Práctica Nº 5
Método Monte-Carlo

Nombre: José Adrián García Fuentes **Profesor:** Satu Elisa Schaeffer
Fecha: 16/Marzo/2021

---

1. Introducción

El método Monte-Carlo es idóneo para situaciones en las cuales algún valor o alguna distribución no se conoce y resulta complicado de determinar de manera analítica [1], es un método utilizado para conseguir aproximaciones de expresiones matemáticas y de alto grado para ser evaluadas con exactitud. Consiste en usar un espacio delimitado donde se encuentra la función a evaluar, la idea principal del método es generar números aleatorios dentro de ese espacio y calcular la proporción de puntos que están dentro o fuera, por arriba o por abajo de la función y obtener así una aproximación del área cubierta [2].

2. Objetivo

- Determinar el tamaño de muestra requerido por lugar decimal de precisión para el integral, comparando con Wolfram Alpha para por lo menos desde dos hasta cinco decimales [1].
- Representar el resultado como una sola gráfica con el número de decimales correctos contra el tamaño de muestra [1].

3. Metodología

La metodología empleada se realizó a través de RStudio [3] llevando a cabo los pasos señalados en la *Práctica 5: Método Monte-Carlo* [1], se implementa el método Monte-Carlo para aproximar el valor de la integral (1)

$$\int_a^b f(x)dx \tag{1}$$

para la función (2)

$$f(x) = \frac{1}{exp(x) + exp(-x)} \tag{2}$$

esta aproximación es posible porque la función (3)

$$2f(x)/\pi \tag{3}$$

es una distribución de probabilidad válida igual a 1, observe la ecuación (4).

$$\int_{-\infty}^{\infty} \frac{2}{\pi} f(x)dx = 1 \tag{4}$$

Este hecho permite generar números pseudoaleatorios con la distribución $g(x) = \frac{2f(x)}{\pi}$, así estimar $\int_a^b g(x)dx$, y de ahí normalizar el estimado para que sea $\int_a^b f(x)dx$.

A partir del código en el repositorio de Schaeffer [4] se puede realizar el gráfico de distribución de probabilidad como el de la figura 1, el código completo de la metodología empleada se encuentra en el repositorio de GitHub [5].

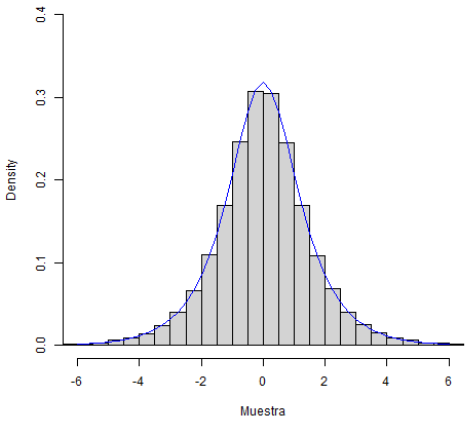

Figura 1:
Histograma de g(x) comparado con g(x)

4. Resultados

Se obtuvo el código secuencia de GitHub de Schaeffer [4] se adecuaron modificaciones con el fin de determinar el tamaño de muestra requerido por lugar decimal de precisión para el integral, se representaron los resultados en una sola gráfica con el número de decimales correctos contra el tamaño de la muestra (ver figura 2).

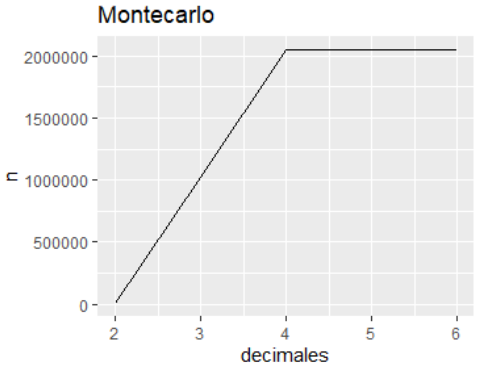

Figura 2:
Gráfico precisión de decimales variando tamaño de muestra [6].

A continuación se muestra parte del código del experimento [5].

```
library(distr)
library(ggplot2)
wolf=0.048834
rate=0.80
exitos=as.data.frame(matrix(nrow = 5, ncol = 1))
wolfram=cbind(round(wolf,digits=2),round(wolf,digits=3),round(wolf,
    digits=4),round(wolf,digits=5),wolf)
montecarlosf<-function(n) {
  desde <- 3
  hasta <- 7
  f <- function(x) { return(1 / (exp(x) + exp(-x))) }
  g <- function(x) { return((2 / pi) * f(x)) }
  suppressMessages(library(distr))
  generador  <- r(AbscontDistribution(d = g))
  valores <- generador(n)
  mc=sum(valores >= desde & valores <= hasta)
  integral <- sum(mc) / n
  resultado=(pi / 2) * integral
  return(resultado)
}
replica <- function() { replicate(reps, montecarlosf(nsample))  }
for (c in 2:6) {
  reps=20
  nsample=5000
  paso=1000
for (i in 1:10) {
resultado<-round(replica(),digits=c)
if (as.numeric(length(resultado[resultado == wolfram[1,c-1] ])/reps
    ) <= rate){
  paso=paso+paso
  nsample=nsample+paso
} else {
  break
}
}
  exitos[c-1,1]=nsample
}
ggplot()+geom_line(data=exitos,aes(x=c(2:6),y=exitos$V1))+labs(
    title="Montecarlo",y="n",x="decimales")
```

A medida que aumenta la cantidad de la muestra, el resultado se aproxima mayormente al valor obtenido en Wolfram Alpha (0,048834). Por tanto la precisión de los decimales se ve afectada por el tamaño de la muestra, observe como en la figura 2 se logra una mayor presición en el número de decimales mientras más grande es la muestra.

5. Conclusión

Si se aumenta el tamaño de n para la estimación de la integral, se obtendrá una mayor precisión con respecto al valor de Wolfram Alpha, este método es una opción viable para replicar el cálculo una cantidad de veces, ya que se trabaja con valores aleatorios, para el caso de muestras grandes se pueden tener aproximaciones en períodos cortos de tiempo con una mayor exactitud.

6. Reto 1

Implementar la estimación del valor de π de Kurt y determinar la relación matemática entre el número de muestras obtenidas y la precisión obtenida en términos del error absoluto [1].

Para la comprensión es necesario saber como determinar el área de un círculo (5)

$$A = \pi r^2 \tag{5}$$

y el área de un cuadrado que contiene ese círculo (6)

$$A = 4r^2 \tag{6}$$

La relación del área del círculo con el área de la plaza es (7)

$$\frac{\pi r^2}{4r^2} \tag{7}$$

que se puede reducir a (8).

$$\frac{\pi}{4} \tag{8}$$

Dado este hecho, si se puede determinar la relación del area del círculo con el área de la plaza [6].

```
runs <- 100000
xs <- runif(runs,min=-0.5,max=0.5)
ys <- runif(runs,min=-0.5,max=0.5)
in.circle <- xs^2 + ys^2 <= 0.5^2
mc.pi <- (sum(in.circle)/runs)*4
plot(xs,ys,pch='.',col=ifelse(in.circle,"blue","grey")
     ,xlab='',ylab='',asp=1,
     main=paste("MC Aproximacion de Pi =",mc.pi))
```

En la figura 3 se muestra la estimación del valor de π de kurt

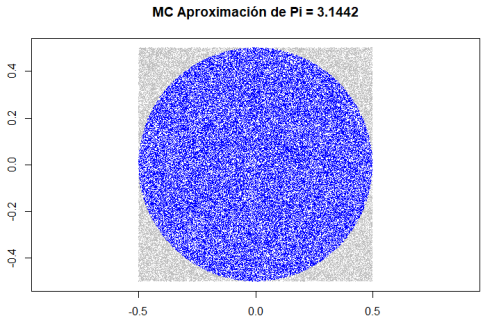

Figura 3:
Estimación del valor de π de Kurt [7].

De igual manera que la primera parte de esta práctica, se logra visualizar el mismo patrón a medida que n cambia [8]

7. Reto 2

Aplicar un método Monte Carlo para estimar la cantidad de pintura necesaria en un mural, comparando conteos exactos de píxeles de distintos colores con conteos estimados con muestreo aleatorio [1].

```
install.packages("jpeg")
library(jpeg)
img <- readJPEG("C:/Users/ADRIAN GARCIA/Desktop/img.jpg")
img <- as.raster(img)
tab <- table(img)
tab <- data.frame(Color = names(tab), Count = as.integer(tab))
RGB <- t(col2rgb(tab$Color))
tab <- cbind(tab, RGB)
tot=sum(tab$Count)
ratios=(tab$Count/tot)*100
tab$'cantidadpintura%'<-round(ratios,2)
```

En la figura 4 se muestra una pintura de mural en la que se comparo un total de 288000 pixeles de los cuales se obtuvo un total de variables de 6 colores distintos sin embargo se determino la cantidad de pintura en porcentaje para utilizar solo entre dos tipos de colores los datos obtenidos se encuentran dentro del repositorio [5]

Figura 4:
Pintura Mural [9]

Referencias

[1] E. Schaeffer, "Práctica 5: Método monte-carlo," Marzo 2021. https://elisa.dyndns-web.com/teaching/comp/par/p5.html.

[2] D. Peña Sanchez, "Deducción de distribuciones: el método de monte carlo," Fundamentos de estadistica Madrid. Recuperado el 15 de marzo del 2021.

[3] J. J. Allaire, "Rstudio," Marzo 2021. https://rstudio.com.

[4] E. Schaeffer, "Práctica 5: Método monte-carlo," Marzo 2021. https://github.com/fuentesadrian/Simulation/tree/master/MonteCarlo.

[5] J. A. Garcia Marzo 2021. https://github.com/fuentesadrian/SIMULACION-DE-NANOMATERIALES/tree/main/Tarea%205.

[6] O. Cheong, "Computational geometry: Algorithms and applications," Marzo 2008. Recuperado el 05 de marzo del 2021.

[7] kurt Will, "Monte carlo simulations in r," 2015. https://www.countbayesie.com.

[8] P. Pérez, "Aplicación de la técnica de simulación monte carlo," 2018. https://www.famaf.unc.edu.ar/~pperez1/manuales/cim/cap6.html.

[9] H. Weller, "Introduction to countcolors package," 2019. https://cran.r-project.org/web/packages/countcolors/vignettes/Introduction.html.