# Métodos de MonteCarlo

Curso 2023

Obligatorio 3 - Ejercicio 5.1

Grupo 05

Juan Manuel Varela

Nicolás Farías

# Contenido

# Descripción del Problema

Entrega 3: Ejercicio 5.1: [tarea en grupo]

Problema (enunciado en sesión 3, ejercicio 3.1): Se desea estimar el volumen de una región R de $[0, 1]^6$ definida por todos los puntos de la hiper-esfera de centro (0.45, 0.5, 0.6, 0.6, 0.5, 0.45) y radio 0.35, que además cumplan las restricciones siguientes: $3x_1 + 7x_4 \leq 5$; $x_3 + x_4 \leq 1$; $x_1 - x_2 - x_5 + x_6 \geq 0$.

- Parte a: Compartir en el grupo los códigos desarrollados para la parte a, validarlos revisando los códigos, y verificando si las salidas para tamaños de muestra de $10^6$ son consistentes. Indicar si se detectaron errores en los mismos, y en ese caso dar los códigos corregidos. Elegir uno de los códigos para las partes siguientes, explicar los motivos de la selección.

- Parte b: calcular la cantidad de replicaciones a realizar para garantizar un error menor a $1.0 \times 10^{-4}$ con probabilidad 0.95, utilizando el criterio de peor caso de Hoeffding.

- Parte c: utilizando el código elegido en la parte a, y la cantidad de replicaciones definida en el punto anterior, calcular el intervalo de confianza de nivel 0.95 utilizando el criterio de Chebyshev, y el criterio de Agresti-Coull. Comparar el ancho de estos intervalos entre sí y con el criterio de error manejado en el punto previo.

# Solución

## Parte a

No se detectaron errores en los códigos revisados. Al estar implementados en el mismo lenguaje (Python) y utilizando la misma biblioteca para generar los números pseudoaleatorios, se pudo ejecutar los dos con la misma semilla y se obtuvieron resultados idénticos. Debido a algunas diferencias en la implementación, uno de los códigos demoraba un poco menos al ejecutar ambos con el mismo valor de n (cantidad de replicaciones). Por lo tanto decidimos elegir este último para las partes siguientes.

## Parte b

Aplicamos la fórmula:

- $nH(\varepsilon, \delta) = \lceil \ln(2/\delta) / (2\varepsilon^2) \rceil$ con valores $\varepsilon = 0.0001$ y $\delta = 0.05$

La expresión de Python utilizada para el cálculo fue:

- `nH = math.ceil(math.log(2 / delta) / (2 * epsilon**2))`

El resultado que obtuvimos fue: 184443973.

## Parte c

Al ejecutar el código elegido en la parte a, con un valor de n = 184443973, obtuvimos un valor de S = 52627, donde S es la cantidad de puntos sorteados que caen en la región R. Por lo tanto pasamos a calcular los intervalos utilizando los siguientes valores:

- n = 184443973
- S = 52627
- δ = 0.05

### Intervalo de Chebyshev

$$\omega_1(S, n, \delta^{-1/2}) = \frac{S + \delta/2 - \delta^{-1/2}\sqrt{\delta/4 + S(n-S)/n}}{n + \delta}$$

$$\omega_2(S, n, \delta^{-1/2}) = \frac{S + \delta/2 + \delta^{-1/2}\sqrt{\delta/4 + S(n-S)/n}}{n + \delta}$$

$\omega 1(S, n, \delta^{-1/2}) = 0.000280$
$\omega 2(S, n, \delta^{-1/2}) = 0.000291$
Ancho del intervalo = 0.000011

## Intervalo de Agresti-Coull

El intervalo está definido por la siguiente fórmula:

$$CI_{AC} = \tilde{p} \pm \kappa(\tilde{p}\tilde{q})^{1/2}\tilde{n}^{-1/2}$$

Donde:

$$\tilde{p} = \tilde{S}/\tilde{n}$$
$$\kappa = \Phi^{-1}(1 - \delta/2)$$
$$\tilde{S} = S + \kappa^2/2$$
$$\tilde{n} = n + \kappa^2$$
$$\tilde{q} = 1 - \tilde{p}$$

$AC_1 = 0.000283$
$AC_2 = 0.000288$

Ancho del intervalo = 0.000005

## Comparación

Se puede observar que el intervalo de Chebyshev contiene completamente al intervalo de Agresti-Coull, teniendo a su vez un ancho de aproximadamente el doble. Por lo tanto, podemos decir que el criterio de Chebyshev es más conservador.

En la parte b, calculamos la cantidad de replicaciones necesarias para garantizar un error menor a 0.0001, con un nivel de confianza mayor o igual a 0.95. Obtuvimos un valor de n = 184443973.
En la parte c estamos haciendo el camino en el otro sentido. Es decir, dado ese valor de n, y manteniendo el nivel de confianza de 0.95, calculamos los intervalos que nos dan una idea del margen de error. Por lo tanto podemos comparar el ancho de los intervalos con el valor de $\varepsilon = 0.0001$ de la parte b.
Para ambos intervalos se puede ver que el ancho es bastante menor a 0.0001.

# Anexo

## Detalles Técnicos

El código se implementó en lenguaje Python (versión 3.11.2). Al volver a ejecutar el código de la parte a, con la cantidad de replicaciones calculadas en la parte b, se mantuvo la semilla 12345.
Se ejecutaron los programas en una notebook con las siguientes características: MacBook Pro 2017, Procesador 3.1 GHz Quad-Core Intel Core i7, Memoria RAM 16 GB 2133 MHz LPDDR3, Sistema Operativo macOS Ventura 13.2.1

## Log de Ejecución

Se muestra a continuación una captura de pantalla de la ejecución del programa para el cálculo de la parte b.

```
n para el criterio de Hoeffding:184443973
```

Se muestra a continuación una captura de pantalla de la ejecución del programa seleccionado en la parte a, con el valor de n calculado en la parte b.

```
S:     52627
N: 184443973
Estimador:  0.000285
Desviación Estándar:  0.000001
Tiempo(segundos):392.881355
```

Se muestra a continuación una captura de pantalla de la ejecución del programa desarrollado para los cálculos de la parte c.

```
Intervalo de Chebyshev:(0.000280, 0.000291) Ancho:0.000011
Intervalo de Agresti-Coull:(0.000283, 0.000288) Ancho:0.000005
```

## Código Fuente

Parte a:

```
import sys
import random
import math
import time
```

```
# recibo N como parámetro de línea de comando
n = int(sys.argv[1])

# datos de la esfera
centro = [0.45, 0.5, 0.6, 0.6, 0.5, 0.45]
radio = 0.35
radio2 = radio**2


def pertenece_a_esfera(x):
    """Verifica si el punto x pertenece a la esfera."""
    return sum((xi - ci)**2 for xi, ci in zip(x, centro)) <= radio2


def cumple_restricciones(x):
    """3x1 + 7x4 ≤ 5; x3 +x4 ≤ 1; x1 −x2 −x5 +x6 ≥ 0."""
    return (
        3*x[0] + 7*x[3] <= 5
        and x[2] + x[3] <= 1
        and x[0] - x[1] - x[4] + x[5] >= 0)


# tiempo inicial
start = time.time()

# inicialización
esperanza = 0

# inicializa generador de números con semilla
random.seed(12345)

for i in range(n):
    x = [random.random() for j in range(6)]
    if pertenece_a_esfera(x) and cumple_restricciones(x):
        esperanza += 1

print(f'S:{esperanza:10}')

esperanza = esperanza / n
varianza = esperanza * (1 - esperanza) / (n - 1)
desviacion = math.sqrt(varianza)
```

```
# tiempo final
end = time.time()

print(f'N:{n:10}\
    \nEstimador:{esperanza:10f}\
    \nDesviación Estándar:{desviacion:10f}\
    \nTiempo(segundos):{end - start:10f}')
```

Parte b:

```
import math

delta = 0.05
epsilon = 0.0001

# Cantidad de replicaciones para el criterio de Hoeffding
nH = math.ceil(math.log(2 / delta) / (2 * epsilon**2))
print(f'n para el criterio de Hoeffding:{nH}')
```

Parte c:

```
import math
import scipy

delta = 0.05

# Intervalo de Chebyshev
S = 52627
n = 184443973
delta = 0.05

z = S
beta = delta ** (-0.5)

w1 = (z + beta**2 / 2 - beta * math.sqrt(beta**2 / 4 + z * (n - z) /
n)) / (n + beta**2)
w2 = (z + beta**2 / 2 + beta * math.sqrt(beta**2 / 4 + z * (n - z) /
n)) / (n + beta**2)
print(f'Intervalo de Chebyshev:({w1:f}, {w2:f}) Ancho:{w2-w1:f}')

# Intervalo de Agresti-Coull
k = scipy.stats.norm.ppf(1 - delta/2)
X_ = S + k**2 / 2
```

```
n_ = n + k**2
p_ = X_ / n_
q_ = 1 - p_
ac1 = p_ - k * (p_ * q_)**0.5 * n_**(-0.5)
ac2 = p_ + k * (p_ * q_)**0.5 * n_**(-0.5)
print(f'Intervalo de Agresti-Coull:({ac1:f}, {ac2:f})
Ancho:{ac2-ac1:f}')
```