TAREA # 5
Método Monte-Carlo

Natalia Berenice Pérez López

29 de septiembre de 2021

1. Objetivo

El objetivo de esta práctica es estudiar estadísticamente la convergencia de la precisión del estimado de la integral con en método Monte-Carlo, comparado con el valor producido por `Wolfram Alpha`, en términos del número de decimales correctos, aumentando el tamaño de muestra.

2. Desarrollo

Para generar el código objetivo de esta práctica primero se realizó un código que solamente compara los decimales de dos números y muestra como resultado la cantidad de decimales que coinciden entre ellos, para esto se realizaron algunas ideas de código, las cuales se encuentran en mi repositorio en GitHub. A continuación se muestra el código generado:

```
#IDEA 4
bueno = 0.048834 #numero de Wolfram Alpha
prueba = 0.048214

n = seq(1, 6, 1)

for (i in n) {
  b = trunc(bueno*10^i)/10^i
  p = trunc(prueba*10^i)/10^i

  if (p == b) {
    deci = i
  } else {
    print(deci)
    break
  }
}
```

Listing 1: Código para obtener la cantidad de decimales que coinciden entre dos números.

Posteriormente, se utilizó como base el código revisado en clase [3] para calcular el valor de la integral definida (1) para la función (2) utilizando el método Monte-Carlo.

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

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

Este código permite generar números pseudoaleatorios con la distribución de la función (3), para así estimar la integral definida (4) y después normalizar el estimado para que sea la integral definida (1).

$$g(x) = \frac{2f(x)}{\pi} \tag{3}$$

$$\int_3^7 g(x)dx \tag{4}$$

Utilizar la función (3) es posible porque es una función de distribución válida, ya que su integral es igual a 1 (5).

$$\int_{-\infty}^{\infty} \frac{2}{\pi} f(x) \mathrm{d}x = 1 \tag{5}$$

Las modificaciones que se le realizaron al código revisado en clase [3] fueron; agregar un ciclo `for` para variar la `muestra` en tamaños de 1000, 10000, 100000, 1000000 y 10000000, agregar otro ciclo `for` para hacer 30 réplicas con cada tamaño de `muestra` y agregar el código mostrado anteriormente para obtener la cantidad de decimales que coninciden entre el valor de la integral definida (1) calculado con el método Monte Carlo y el valor que se obtiene con el programa `Wolfram Alpha` para dicha integral, el cual es de 0,048834. A continuación se muestra el código modificado para esta práctica:

```
library(ggplot2)
desde = 3
hasta = 7
bueno = 0.048834 #numero de Wolfram Alpha
n = seq(1, 6, 1)
muestra = c(10^3, 10^4, 10^5, 10^6, 10^7) # puntitos en el cuadro
df = data.frame()

for (m in muestra){
  for (replica in 1:30){
    f <- function(x) { return(1 / (exp(x) + exp(-x))) } # funcion que piden
    g <- function(x) { return((2 / pi) * f(x)) } # normalizado a distr

    suppressMessages(library(distr)) # paquete
    generador  <- r(AbscontDistribution(d = g)) # creamos un generador
    valores <- generador(m) # generamos valores
    montecarlo = sum(valores >= desde & valores <= hasta) # checamos
    integral <- sum(montecarlo) / m # tasa: integral para g(x)
    resultado <- (pi / 2) * integral # integral para f(x) (renorm)
    for (i in n) {
      b = trunc(bueno*10^i)/10^i
      r = trunc(resultado*10^i)/10^i

      if (r == b) {
        deci = i
      } else {
        break
      }
    }
    datos <- c(m, replica, resultado, deci)
    df = rbind(df, datos)
    #cat(m, replica, resultado, deci,'\n')
  }
}

names(df) <- c("Muestra", "Replica", "Resultado", "Decimales")
df$Muestra = as.factor(df$Muestra)
ggplot(df, aes(x= Muestra, y= Decimales, fill= Muestra)) +
  geom_boxplot()+
  labs(x = "Muestra", y = "Decimales correctos") + #nombres
  scale_x_discrete(labels = c("1K", "10K", "100K", "1M", "10M"))+
  scale_fill_discrete(labels = c("10^3", "10^4", "10^5", "10^6", "10^7"))
```

Listing 2: Código para obtener la cantidad de decimales que coinciden al comparar el valor de la integral (1) calculado con el método Monte Carlo y el valor que se obtiene con el programa `Wolfram Alpha`.

Para analizar los resultados obtenidos se graficaron en un diagrama caja-bigote (Ver figura 1). En este diagrama podemos ver que al aumentar el tamaño de la muestra se obtiene como resultado un valor más preciso, es decir el valor de la integral con el método Monte Carlo es más parecido al resultado generado con el programa `Wolfram Alpha`. Por lo tanto se podría decir que a mayor tamaño de muestra, mayor es la precisión de decimales.

Para analizar si existe una relación entre la variación del tamaño de muestra y la cantidad de decimales correctos, primero se eligió utilizar la prueba estadística ANOVA de una vía, pero debido a los resultados obtenidos al revisar la normalidad de los datos, se eligió realizar la prueba estadística `Kruskal Wallis` [1].

.

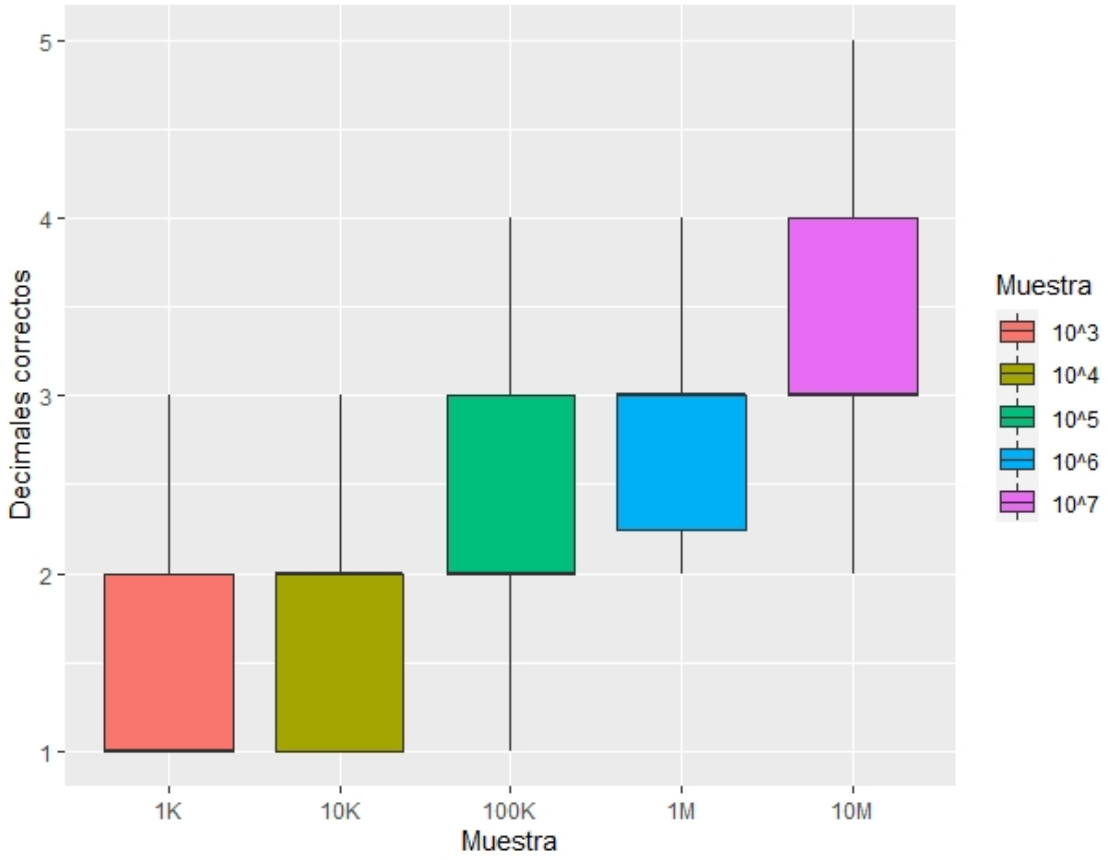

Figura 1: Cantidad de decimales que coinciden al variar el tamaño de la muestra en la comparación del valor de la integral (1) calculado con el método Monte Carlo y el valor que se obtiene del programa `Wolfram Alpha`.

En el cuadro 1 se resumen los resultados de la revisión de los supuestos para poder aplicar la prueba estadística ANOVA. El supuesto outliers se refiere a la cantidad de valores atípicos que existen en los grupos, la normalidad por grupos se obtuvo con la prueba de `Shapiro Wilk`, y la homogeneidad de varianza se obtuvo con la prueba de `Levene`.

Cuadro 1: Resultados del los supuestos para aplicar la prueba estadística ANOVA.

Outliers	0
Normalidad por grupo	1×10^3: $p = 2{,}92\times10^{-6}$ 1×10^4: $p = 7{,}59\times10^{-6}$ 1×10^5: $p = 1{,}58\times10^{-4}$ 1×10^6: $p = 3{,}26\times10^{-6}$ 1×10^7: $p = 3{,}51\times10^{-4}$
Homogeneidad de varianza	$p = 0{,}449$

En los resultados se observa que ningún valor de p es mayor a 0,05, por lo tanto no tienen normalidad, así que es necesario realizar la prueba estadística `Kruskal Wallis` ya que ésta es útil para cuando no se tiene normalidad en los grupos de datos.

Al realizar la prueba `Kruskal Wallis` se obtienen los resultados mostrados en el cuadro 2.

.

Cuadro 2: Resultados al aplicar la prueba estadística `Kruskal Wallis`.

Chi cuadrada	Valor de p
82,532	$2{,}2 \times 10^{-16}$

Hipótesis nula : Las medias son iguales en todos los grupos.

Hipótesis alternativa: Debido a que $p < 0{,}05$ se rechaza la hipótesis nula, es decir que si existen diferencias significativas entre las medias de los grupos.

Se entiende entonces que la variación del tamaño de la muestra si tiene un efecto significativo en la precisión de los decimales del resultado de la integral con el método Monte Carlo.

También podemos realizar la prueba de suma de rangos de `Wilcoxon` por pares [2], ya que esta se puede utilizar como una alternativa a la prueba `t de Student` cuando no se tiene normalidad en las muestras, de esta forma podemos observar los resultados de p y determinar si existen diferencias al comparar entre ellas cada una de las muestras (Ver cuadro 3).

Cuadro 3: Resultados al aplicar la prueba `Wilcoxon`.

Valor de p	1×10^3	1×10^4	1×10^5	1×10^6
1×10^4	0,1061	-	-	-
1×10^5	$4{,}7 \times 10^{-5}$	0,0011	-	-
1×10^6	$6{,}3 \times 10^{-8}$	4×10^{-7}	0,0327	-
1×10^7	$2{,}9 \times 10^{-9}$	$5{,}6 \times 10^{-9}$	$4{,}7 \times 10^{-5}$	0,0046

En los resultados de la prueba podemos ver que solamente las muestras de 1000 y 10000 tienen una $p > 0{,}05$, por lo tanto se puede determinar que solamente en esas dos muestras no existen diferencias entre ellas.

A continuación se muestra el código utilizado para realizar la prueba estadística `Kruskal Wallis`:

```
#Estadisticas descriptivas
df %>%
  group_by(Muestra) %>%
  get_summary_stats(Decimales, type = "mean_sd")

#SUPUESTOS PARA ANOVA
#1:Outliers
df %>%
  group_by(Muestra) %>%
  identify_outliers(Decimales)

#2:Normalidad por Shapiro
df %>%
  group_by(Muestra) %>%
  shapiro_test(Decimales)

#3:Homogeneidad de varianza con prueba Levene
df %>%
  levene_test(Decimales~Muestra)

#PRUEBA ESTADISTICA KRUSKAL WALLIS
kruskal.test(Decimales ~ Muestra, data = df)

#PRUEBA WILCOXON
pairwise.wilcox.test(df$Decimales, df$Muestra)
```

Listing 3: Código para la prueba estadística `Kruskal Wallis` y la prueba `Wilcoxon`.

.

3. Reto 1

El primer reto consiste en realizar el mismo objetivo de la práctica pero ahora para la estimación del valor de π de Kurt.

Se utilizó como base el código revisado en clase [4] para generar el valor de π mediante el método Monte Carlo. Las modificaciones que se le realizaron al código fueron practicamente las mismas que en la tarea base; agregar un ciclo `for` para variar la `muestra` en tamaños de 1000, 10000, 100000, 1000000 y 10000000, agregar otro ciclo `for` para hacer 30 réplicas con cada tamaño de `muestra` y agregar el código para obtener la cantidad de decimales que coninciden entre el valor de π calculado con el método Monte Carlo y el valor real de π, el cual se considero como 3,141592. A continuación se muestra el código modificado para este reto:

```
library(ggplot2)
bueno = 3.141592 #numero pi
n = seq(1, 6, 1)
muestra = c(10^3, 10^4, 10^5, 10^6, 10^7) # puntitos en el cuadro
df = data.frame()

for (muchos in muestra) {
  for (replica in 1:30){
    interior = 0
    for (r in 1:muchos) {
      x = runif(1, -1, 1)
      y = runif(1, -1, 1)
      d = sqrt(x*x + y*y)
      if (d < 1) {
        interior = interior + 1
      }
    }

    tasa = interior / muchos
    pi = 4 * tasa
    #print(pi)
    for (i in n) {
      b = trunc(bueno*10^i)/10^i
      r = trunc(pi*10^i)/10^i

      if (r == b) {
        deci = i
      } else {
        break
      }
    }
    datos <- c(muchos, replica, pi, deci)
    df = rbind(df, datos)
    #cat(muchos, replica, pi, deci,'\n')
  }
}

names(df) <- c("Muestra", "Replica", "Resultado", "Decimales")
df$Muestra = as.factor(df$Muestra)
ggplot(df, aes(x= Muestra, y= Decimales, fill= Muestra)) +
  geom_boxplot()+
  labs(x = "Muestra", y = "Decimales correctos") + #nombres
  scale_x_discrete(labels = c("1K", "10K", "100K", "1M", "10M"))+
  scale_fill_discrete(labels = c("10^3", "10^4", "10^5", "10^6", "10^7"))
```

Listing 4: Código para obtener la cantidad de decimales que coinciden al comparar el valor de π calculado con el método Monte Carlo y el valor real de π considerado como 3,141592.

Los resultados obtenidos se graficaron en un diagrama caja-bigote (Ver figura 2). En este diagrama también podemos notar que al aumentar el tamaño de la muestra se obtiene como resultado un valor más preciso, es decir el valor de π calculado con el método Monte Carlo es más parecido al valor real de π cuando se tiene un tamaño de muestra mayor.

.

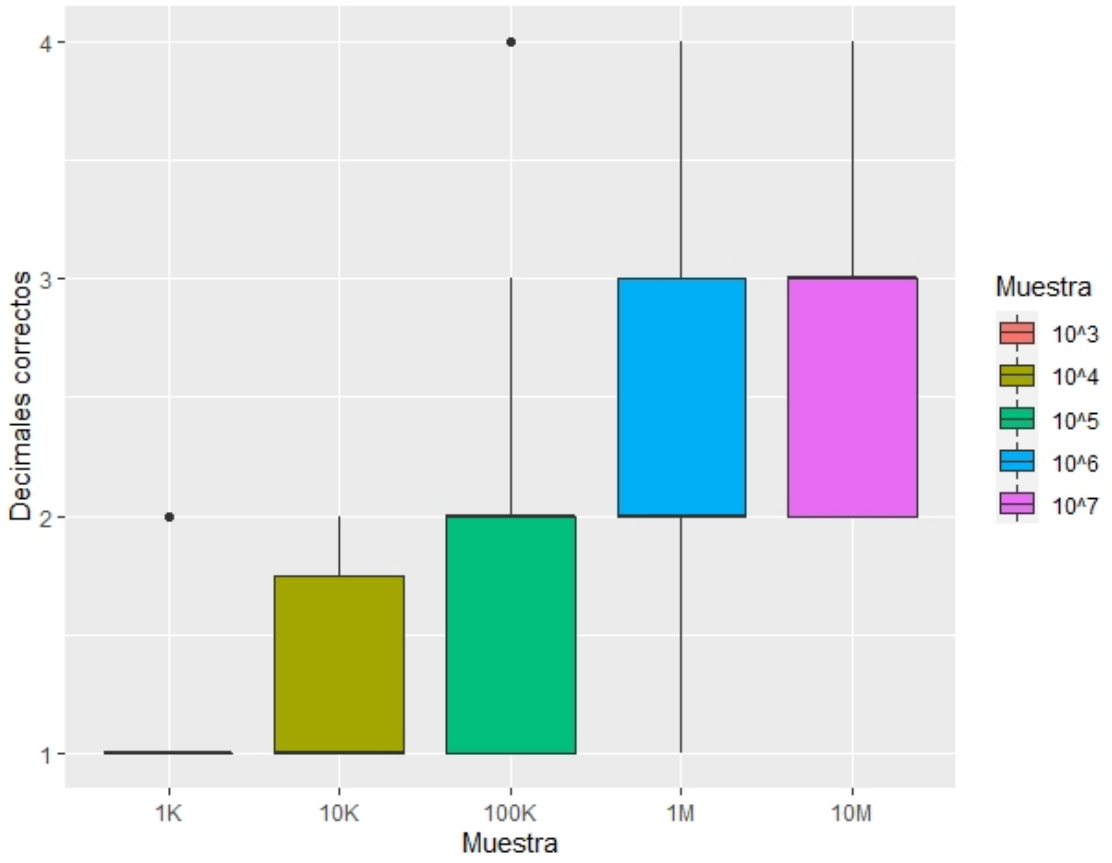

Figura 2: Cantidad de decimales que coinciden al variar el tamaño de la muestra en la comparación del valor π calculado con el método Monte Carlo y el valor de π considerado como 3,141592.

Para analizar si existe una relación entre la variación del tamaño de muestra y la cantidad de decimales correctos, se realizó la prueba estadística `Kruskal Wallis` debido a que los resultados obtenidos no presentan normalidad. En el cuadro 4 se resumen los resultados de la revisión de los supuestos para poder aplicar la prueba estadística. La normalidad por grupos se obtuvo con la prueba de `Shapiro Wilk`, y la homogeneidad de varianza se obtuvo con la prueba de `Levene`.

Cuadro 4: Resultados del los supuestos para aplicar la prueba estadística.

Outliers	10
Normalidad por grupo	1×10^3: $p = 1{,}78\times10^{-10}$ 1×10^4: $p = 2{,}09\times10^{-8}$ 1×10^5: $p = 1{,}32\times10^{-4}$ 1×10^6: $p = 1{,}54\times10^{-3}$ 1×10^7: $p = 3{,}11\times10^{-6}$
Homogeneidad de varianza	$p = 1{,}12\times10^{-5}$

Al realizar la prueba `Kruskal Wallis` se obtienen los resultados mostrados en el cuadro 5.

Hipótesis nula : Las medias son iguales en todos los grupos.

Hipótesis alternativa: Debido a que $p < 0{,}05$ se rechaza la hipótesis nula, es decir que si existen diferencias significativas entre las medias de los grupos.

También se realizó la prueba de suma de rangos de `Wilcoxon` por pares, los resultados de esta prueba se resumen el cuadro 6.

Cuadro 5: Resultados al aplicar la prueba estadística `Kruskal Wallis`.

Chi cuadrada	Valor de p
75,087	$1{,}91 \times 10^{-15}$

Cuadro 6: Resultados al aplicar la prueba `Wilcoxon`.

Valor de p	1×10^3	1×10^4	1×10^5	1×10^6
1×10^4	0,2006	-	-	-
1×10^5	$5{,}6 \times 10^{-4}$	0,0252	-	-
1×10^6	$3{,}2 \times 10^{-8}$	$2{,}4 \times 10^{-6}$	0,0478	-
1×10^7	$9{,}7 \times 10^{-11}$	$4{,}1 \times 10^{-9}$	0,0015	0,2228

En los resultados mostrados en el cuadro 6 podemos ver que solamente las comparaciones de 1000 con 10000 y 1000000 con 10000000 tienen una $p > 0{,}05$, por lo tanto se puede determinar que solamente en estas comparaciones no existen diferencias significativas.

A continuación se muestra el código utilizado para realizar la prueba estadística `Kruskal Wallis`:

```
#Estadisticas descriptivas
df %>%
  group_by(Muestra) %>%
  get_summary_stats(Decimales, type = "mean_sd")

#SUPUESTOS PARA ANOVA
#1:Outliers
df %>%
  group_by(Muestra) %>%
  identify_outliers(Decimales)

#2:Normalidad por Shapiro
df %>%
  group_by(Muestra) %>%
  shapiro_test(Decimales)

#3:Homogeneidad de varianza con prueba Levene
df %>%
  levene_test(Decimales~Muestra)

#PRUEBA ESTADISTICA KRUSKAL WALLIS
kruskal.test(Decimales ~ Muestra, data = df)

#PRUEBA WILCOXON
pairwise.wilcox.test(df$Decimales, df$Muestra)
```

Listing 5: Código para la prueba estadística `Kruskal Wallis` y la prueba `Wilcoxon`.

.

4. Conclusión

Con base en los diagramas caja-bigote y los resultados obtenidos en las pruebas estadísticas `Kruskal Wallis` puedo concluir que el hecho de variar el tamaño de la muestra si influye en la precisión de los resultados que se obtienen al evaluar funciones con el método Monte Carlo, también considero que si se utiliza un tamaño de muestra mayor a diez millones se tendría una mejor presición, esto lo intenté realizar pero debido a que la muestra es demasiado grande mi computadora se inhibió por varios minutos, por lo cual decidí no tomar muestras mayores a diez millones.

En general ésta práctica fue mucho de mi agrado porque la pude realizar sin grandes complicaciones como me sucedió en prácticas anteriores.

.

Referencias

[1] José Antonio: Estadística Aplicada. Kruskall-wallis en rstudio, 2020. URL https://www.youtube.com/watch?v=WEjudFpbCcE&t=996s&ab_channel=Jos%C3%A9Antonio%3AEstad%C3%ADsticaAplicada.

[2] RDocumentation. Wilcox.test: Wilcoxon rank sum and signed rank tests. URL https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/wilcox.test.

[3] Elisa Schaeffer. Montecarlo, 2021. URL https://github.com/satuelisa/Simulation/blob/master/MonteCarlo/nopar.R.

[4] Elisa Schaeffer. Montecarlo, 2021. URL https://github.com/satuelisa/Simulation/blob/master/MonteCarlo/pi.R.