tener que poner 4 'for' como lo estaba haciendo. El código final para obtener el Diagrama caja-bigote de acuerdo al objetivo de la práctica se muestra a continuación:

```
library(dplyr)
library(ggplot2) #libreria para graficar boxplot agrupado
library(scales) #libreria para hacer modificaciones a la grafica
datos = data.frame() #para almacenar los datos obtenidos
niveles = c(20, 200, 2000, 20000) #caminatas: Nivel 1, Nivel 2, Nivel 3 y Nivel 4

for (dimension in c(2**1, 2**2, 2**3, 2**4, 2**5, 2**6, 2**7)) { #seleccionar dimension
  for (duracion in niveles) { #seleccionar la caminata
    for (replica in 1:12) { #repetir el experimento
      pos = rep(0, dimension)
      mayor = 0
      for (t in 1:duracion) {
        cambiar = sample(1:dimension, 1)
        cambio = 1
        if (runif(1) < 0.5) {
          cambio = -1
        }
        pos[cambiar] = pos[cambiar] + cambio
        d <- sum(abs(pos))
      }
    }
  }
}
```

```

    if (d > mayor) {
      mayor = d
    }
  }
  resultado <- c(dimension, duracion, replica, mayor) #vector para agrupar resultados
  datos <- rbind(datos, resultado) #para llenar en el dataframe vacio
}
}
names(datos) <- c("dim", "Caminata", "rep", "dist") #nombrar columnas del dataframe

datos$dim = as.factor(datos$dim) #crear vector a partir del dataframe
datos$Caminata = as.factor(datos$Caminata) #crear vector a partir del dataframe

ggplot(datos, aes(x= dim, y= dist, fill= Caminata)) +
  geom_boxplot(width=1)+
  labs(x = "Dimension", y = "Distancia", title = 'Distancia Manhattan')+ #nombres
  scale_y_log10() #cambiar la escala del eje "y" a logaritmo

```

El Diagrama caja-bigote que se obtiene del código anterior para 4 caminatas diferentes (Nivel 1= 20, Nivel 2= 200, Nivel 3 = 2000, Nivel 4 = 20000, con 2, 4, 8, 16, 32, 64 y 128 dimensiones y repitiendo el experimento 12 veces, se muestra en la Figura 4.

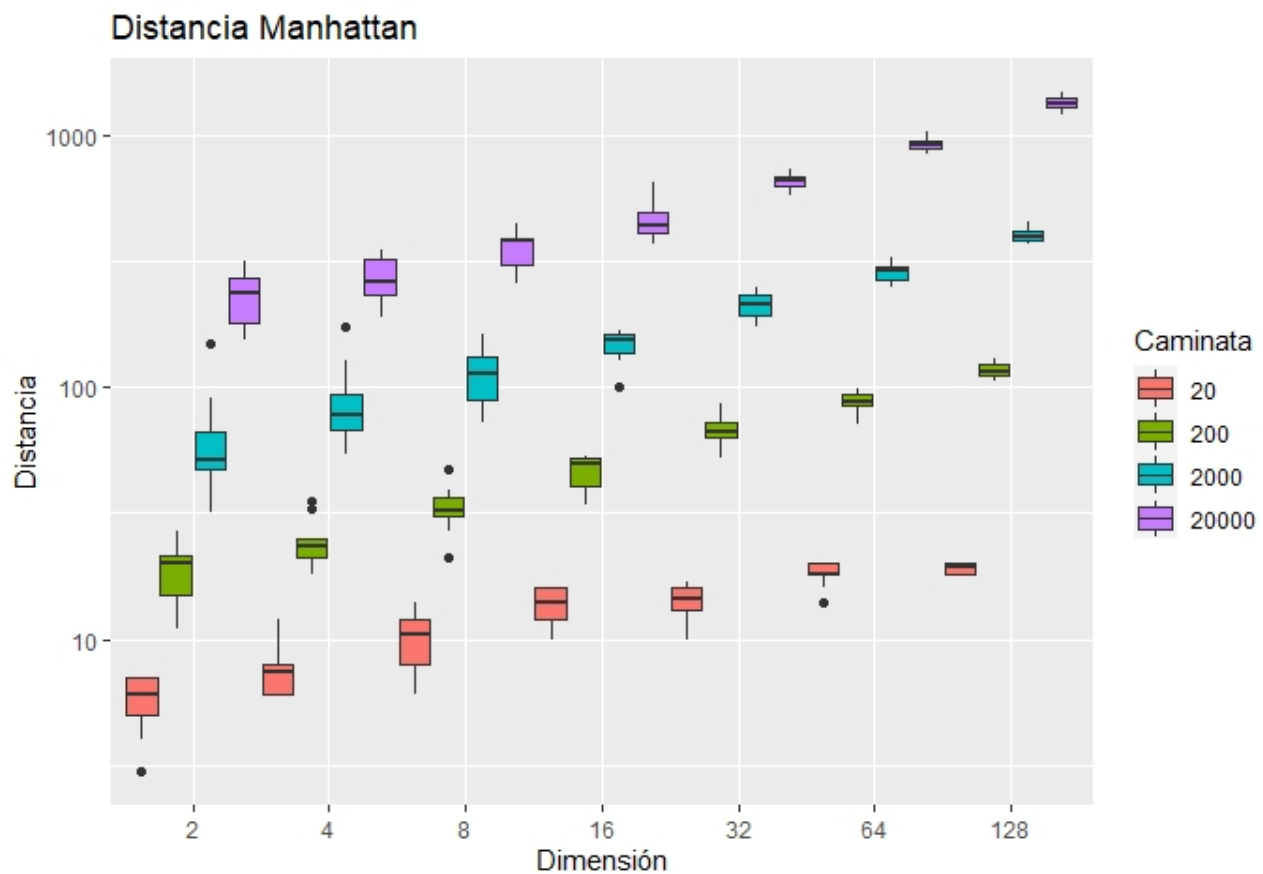


Figura 4: Diagrama caja-bigote con el eje y 'Distancia' en escala log10.

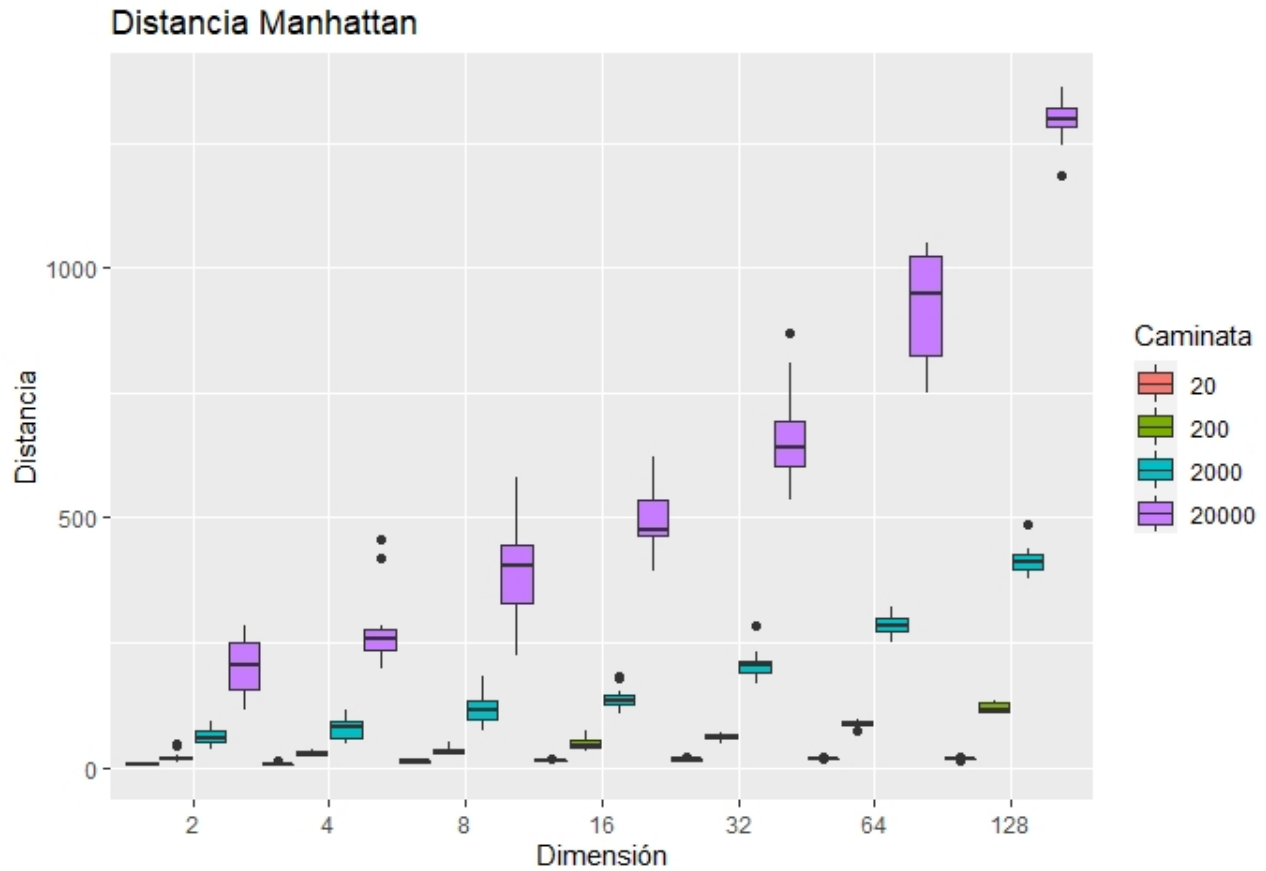


Figura 5: Diagrama caja-bigote sin escala en el eje y.

3. Conclusión

Con base en los resultados que se muestran en los Diagramas Caja-bigote de la Figura 4 y Figura 5 se puede concluir que al aumentar la cantidad de dimensiones en que se puede mover la partícula la distancia que recorre es mayor, considero que esto se debe a que tiene un mayor espacio para moverse y existe menos probabilidad de que regrese al origen. También se puede notar que cuando se aumenta el número de pasos de la caminata de la partícula se tiene un rango de distancias mayor.

En general el desarrollo de la práctica me permitió explorar más a fondo las funciones que tiene el programa RStudio, además de familiarizarme con su interfaz y también obtuve más práctica con el programa Overleaf. Así mismo considero que aún tengo mucho que aprender en cuanto a programación de código, dado que tuve muchas dificultades en esa parte.

Referencias

- [1] Carlos J. Gil Bellosta. Selección de filas y columnas. *R para profesionales de los datos: una introducción*, 2018. URL https://www.datanalytics.com/libro_r/seleccion-de-filas-y-columnas.html.
- [2] R CODER. Función merge en r. 2020. URL <https://r-coder.com/funcion-merge-r/>.
- [3] Codynamico. Crear y manipular dataframes en r-rstudio - using the data frame in r. 2018. URL https://www.youtube.com/watch?v=8G1G_boTT8k&ab_channel=Codynamico.
- [4] Rafa Gonzalez Gouveia. Cómo superponer varias gráficas [ggplot + r y rstudio]. 2019. URL https://www.youtube.com/watch?v=259PhB2pnzo&ab_channel=RafaGonzalezGouveia.
- [5] Yan Holtz. Grouped boxplot with ggplot2. *The R Graph Gallery*, 2018. URL <https://www.r-graph-gallery.com/265-grouped-boxplot-with-ggplot2.html>.
- [6] David Nelson. Rstudio scripts for side by side boxplots and clustered bar charts. 2020. URL https://www.youtube.com/watch?v=NmbK4lZVGqw&ab_channel=DavidNelson.
- [7] Raúl Ortiz. Unir dataframes. rbind() y cbind(). *RPubs by RStudio*, 2015. URL <https://rpubs.com/Rortizdu/140154>.
- [8] Dra. Elisa Schaeffer. Brownianmotion. *Repositorio, GitHub*, 2019. URL <https://github.com/satuelisa/Simulation/tree/master/BrownianMotion>.
- [9] Dra. Elisa Schaeffer. dataframe.r. 2021. URL <https://github.com/satuelisa/Simulation/blob/master/BrownianMotion/dataframe.R>.
- [10] Statistical tools for high-throughput data analysis. Ggplot2 box plot quick start guide r software and data visualization. URL <http://www.sthda.com/english/wiki/ggplot2-box-plot-quick-start-guide-r-software-and-data-visualization>.
- [11] Juan Bosco Mendoza Vega. Data frames. *R para principiantes*. URL <https://bookdown.org/jboscomendoza/r-principiantes4/>.
- [12] Hadley Wickham. ggplot2: Elegant graphics for data analysis. 2016. URL <https://ggplot2.tidyverse.org>.