Simulación **Tarea 5: Método Monte-Carlo** 15/Mar/2021

C. María Montemayor Palos

---

1. Objetivo

Examinar y analizar los resultados aplicando el método Monte-Carlo para así estimar el resultado de una integral definida determinando el tamaño de la muestra para la precisión por lugar decimal, comparando el resultado con Wolfram Alpha [1] de dos a cinco decimales. Finalmente representando los resultados en un diagrama de violín [3].

2. Metodología

Para efectos de la tarea se utiliza el programa R versión 4.0.4 [5] para Windows. Se pretende calcular el valor de la integral (1) para la función (2) empleando el método Monte-Carlo y comparar el valor obtenido por Wolfram Alpha de 0.048834.

$$\int_3^7 f(x)dx \tag{1}$$

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

La función es posible ya que la función $\int_{-\infty}^{\infty} \frac{2}{\pi} f(x)\mathrm{d}x$ es una distribución de probabilidad válida, es decir, la integral de esa función es igual a uno; por esto se pueden crear valores aleatorios basados en esta. Se utilizan estos valores para calcular el área entre los límites de la integral, usando la suma de los valores entre los límites derecho e izquierdo se tendrá el valor aproximado del área, como se muestra en la figura 1.

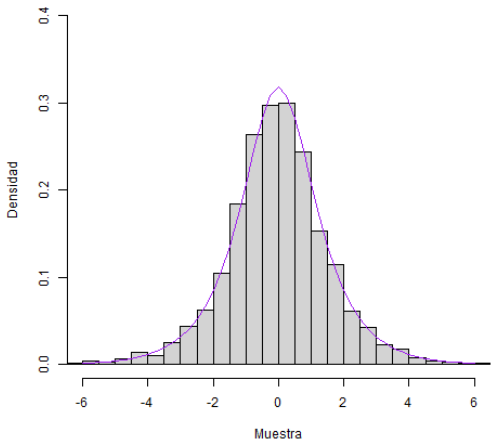


Figura 1. Histograma de g(x) comparado con g(x).

Se modifica el código de Schaeffer [4] para calcular el valor de la integral empleando como números de muestra 50000, 100000, 500000 y 1000000, estableciendo diez réplicas del experimento, posteriormente se comparan los resultados con el valor esperado para comprobar la precisión de los decimales.

3. Código

Se establece el número de réplicas del experimento y la variación en el tamaño de las muestras para cada réplica. El código completo se encuentra en el repositorio de Montemayor [6].

```
    n=c(50000,100000,500000,1000000)
    replica=10
    inicio= -6
    final= -inicio
    aum=0.25
    x=seq(inicio,final,aum)
```

Se ejecuta la función que calcula la integral guardando así los resultados en el vector `datos`.

```
    for(muestra in n){
        for(r in 1:replica){
         montecarlo = foreach(i = 1:cuantos, .combine=c) %dopar% parte()
        integral = sum(montecarlo) / (cuantos * muestra)
        resultado=((pi / 2) * integral)
        datos <- c(datos,resultado)
        print(resultado)
  }
}
```

Se usa la comparación de cadenas de caracteres para comprobar la precisión de los decimales. Posteriormente se convierten los resultados a cadenas de caracteres para comparar y se guarda en el vector `resultados_str`.

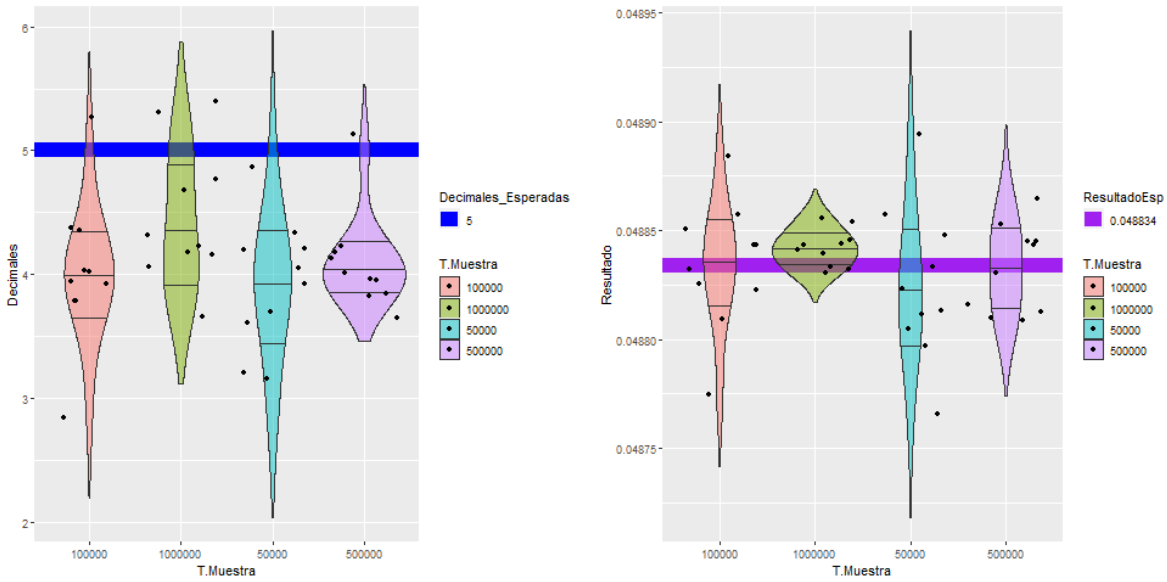
```
    resultados <- c()
    resultados_str <- c()
    res_contadores <- c()

for(i in 1:length(data.matrix(datos))) {
  resultados <- c(resultados,data.matrix(datos)[i])
  resultados_str <- c(resultados_str, strsplit(paste(resultados[i]), split=""))
}
```

Finalmente se calcula la precisión de los decimales, se comparan y se incrementa la variable `contador` en uno cada vez que los resultados aciertan en una decimal. El ciclo se detiene cuando no acierta, pues deja de ser preciso en las siguientes decimales.

```
for(i in 1:length(resultados)) {
    contador <- 0
    for(muestra in 1:7) {
        b <- (unlist(resultados_str[i])[muestra])
        == (unlist(resultadoEsp_str)[muestra])
        if(b) {
            contador <- contador + 1
        } else {
        break;
        }
    }
    res_contadores <- c(res_contadores,contador-2)
}
```

4. Resultados y discusión

En la figura 2 se muestran los resultados obtenidos a partir del método Monte-Carlo para la estimación de la integral. Se observa que a medida en que aumenta la cantidad de la muestra, el resultado se aproxima mayormente al valor obtenido en Wolfram Alpha. La precisión de los decimales se ve directamente afectada por el tamaño de la muestra, observe cómo en la figura 2a se logra alcanzar los cinco decimales de precisión en los cuatro tamaños de muestras, con una precisión promedio de cuatro decimales de exactitud como se muestra en la figura 2b.



(A) (B)

Figura 2. Comparación de los tamaños de muestra de acuerdo a la precisión de decimales y del valor esperado.

5. Conclusión

Si se aumenta el tamaño de las muestras para la estimación de la integral, generará resultados más cercanos al valor real, por lo tanto se recomienda establecer tamaños de muestras más óptimos para tener una buena precisión en los cálculos realizados por el método Monte-Carlo. Así mismo, también es una opción viable replicar el cálculo una cantidad de veces considerable ya que se trabaja con valores aleatorios.

6. Reto 1

El objetivo del primer reto consiste en implementar la estimación del valor de π de Kurt [7] a partir del método Monte-Carlo y determinar la relación matemática entre el número de muestras obtenidas y la precisión obtenida en términos del error absoluto. Se determinan los tamaños de muestras 100, 1000, 10000, 1000000 y 5000000 considerando como valor π= 3.141593 con 20 réplicas del experimento. Se supone un cuadrado el cual contiene un círculo de radio r en su

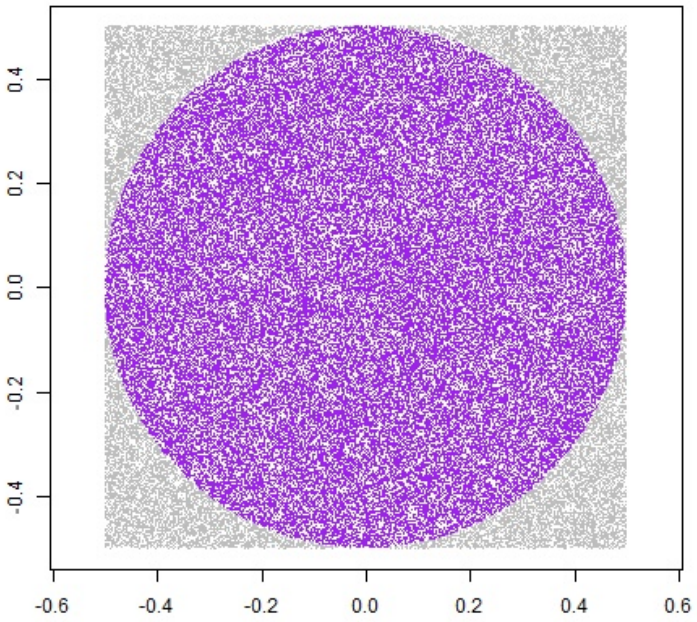


Figura 3. Ilustración del método Monte-Carlo para la aproximación de π.

interior. De esta manera se puede afirmar que el radio del círculo (r) será la mitad de los lados del cuadrado, por lo cual los lados del cuadrado será dos veces el radio del círculo ($2r$). Para determinar qué área del cuadrado pertenece al círculo se establece la relación en la ecuación (3).

$$R = \frac{\pi r^2}{4r^2} \tag{3}$$

En donde R es la relación del área del cuadrado y el área del círculo, se puede cancelar y despejar π para así de esta manera obtener la ecuación (4) que estaremos utilizando para estimar el valor de π.

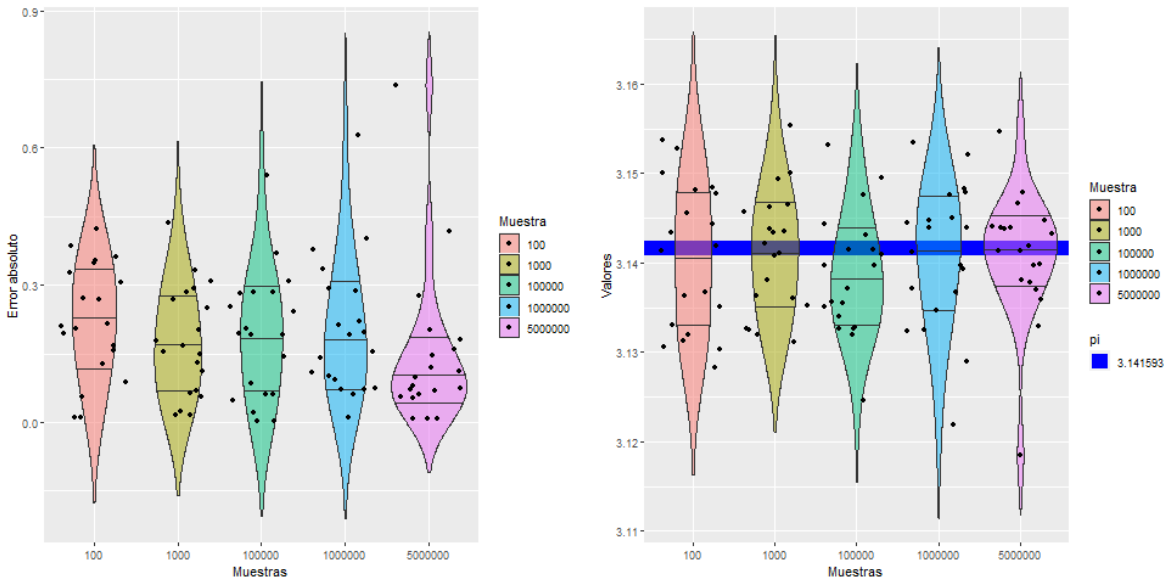
$$\pi = 4 * R \tag{4}$$

Se estima el valor de π.

```
  calculando <- function() {
  xs <- runif(i,min=-0.5,max=0.5)
  ys <- runif(i,min=-0.5,max=0.5)
  in.circle <- xs^2 + ys^2 <= 0.5^2
  numero <- (sum(in.circle)/i)*4
  return(numero)
}
```

Se utiliza el método Monte-Carlo y se obtiene la precisión en términos del error absoluto.

```
 for(i in n){
  for(r in 1:replicas){
    montecarlo<-foreach(i = 1:cuantos, .combine=c) %dopar% calculando()
    real <- sum(montecarlo) / cuantos
    Números <- c(Números, real)
    error<-((abs(pi-real))/(pi))*100
    errores<-c(errores,error)
  }
}
```



(A) (B)

FIGURA 4. Comparación de los tamaños de muestra de acuerdo a la precisión obtenida en términos del error absoluto y del valor esperado.

Se observa en la figura 4a que a medida en que se incrementa los valores de la muestra, menor es el porcentaje de error para la aproximación del valor de π, entre mayor sea el tamaño de muestra será más acertado el valor para estimar π. Esto se comprueba viendo que los valores promedio de todas las replicas y el tamaño de muestra, es decir, los valores generados son muy cercanos al valor esperado de π el cual está representado con una línea color azul en la figura 4b.

7. Reto 2

El objetivo del segundo reto es estimar la cantidad necesaria de pintura a utilizar para pintar un mural, aplicando el método Monte-Carlo. Se coomparan los conteos exactos de pixeles con conteos estimados con el muestreo aleatorio. Se escoge una imagen para el mural, en este caso el logo de Starbucks [2], en el cual se toma en cuenta que para cada pixel le corresponde $10cm^2$ del mural, y que cada litro de pintura rinde $10m^2$. Se emplea la librería "countcolors" en R [5], la cual toma en cuenta rangos definidos de colores de pixeles, además genera una imagen en donde sustituye los pixeles a contar por algún otro color de elección para confirmar que corresponda a la región de interés. Se definen los rangos de color y se cuentan exactamente los pixeles por cada rango de color definido.

```
white.center = c(1, 1, 1)
green.center=c(0, 0, 0)
png("star-blanco.png")
star.white=countcolors::sphericalRange(star, center = white.center, radius = 0.5,
    color.pixels = FALSE, plotting = TRUE, target.color="cyan")
blanco=star.white$pixel.count
graphics.off()
png("star-verde.png")
star.green=countcolors::sphericalRange(star, center = green.center, radius = 0.3,
    color.pixels = FALSE, plotting = TRUE, target.color="blue")
verde=star.green$pixel.count
graphics.off()
```

(A) (B) (C)

Figura 5. Imagen original del logo de Starbucks y las sustituciones de color de pixeles generadas con countcolors en R.

Para la estimación por el método Monte-Carlo se realiza un muestreo al azar de 10,000 datos y 1,000 repeticiones del experimento, para obtener la media de pixeles correspondientes al color verde, y con ello la estimación de Monte-Carlo. La diferencia del resultado con el total de pixeles de la imagen (307,343) corresponde a la estimación de pixeles de color blanco.

```
runs=1000
datos=data.frame()
for(r in 1:length(runs)){
  for(s in 1:10000){
    blue = star[,,3]
    x =sample (blue, runs[r])
    y=sum(x < 0.5)
    print(y)
    datos=rbind(datos,c(s,runs[r],y))
  }
}
```

Finalmente se calcula la cantidad de pintura necesaria para un mural de $307m^2$ de la imagen.

```
pixelesmc=c(mcblanco,montecarloverde)
pixelesmc
pinturamc=pixelesmc*0.001
pinturamc
```

En el cuadro 1 se presenta el número de pixeles de color verde y blanco y su respectivo equivalente en litros de pintura para la estimación con números aleatorios y la cuenta exacta. En la figura 5 se muestra la figura original del logo de Starbucks, así como las figuras generadas con la librería countcolors, la cual sustituye los pixeles verdes por el color azul 5b, y los pixeles blancos por el color turquesa 5c, como confirmación de la cuenta exacta de los pixeles.

CUADRO 1. Estimación de pixeles y litros de pintura por cada color.

Color	**Monte Carlo**		**Cuenta exacta**	
	Pixeles	Pintura	Pixeles	Pintura
Verde	88,650	88.65 L	76,491	76.49 L
Blanco	218,692	218.69 L	230,852	230.85 L

En conclusión, se observa que en la aproximación realizada por el método Monte-Carlo se obtuvo una significativa diferencia respecto a la cuenta exacta de pixeles, pero para efectos prácticos si se puede recurrir a este método para darse una idea aproximada de los litros de pintura a usar si se consideran tamaños de murales más chicos.

Referencias

[1] Wolfram Alpha, 2021. URL https://www.wolframalpha.com/.

[2] Starbucks Coffee Logo. Starbucks Coffee Logo, 2010. URL https://i.etsystatic.com/11085793/r/il/fcb381/828448218/il_570xN.828448218_55fp.jpg.

[3] Schaeffer E. Práctica 5: Método Monte-Carlo, 2021. URL https://elisa.dyndns-web.com/teaching/comp/par/p5.html.

[4] Schaeffer E. Método Monte-Carlo., 2021. URL https://github.com/satuelisa/Simulation/blob/master/MonteCarlo/integral.R.

[5] The R Foundation. The R Project for Statistical Computing, 2021. URL https://www.r-project.org/.

[6] Montemayor M. Tarea 5: Método Monte-Carlo, 2021. URL https://github.com/MariaMontemayor/Simul/tree/main/tarea5.

[7] Kurt W. 6 Neat Tricks with Monte Carlo Simulations — Count Bayesie., 2021. URL https://www.countbayesie.com/blog/2015/3/3/6-amazing-trick-with-monte-carlo-simulations.