Práctica 5: método Monte-Carlo

Ricardo Rosas Macías

21 de mayo de 2019

1. Introducción

El método Monte Carlo es una herramienta matemática para calcular las probabilidades mediante una serie de números aleatorios, en virtud de ello nos permite identificar cuántos puntos se necesitan para estimar el valor de interés.

2. Objetivo

En términos generales, se realizó cambios en el código de modo que proporcionó el estimado en función de los decimales, para determinar el tamaño de muestra necesario para la aproximación del valor real.

2.1. Descripción

Lo que se debe hacer es [3]:

> "Determinar el tamaño de muestra requerido por cada lugar decimal de precisión del estimado obtenido para la integral, comparando con la calculadora virtual Wolfram Alpha [2] de uno hasta siete decimales, asimismo representar el resultado como una sola gráfica. De igual manera para el primer reto, implementar la estimación del valor de π de Kurt [1] con paralelismo, determinar la relación matemática entre el número de muestras obtenidas y la precisión obtenida en términos de la cantidad de lugares decimales correctos"

3. Resultados y conclusiones

Para obtener los resultados se tomaron en base a la integral $\int_3^7 f(x)dx$ respecto a la función: $f(x) = \frac{1}{\exp(x) + \exp(-x)}$. Asimismo con los datos se realizó una normalización ($g(x) = 2f(x)/\pi$) para obtener la distribución arbitraria, como se muestra en las siguientes líneas del código.

```
desde <- 3
hasta <- 7
```

```r
Pedazos <- 300
cuantos <- 100
muestra <- c(50, 100, 500, 1000, 5000, 10000)
wolfram<-0.048834 # Valor real
repeticiones <- 20
parte <- function() {
  val <- generador(pedazo)
  return(sum(val >= desde & val <= hasta))
}
Info <- data.frame()
for (pedazo in Pedazos) {
  for (cuantos in muestra){
    for (vez in 1:repeticiones){
      suppressMessages(library(doParallel))
      clust <- makeCluster(2)
      registerDoParallel(clust)
      montecarlo <- foreach(i = 1:cuantos, .combine=c) %dopar% parte()
      stopCluster(clust)
      integral <- sum(montecarlo) / (cuantos * pedazo)
      piInt <- (pi / 2) * integral
      pI <- c(piInt, cuantos, as.integer(pedazo))
      print(cuantos)
      Info <- rbind(Info, pI)
    }
  }
}
colnames(Info) <- c("val", "rep", "esp")
Info$esp <- as.integer(Info$esp)
Info$rep <- as.factor(Info$rep)
print(Info)

agregando <- function(Espacio){
  Espacio <- as.factor(Espacio)
  return(paste("n = ", Espacio, sep = ""))
}
valmin <- (wolfram-50)
```

El resultado final del código proporcionado muestra un valor aproximado que posteriormente se comparó con el valor real proporcionado de la calculadora Wolfram [2]. En la figura 1 se observa la aproximación al valor real, en donde la secuencia de la muestra del experimento fue 300 realizando cambios en su tamaño; 50, 100, 500, 1000, 5000 y 10000, con 10 repeticiones para cada una. Esto nos permito conocer la aproximación al valor real: 0.0488340. De manera que la gráfica nos muestra que al incrementar el tamaño de muestra la precisión incrementa en el resultado final, por lo tanto a 10000 la muestra tendrá mayor exactitud en los decimales del valor real, así como poca variación de los números aleatorios.

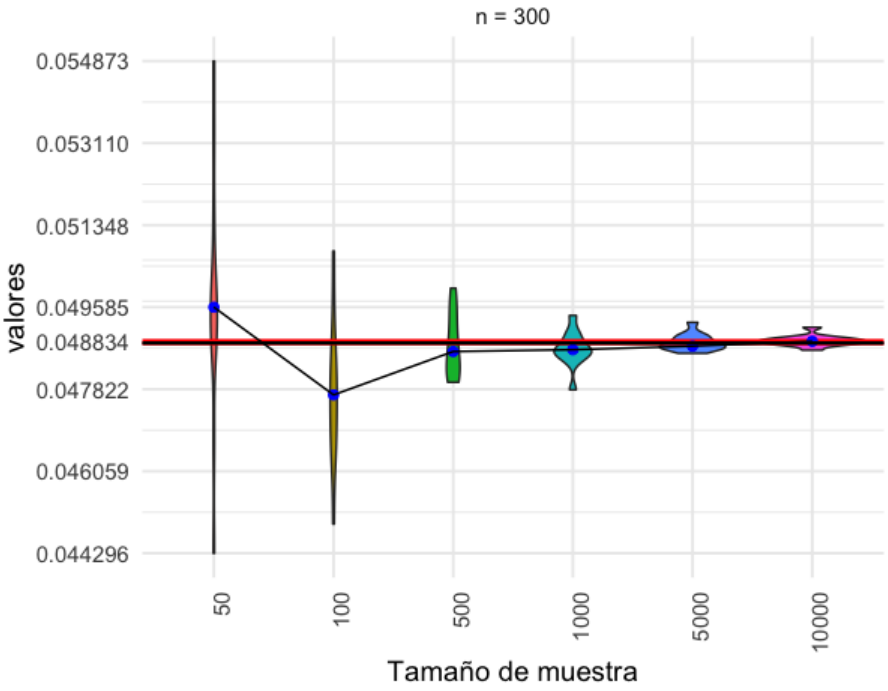


Figura 1: Aproximación al valor real

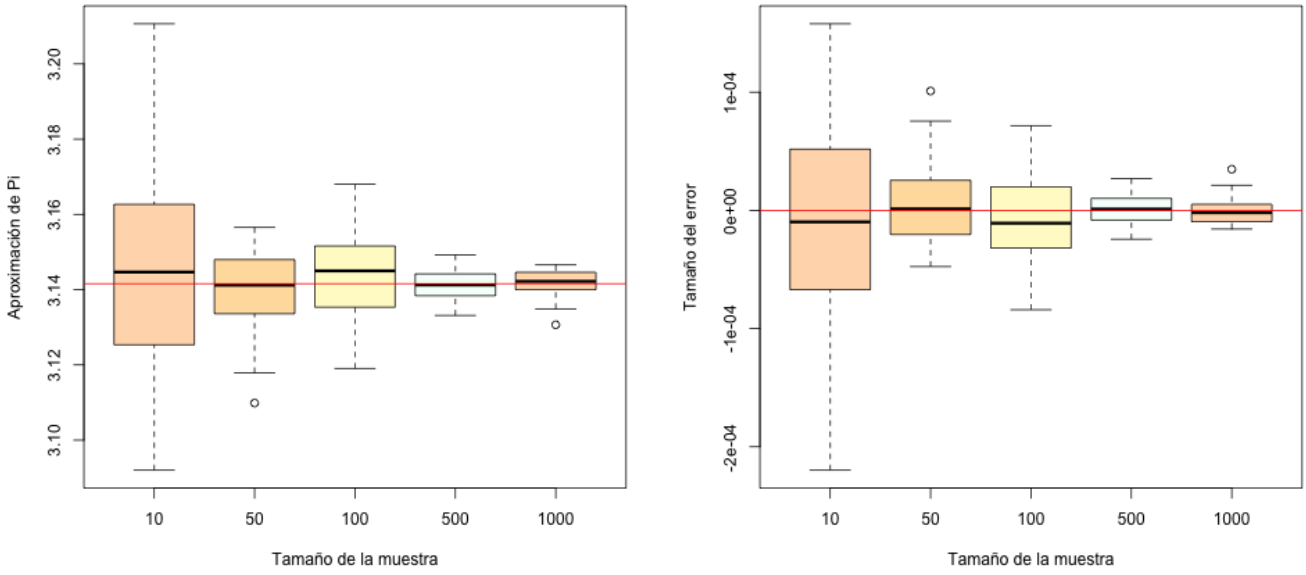
3.1. Reto 1

De acuerdo a la ponderación anterior, para el primer reto se realizó el experimento a 10 repeticiones y con una secuencia de 300; en el cual las condiciones permitieron observar la aproximación de π (3.14159265359), como se muestra en las siguientes líneas de código.

```
inicio <- -0.1
final <- -inicio
pi <- 3.14159265359
muestra <- c(10,50,100,500,1000)
repeticiones <- 30

calcpi=function() {
  xs <- runif(replicas, min= inicio, max= final)
  ys <- runif(replicas, min= inicio, max= final)
  in.circle <- xs^2 + ys^2 <= inicio^2
  mc.pi <- (sum(in.circle)/replicas)*4
  return(mc.pi)
}
suppressMessages(library(doParallel))
registerDoParallel(makeCluster(detectCores() - 1))

resultados=data.frame()

for(replicas in muestra) {
  for(i in 1:repeticiones) {
    montecarlo <- foreach(i = 1:300, .combine=c) %dopar% calcpi()
    mc.pi=sum(montecarlo)/300
```

```
      diferencia=(pi-mc.pi)/(pi*100)
      pii <- pi
      resultados=rbind(resultados,c(replicas,i,pii,mc.pi,diferencia))
    }
}
names(resultados)=c("muestra","replicas","Valor de Pi","aprox.pi","error")
```

De modo que en la gráfica a de la figura 2 podemos notar que en comparación a los resultados anteriores hay una diferencia al tener un tamaño de muestra menor, este presenta una mejor aproximación al valor, como es el caso de la muestra 500; la cual denota un ligero cambio porcentual obtenido, así como su tamaño de error es ligeramente desapercibido en contraposición de las demás muestras.



(a) Aproximación a π

(b) Tamaño de error

Figura 2: Estimación al valor de π

Referencias

[1] Will Kurt. 6 neat tricks with Monte Carlo simulations - count Bayesie; probably a probability blog, 2015. URL `https://www.countbayesie.com/blog/2015/3/3/6-amazing-trick-with-monte-carlo-simulations`.

[2] Wolfram Research. Mathematica, 2019. URL `https://www.wolframalpha.com/input/?i=integrate+(2%2Fpi)+*+(1%2F(exp(x)%2Bexp(-x)))+from+-infty+to+infty`.

[3] Elisa Schaeffer. Práctica 5: método Monte-Carlo, 2019. URL `https://elisa.dyndns-web.com/teaching/comp/par/p5.html`.