Actividad 1: Simulación Estocástica

Curso: Temas Selectos I: O25 LAT4032 1
Profesor: Rubén Blancas Rivera
Alumnos: Ximena Bravo, Heriberto Espino, Celeste Nuñez
Universidad: Universidad de las Américas Puebla
Fecha: 22 de octubre de 2025

Funciones para graficar

[1]:
```python
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(style='whitegrid')
```

[2]:
```python
def histograma(muestras, montecarlo, bins=50):
    fig, ax = plt.subplots(figsize=(10, 6))
    n, _, _ = ax.hist(muestras, bins=bins, alpha=0.7)
    ax.axvline(montecarlo, color='r', linestyle='--',
               linewidth=2, zorder=3, label=f'Monte Carlo {montecarlo:.4f}')
    xmin, xmax = ax.get_xlim()
    if montecarlo < xmin or montecarlo > xmax:
        ax.set_xlim(min(xmin, montecarlo), max(xmax, montecarlo))
    ax.set_title('Muestras')
    ax.set_xlabel('Valor')
    ax.set_ylabel('Frecuencia')
    ax.legend(facecolor='white', edgecolor='none')
    plt.tight_layout()
    plt.show()
```

[3]:
```python
def tlc(muestras, valor_verdadero=None):

    k = len(muestras)
    medias = np.cumsum(muestras) / np.arange(1, k + 1)

    plt.figure(figsize=(10, 6))
    plt.plot(medias)
    plt.xlabel('Número de simulaciones')
    plt.ylabel('Media acumulada')
    plt.title(f'Convergencia de la media Monte Carlo')
    plt.axhline(y=medias[-1], color='r', linestyle='--', label=f'Media final␣
 ↪{medias[-1]:.4f}')
    if valor_verdadero is not None:
        plt.axhline(y=valor_verdadero, color='g', linestyle='--', label=f'Valor␣
 ↪verdadero {valor_verdadero:.4f}')
    plt.legend(facecolor='white', edgecolor='none')
    plt.gca().yaxis.set_ticks_position('right')
    plt.gca().yaxis.set_label_position('right')
    plt.show()
```

Ejercicio 1

Si $x_0 = 5$ y $x_n = 2x_{n-1} \mod 150$. Encontrar $x_1, \ldots, x_{10}$.

$$x_n = ax_{n-1} \mod m$$

$$ 10 = 2 $\cdot$5 $\mod 150$\

20 = 10 $\cdot$5 $\mod 150$\

40 = 20 $\cdot$5 $\mod 150$\

80 = 40 $\cdot$5 $\mod 150$\

10 = 80 $\cdot$5 $\mod 150$\

$\vdots$

$$

[4]:
```python
pseudoaleatorios = []

x0 = 5
a = 2
m = 150

for i in range(10):
    xn = (a * x0) % m
    x0 = xn
    pseudoaleatorios.append(xn)

pseudoaleatorios
```

[4]:
```
[10, 20, 40, 80, 10, 20, 40, 80, 10, 20]
```

Ejercicio 3

$$\int_0^1 \exp(e^x)\,dx$$

Sea

$$\theta = \int_0^1 \exp\left(e^x\right)dx.$$

Reescritura como valor esperado con $U \sim \mathrm{Unif}(0,1)$:

$$\theta = \mathbb{E}\left[\exp\left(e^U\right)\right].$$

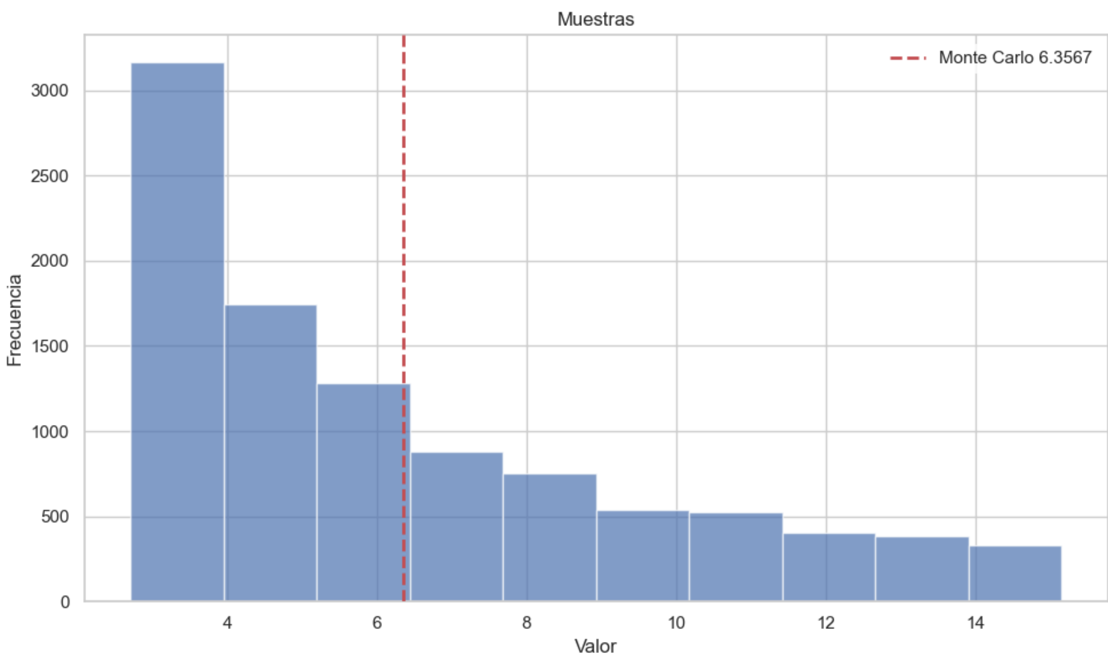
Estimador Monte Carlo con $u_1, \ldots, u_K \overset{\text{iid}}{\sim} \mathrm{Unif}(0,1)$:

$$\widehat{\theta}_K = \frac{1}{K}\sum_{i=1}^{K} \exp\left(e^{u_i}\right).$$

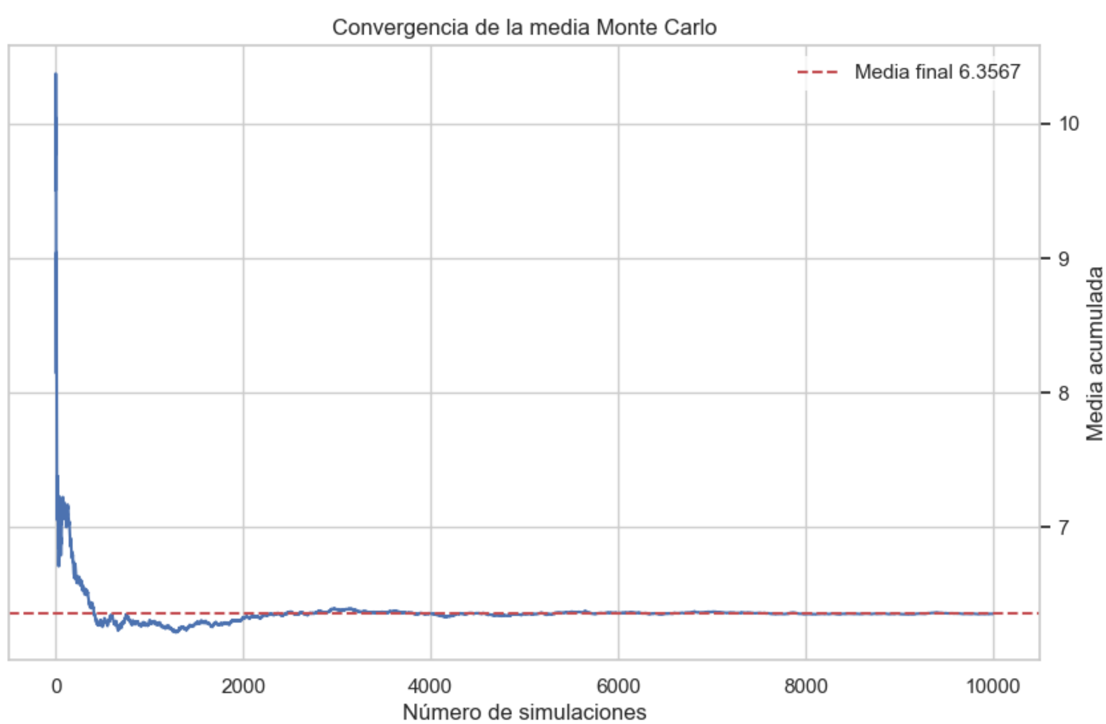
[5]:
```
def h(u):
    return np.exp(np.exp(u))

k = 10000

u = np.random.random(k)

muestras = h(u)
montecarlo = muestras.mean()
montecarlo
```

[5]: np.float64(6.35668829491894)

[6]:
```
histograma(muestras, montecarlo, bins=10)
```




```
[7]: tlc(muestras)
```



Ejercicio 5

$$\int_{-2}^{2} e^{x+x^2}\, dx$$

Sea

$$\theta = \int_{-2}^{2} e^{x+x^2}\, dx.$$

Cambio de variable a ([0,1]):

$$u = \frac{x-(-2)}{2-(-2)} = \frac{x+2}{4}, \qquad x = -2+4u, \qquad dx = 4\,du.$$

Entonces

$$\theta = \int_{0}^{1} 4\,\exp\big[(-2+4u)+(-2+4u)^2\big]\,du.$$

Forma de valor esperado con $U \sim \mathrm{Unif}(0,1)$:

$$\theta = \mathbb{E}[g(U)], \qquad g(u) = 4\,\exp\big[(-2+4u)+(-2+4u)^2\big].$$

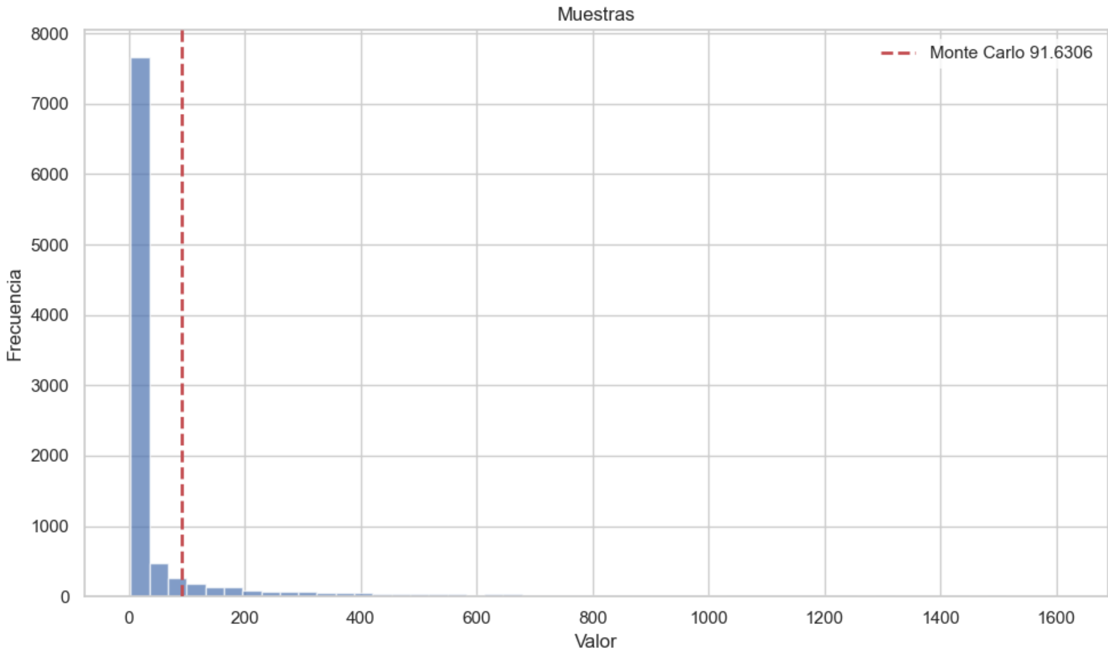
Estimador Monte Carlo:

$$\widehat{\theta}_K = \frac{1}{K}\sum_{i=1}^{K} g(u_i), \qquad u_i \overset{iid}{\sim} \mathrm{Unif}(0,1).$$

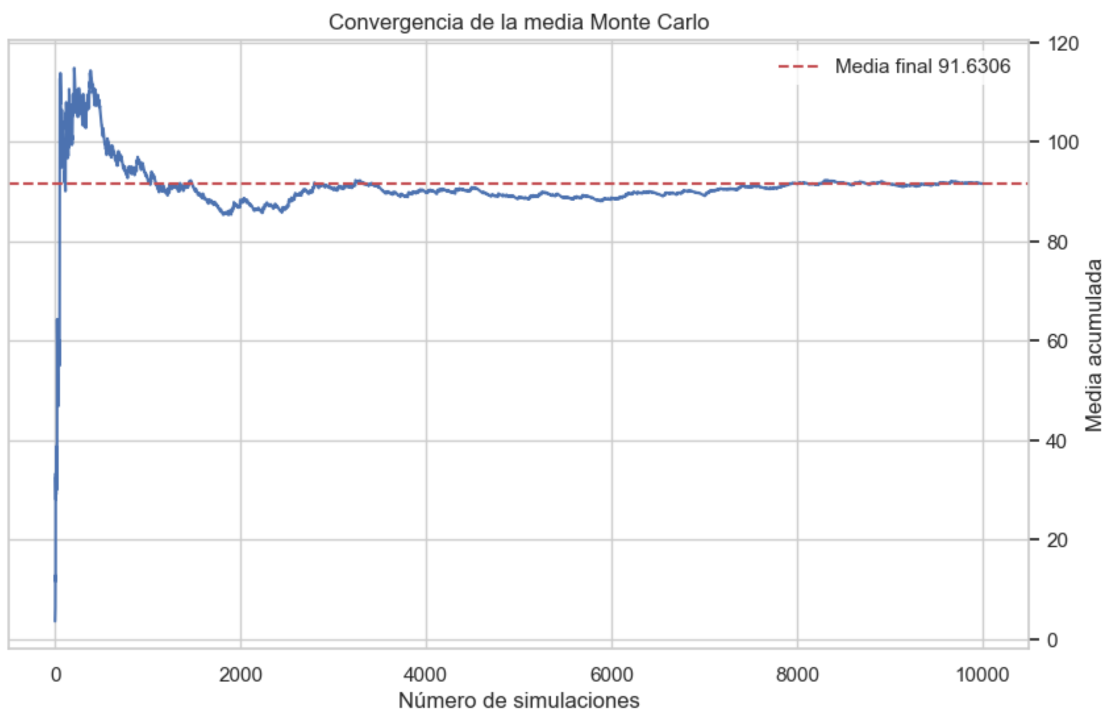
[8]:
```
def h(u):
    return (b-a)*np.exp(a+(b-a)*u + (a+(b-a)*u)**2)

k = 10000
a = -2
b = 2
u = np.random.random(k)

muestras = h(u)
montecarlo = muestras.mean()
montecarlo
```

[8]:
```
np.float64(91.6305599447782)
```

[9]:
```
histograma(muestras, montecarlo)
```



[10]:
```
tlc(muestras)
```



Ejercicio 7

$$\int_0^\infty \frac{x}{(1+x^2)^2}\,dx$$

Estimación Monte Carlo

Sea:

$$\theta = \int_0^\infty \frac{x}{(1+x^2)^2}\,dx.$$

Cambio:

$$y = \frac{1}{x+1}, \qquad dy = -\frac{dx}{(x+1)^2} = -y^2\,dx.$$

Entonces:

$$\theta = \int_0^1 h(y)\,dy, \qquad h(y) = \frac{g\left(\frac{1}{y}-1\right)}{y^2}, \quad g(x) = \frac{x}{(1+x^2)^2}.$$

Forma de esperanza con $U \sim \mathrm{Unif}(0,1)$:

$$\theta = \mathbb{E}[\,h(U)\,].$$

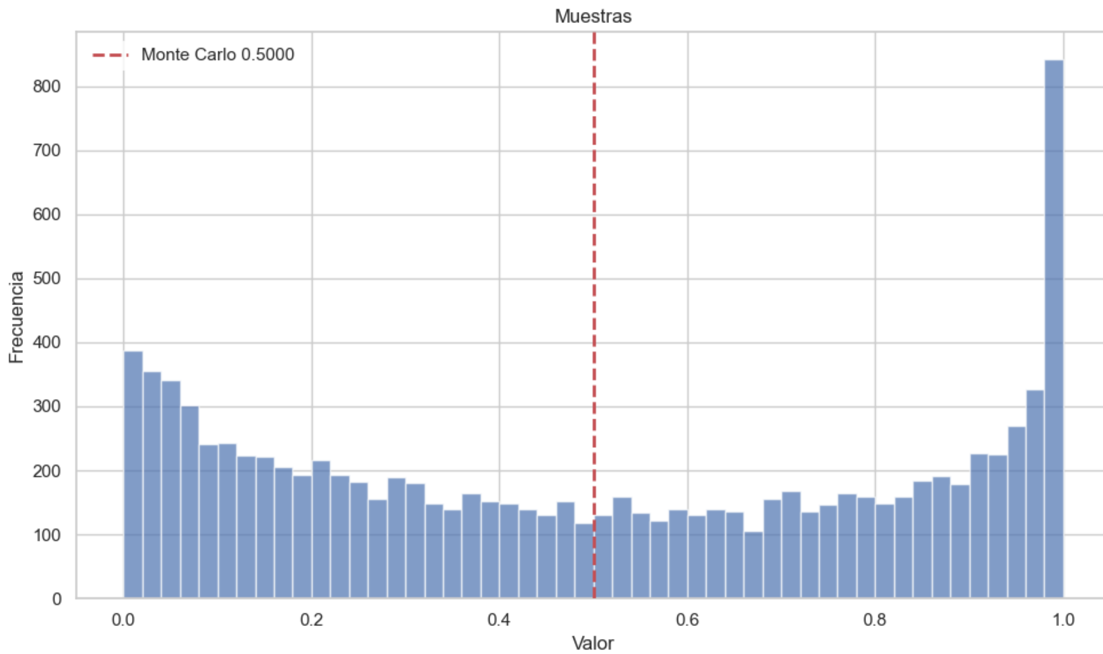
Estimador Monte Carlo:

$$\widehat{\theta}_K = \frac{1}{K}\sum_{i=1}^{K} h(u_i), \quad u_i \overset{\text{iid}}{\sim} \mathrm{Unif}(0,1).$$

[11]:
```
def g(x):
    return x / (1 + x**2)**2

def h(u):
    return g(1/u-1)/u**2

k = 10_000

u = np.random.random(k)
muestras = h(u)
montecarlo = muestras.mean()
montecarlo
```

[11]: np.float64(0.5000219155164649)

[12]:
```
histograma(muestras, montecarlo)
```


Muestras
Monte Carlo 0.5000
Frecuencia
800
700
600
500
400
300
200
100
0
0.0
0.2
0.4
0.6
0.8
1.0
Valor

Calculo analítico

Sea

$$\theta = \int_0^\infty \frac{x}{(1+x^2)^2}\,dx.$$

Integral impropia:

$$\theta = \lim_{b\to\infty} \int_0^b \frac{x}{(1+x^2)^2}\,dx.$$

Sustitución $u = 1 + x^2 \Rightarrow du = 2x\,dx$: cuando $x = 0 \Rightarrow u = 1$, cuando $x = b \Rightarrow u = 1 + b^2$. Entonces

$$\int_0^b \frac{x}{(1+x^2)^2}\,dx = \frac{1}{2}\int_1^{1+b^2} u^{-2}\,du.$$

Primitiva:

$$\int u^{-2}\,du = -u^{-1} + C.$$

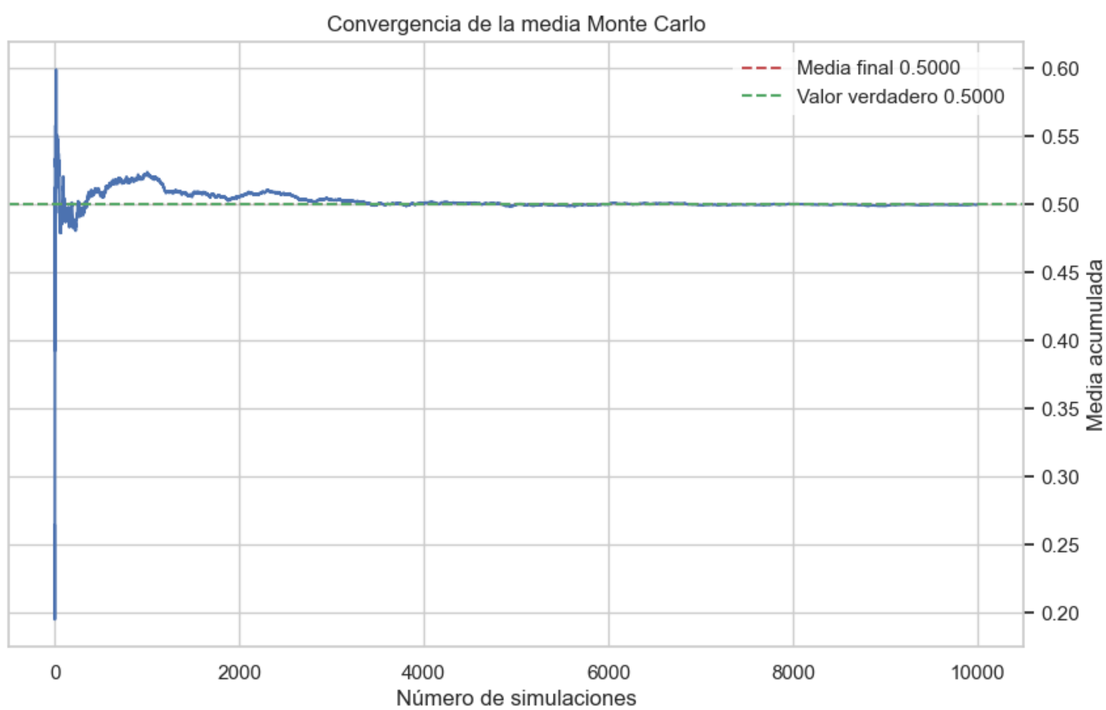
Evaluación:

$$\frac{1}{2}\Big[-u^{-1}\Big]_1^{1+b^2} = \frac{1}{2}\left(-\frac{1}{1+b^2} + 1\right).$$

Límite:

$$\theta = \lim_{b\to\infty} \frac{1}{2}\left(1 - \frac{1}{1+b^2}\right) = \frac{1}{2}.$$

[13]:
```
tlc(muestras, 1/2)
```



Ejercicio 9

$$\int_0^1 \int_0^1 e^{(x+y)^2}\, dy\, dx$$

Sea:

$$\theta = \int_0^1 \int_0^1 e^{(x+y)^2}\, dy\, dx.$$

Entonces:

$$\theta = \int_0^1 \int_0^1 g(x_1, x_2)\, dx_1\, dx_2, \qquad g(x_1, x_2) = e^{(x_1+x_2)^2}.$$

Sabemos que:

$$\theta = \mathbb{E}\big[g(U_1, U_2)\big], \quad U_1, U_2 \overset{iid}{\sim} \mathrm{Unif}(0,1).$$

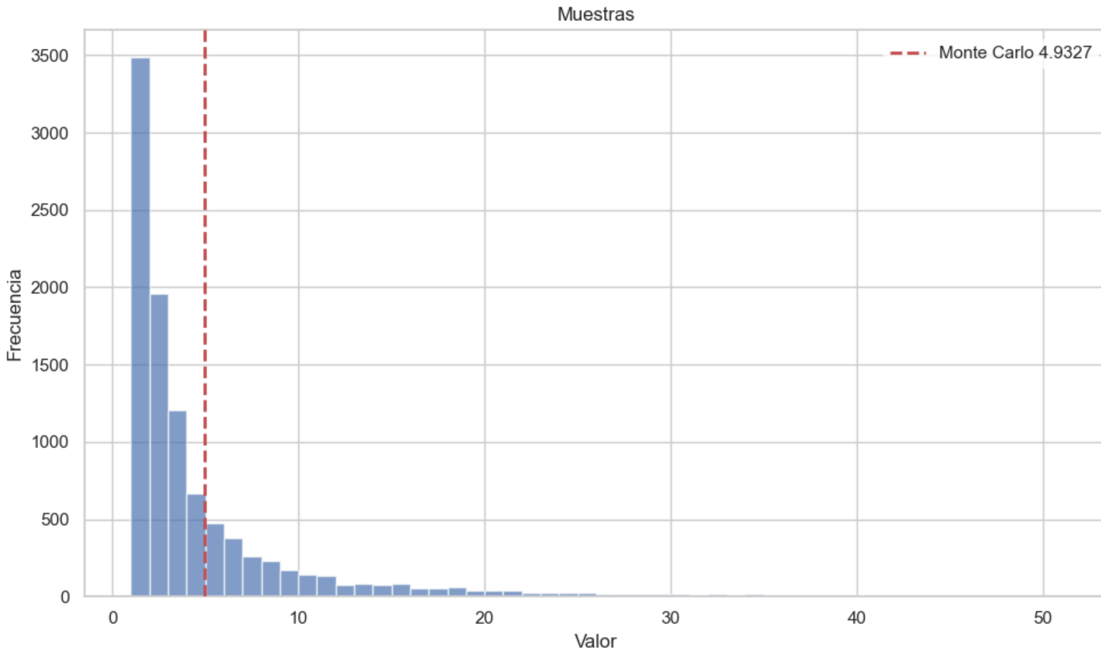
Estimador Monte Carlo:

$$\widehat{\theta}_k = \frac{1}{k}\sum_{i=1}^{k} g(u_{i1}, u_{i2}) = \frac{1}{k}\sum_{i=1}^{k} \exp\big((u_{i1}+u_{i2})^2\big), \quad (u_{i1}, u_{i2}) \overset{iid}{\sim} \mathrm{Unif}(0,1).$$

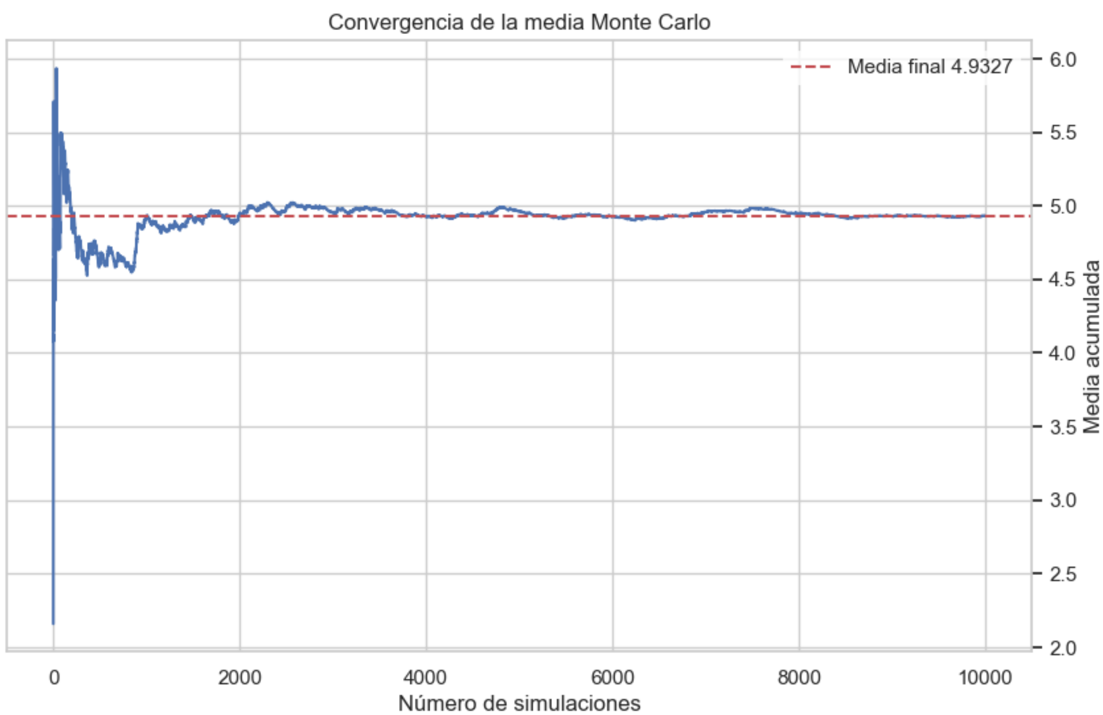
[14]:
```
def h(u1, u2):
    return np.exp((u1 + u2)**2)

k = 10000

u1 = np.random.random(k)
u2 = np.random.random(k)
muestras = h(u1, u2)
montecarlo = muestras.mean()
montecarlo
```

[14]:
```
np.float64(4.932739519320463)
```

[15]:
```
histograma(muestras, montecarlo)
```



[16]:
```
tlc(muestras)
```



Ejercicio 11

Usar simulación para aproximar $\mathrm{Cov}(U, e^U)$, donde $U \sim \mathcal{U}(0,1)$. Comparar con la respuesta exacta.

Por definición,

$$\mathrm{Cov}(X,Y) = \mathbb{E}\big[(X - \mathbb{E}X)(Y - \mathbb{E}Y)\big].$$

Expansión lineal:

$$\mathrm{Cov}(X,Y) = \mathbb{E}[XY] - \mathbb{E}[X]\ \mathbb{E}[Y].$$

Aplicando a $X = U$ y $Y = e^U$:

$$\mathrm{Cov}(U, e^U) = \mathbb{E}\big[Ue^U\big] - \mathbb{E}[U]\ \mathbb{E}[e^U].$$

Estimación Monte Carlo

Sea $u_1, \ldots, u_K \overset{iid}{\sim} \mathrm{Unif}(0,1)$. Entonces

$$\widehat{\mu}_U = \frac{1}{K}\sum_{i=1}^{K} u_i, \qquad \widehat{\mu}_e = \frac{1}{K}\sum_{i=1}^{K} e^{u_i}, \qquad \widehat{m} = \frac{1}{K}\sum_{i=1}^{K} u_i e^{u_i}.$$

Entonces:

$$\widehat{\mathrm{Cov}}^{(MC)} = \widehat{m} - \widehat{\mu}_U \, \widehat{\mu}_e$$

Cálculo analítico

$\mathbb{E}[U]$

$$\int_0^1 u\,du = \left[\frac{u^2}{2}\right]_0^1 = \frac{1^2}{2} - \frac{0^2}{2} = \frac{1}{2}.$$

$\mathbb{E}[e^U]$

$$\int_0^1 e^u\,du = [e^u]_0^1 = e^1 - e^0 = e - 1.$$

$\mathbb{E}[Ue^U]$ **(por partes)**

$$\int_0^1 x\,e^x\,dx, \qquad \begin{cases} u = x & \Rightarrow\ du = dx, \\ dv = e^x\,dx & \Rightarrow\ v = e^x. \end{cases}$$

$$\int_a^b u\,dv = \Big[u\,v\Big]_a^b - \int_a^b v\,du.$$

$$\Big[x\,e^x\Big]_0^1 - \int_0^1 e^x\,dx = \big(1 \cdot e - 0 \cdot 1\big) - \Big[e^x\Big]_0^1 = e - (e-1) = 1.$$

Covarianza

$$\mathrm{Cov}(U, e^U) = \mathbb{E}[Ue^U] - \mathbb{E}[U]\mathbb{E}[e^U] = 1 - \frac{1}{2}(e-1) = \frac{3-e}{2} = 0.140859086$$

[17]:
```
def valor_esperado_1(u):
    return u * np.exp(u)

def valor_esperado_2(u):
    return u

def valor_esperado_3(u):
    return np.exp(u)

k = 1000000

u = np.random.random(k)

montecarlo = valor_esperado_1(u).mean() - valor_esperado_2(u).mean() *␣
 ↪valor_esperado_3(u).mean()
```

[18]:
```
montecarlo
```

[18]: np.float64(0.14075041201920013)

[19]:
```
1 - 1/2*(np.e -1)
```

```
[19]: 0.14085908577047745
```

Ejercicio 13

Para variables aleatorias uniformes $U_1, U_2, \ldots$ definir

$$N = \min\left\{n : \sum_{i=1}^{n} U_i > 1\right\}.$$

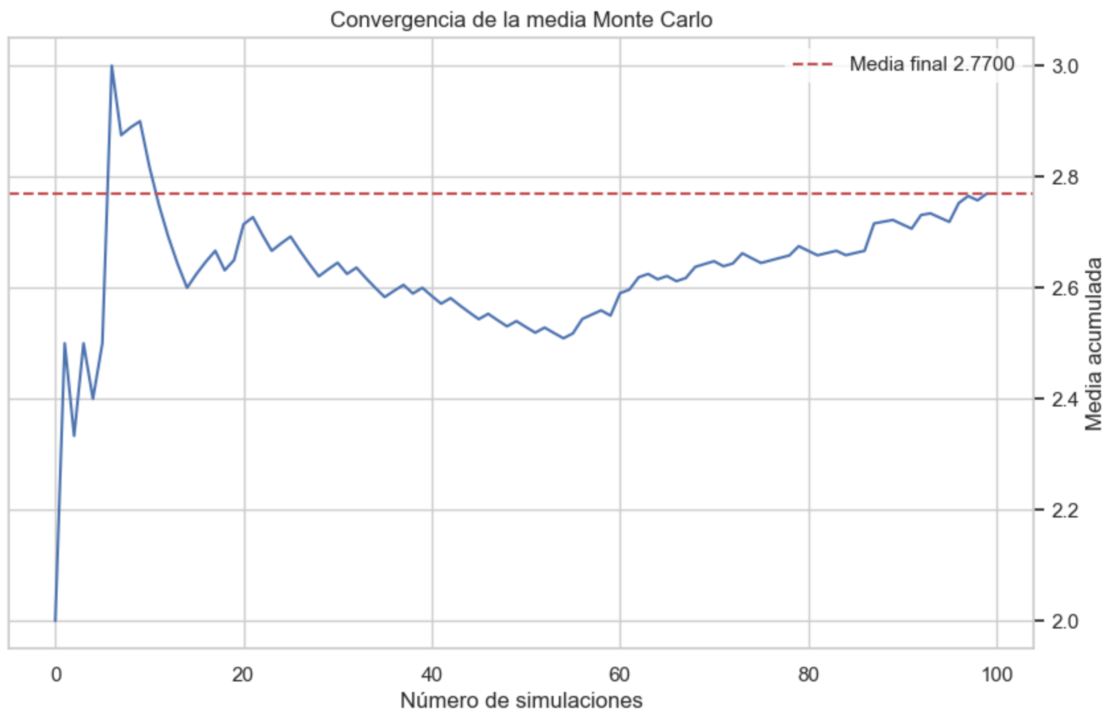
Estimar $\mathbb{E}[N]$ por simulación con: a) 100 valores, b) 1000 valores, c) 10000 valores, d) Discutir el valor esperado.

```
PSEUDOCÓDIGO - MINIMO_N(k)
total_contadores ← 0
PARA i ← 1 HASTA k HACER:
    suma ← 0
    contador ← 0
    MIENTRAS suma < 1 HACER:
        contador ← contador + 1
        suma ← UNIFORME(0,1)
    total_contadores ← total_contadores + contador
RETORNAR total_contadores / k
```

[20]:
```python
def minimo_N(k):
    lista_contadores = []
    for _ in range(k):
        suma = 0
        contador = 0
        while suma < 1:
            contador += 1
            suma += np.random.random()
        lista_contadores.append(contador)
    return lista_contadores
```

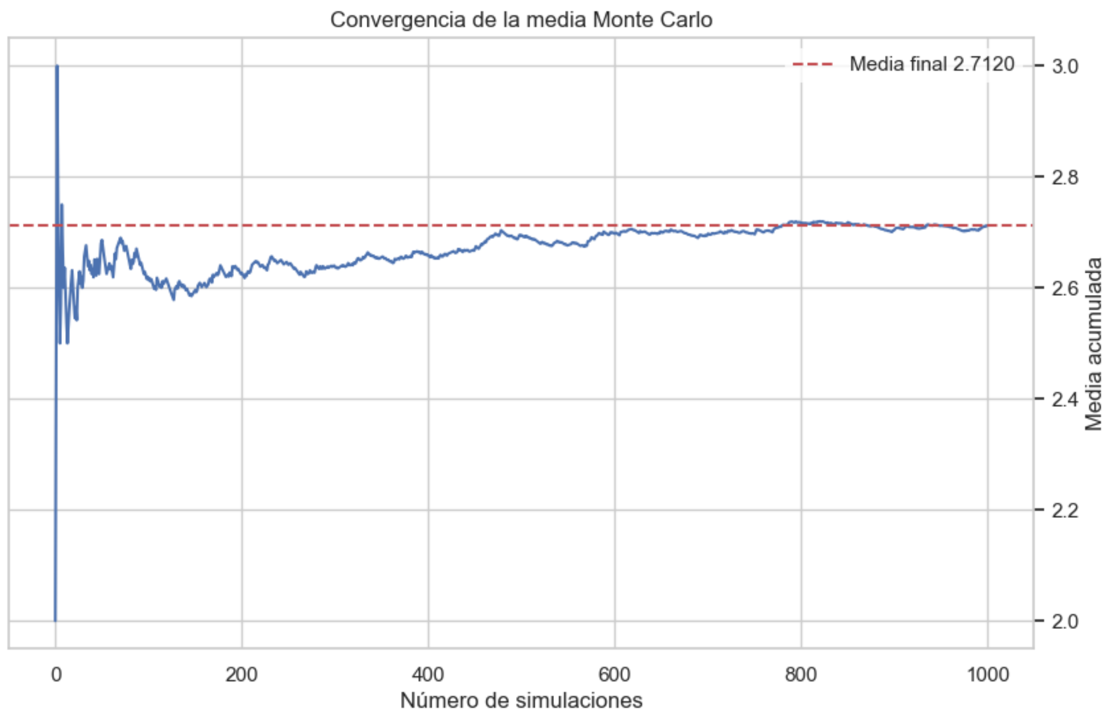
100 valores

[21]:
```
tlc(minimo_N(100))
```



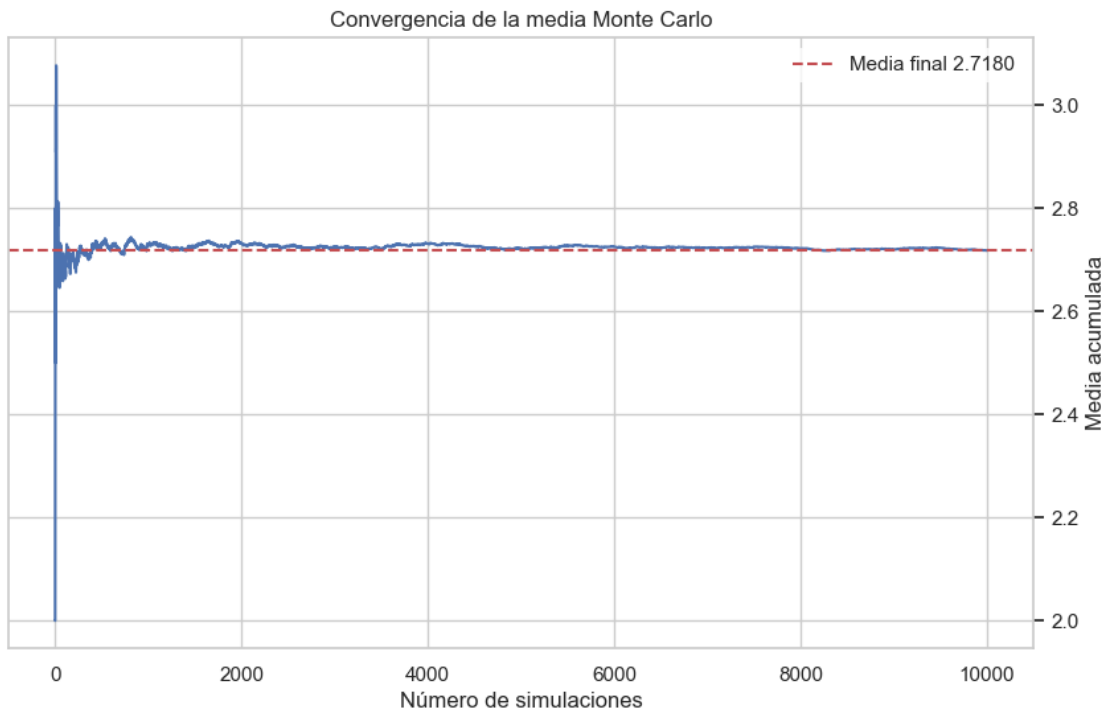
1000 valores

[22]:
```
tlc(minimo_N(1000))
```



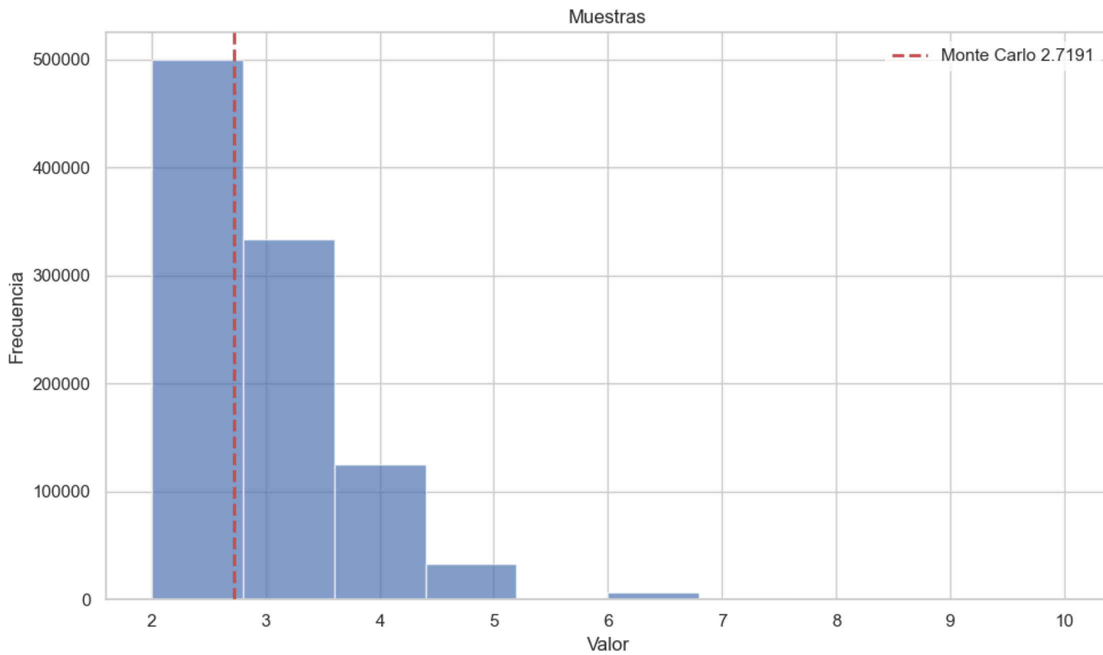
10000 valores

[23]:
```
tlc(minimo_N(10000))
```



Discución de n

[24]:
```
muestras = minimo_N(1000000)
montecarlo = np.mean(muestras)
histograma(muestras, montecarlo, bins=10)
```



[25]:
```
np.e
```

[25]: 2.718281828459045

Ejercicio 2

Si $x_0 = 3$ y $x_n = (5x_{n-1} + 7) \bmod 200$. Encontrar $x_1, \dots, x_{10}$.

$$x_n = ax_{n-1} \bmod m$$

$$ 22 = (5 ·3 + 7) mod 200\

117 = (22 ·3 + 7) mod 200\

192 = (117 ·3 + 7) mod 200\

167 = (192 ·3 + 7) mod 200\

42 = (167 ·3 + 7) mod 200\

17 = (42 ·3 + 7) mod 200\

92 = (17 ·3 + 7) mod 200\

67 = (92 ·3 + 7) mod 200\

142 = (67 ·3 + 7) mod 200\

117 = (142 ·3 + 7) mod 200\
⋮

$$

[26]:
```python
pseudoaleatorios = []

x0 = 3
a = 5
m = 200
c = 7

for i in range(10):
    xn = (a * x0 + c) % m
    x0 = xn
    pseudoaleatorios.append(xn)

pseudoaleatorios
```

[26]: [22, 117, 192, 167, 42, 17, 92, 67, 142, 117]

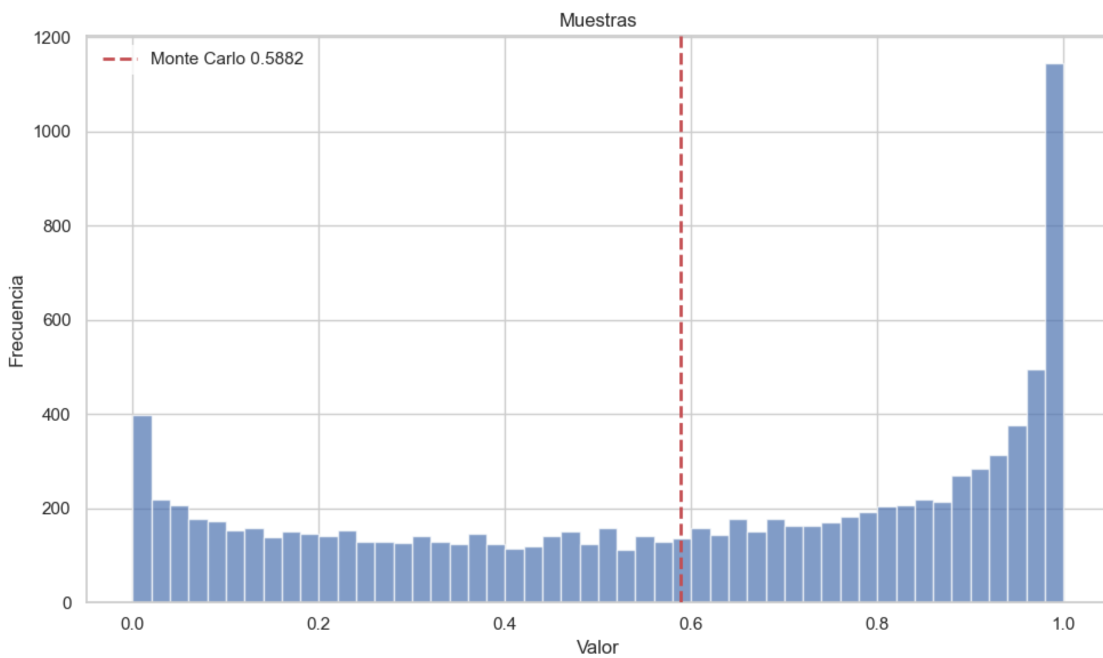
Ejercicio 4

$$\int_0^1 (1-x^2)^{3/2}\,dx.$$

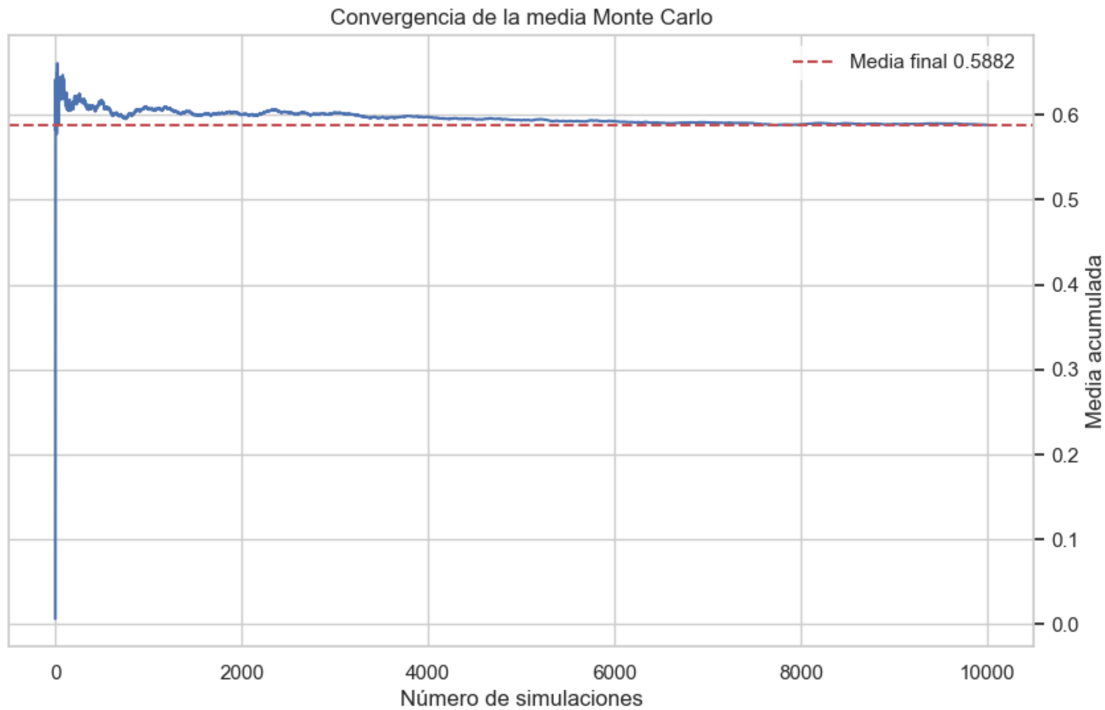
[27]:
```
def h(u):
    return (1-u**2)**(3/2)

k = 10000

u = np.random.random(k)

muestras = h(u)
montecarlo = muestras.mean()
montecarlo
```

[27]: np.float64(0.5882433134915226)

[28]:
```
histograma(muestras, montecarlo)
```



[29]:
```
tlc(muestras)
```


Convergencia de la media Monte Carlo
Media final 0.5882
0.6
0.5
0.4
0.3
0.2
0.1
0.0
Media acumulada
0
2000
4000
6000
8000
10000
Número de simulaciones

Ejercicio 6

$$\int_0^\infty e^{-x}\,dx.$$

Integrando es la densidad $\mathrm{Exp}(1)$, por lo que el valor exacto es 1.

Estimación Monte Carlo

Sea:

$$\theta = \int_0^\infty e^{-x}\, dx.$$

Cambio:

$$y = \frac{1}{x+1}, \qquad dy = -\frac{dx}{(x+1)^2} = -y^2\, dx.$$

Entonces:

$$\theta = \int_0^1 h(y)\, dy, \qquad h(y) = \frac{g\left(\frac{1}{y} - 1\right)}{y^2}, \quad g(x) = e^{-x}.$$

Forma de esperanza con $U \sim \mathrm{Unif}(0, 1)$:

$$\theta = \mathbb{E}[\, h(U)\,].$$

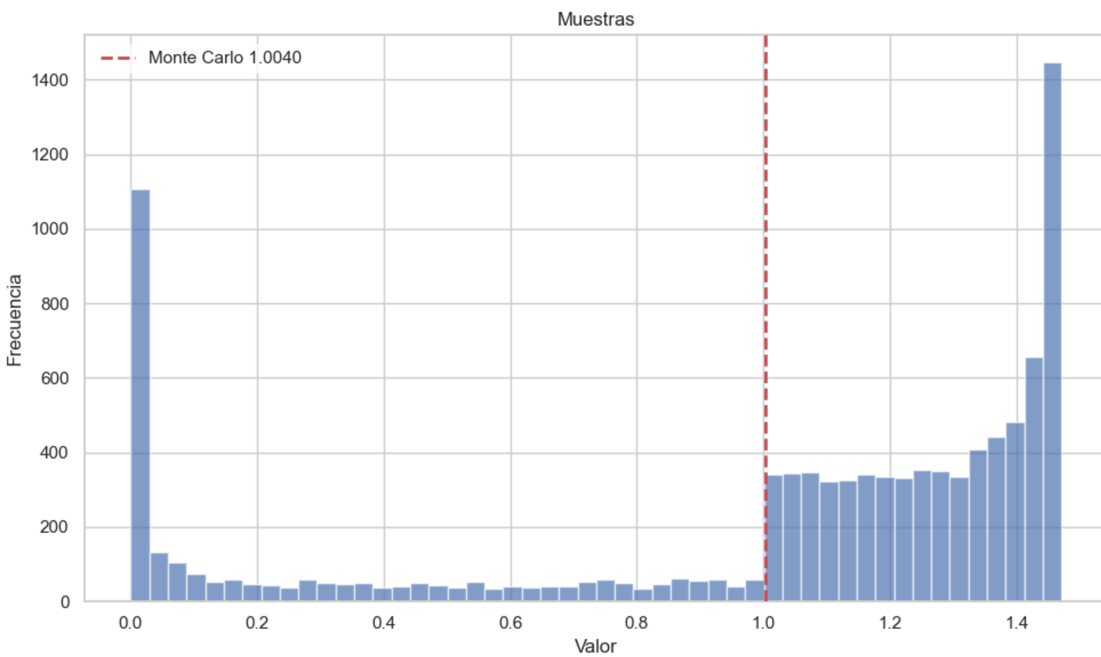
Estimador Monte Carlo:

$$\widehat{\theta}_K = \frac{1}{K}\sum_{i=1}^{K} h(u_i), \quad u_i \overset{\text{iid}}{\sim} \mathrm{Unif}(0, 1).$$

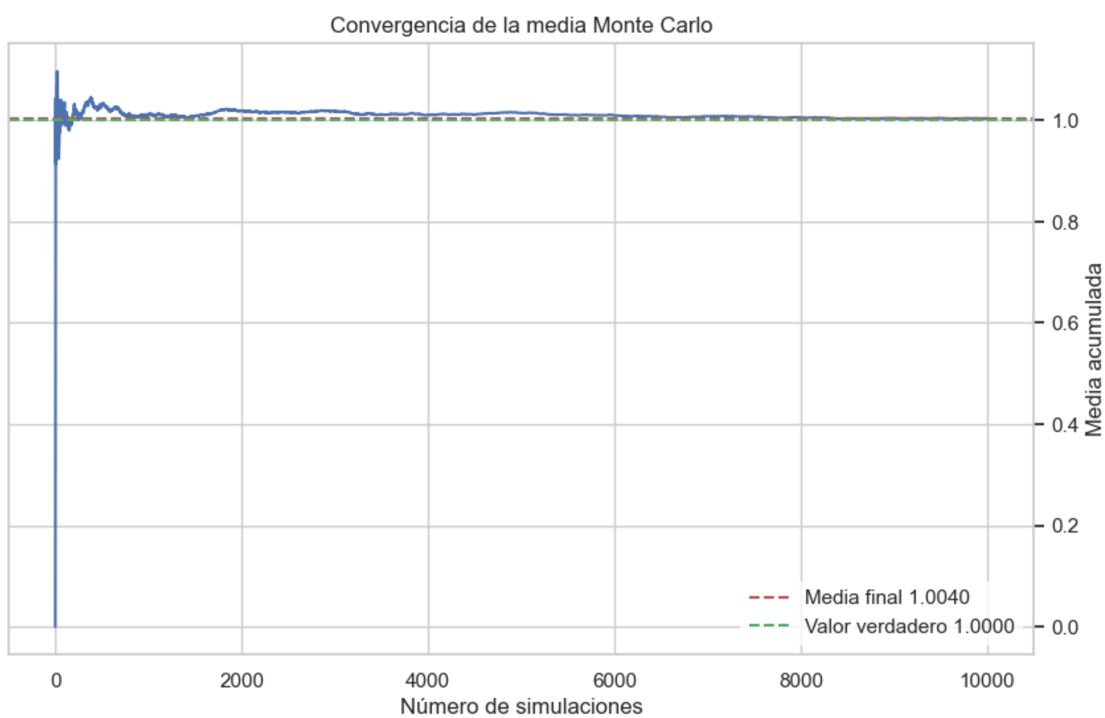
[30]:
```python
def g(x):
    return np.exp(-x)

def h(u):
    return g(1/u-1)/u**2

k = 10_000

u = np.random.random(k)
muestras = h(u)
montecarlo = muestras.mean()
montecarlo
```

[30]:
```
np.float64(1.003981922206577)
```

[31]:
```python
histograma(muestras, montecarlo)
```



[32]:
```
tlc(muestras, 1)
```



Ejercicio 8

$$\int_{-\infty}^{\infty} e^{-x^2}\, dx.$$

Integrando, el valor exacto es $\sqrt{\pi}$ (Lo vimos en análisis II).

Estimación Monte Carlo

Sea:

$$\theta = \int_{-\infty}^{\infty} e^{-x^2}\, dx.$$

Como tiene símetria par, $g(x) = g(-x)$:

$$\theta = 2\int_{0}^{\infty} e^{-x^2}\, dx.$$

Cambio:

$$y = \frac{1}{x+1}, \qquad dy = -\frac{dx}{(x+1)^2} = -y^2\, dx.$$

Entonces:

$$\theta = 2\int_0^1 h(y)\, dy, \qquad h(y) = \frac{g\left(\frac{1}{y}-1\right)}{y^2}, \quad g(x) = e^{-x^2}.$$

Forma de esperanza con $U \sim \mathrm{Unif}(0,1)$:

$$\theta = 2\mathbb{E}[\, h(U)\,].$$

Estimador Monte Carlo:

$$\widehat{\theta}_K = 2 \cdot \frac{1}{K}\sum_{i=1}^{K} h(u_i), \quad u_i \overset{\text{iid}}{\sim} \mathrm{Unif}(0,1).$$

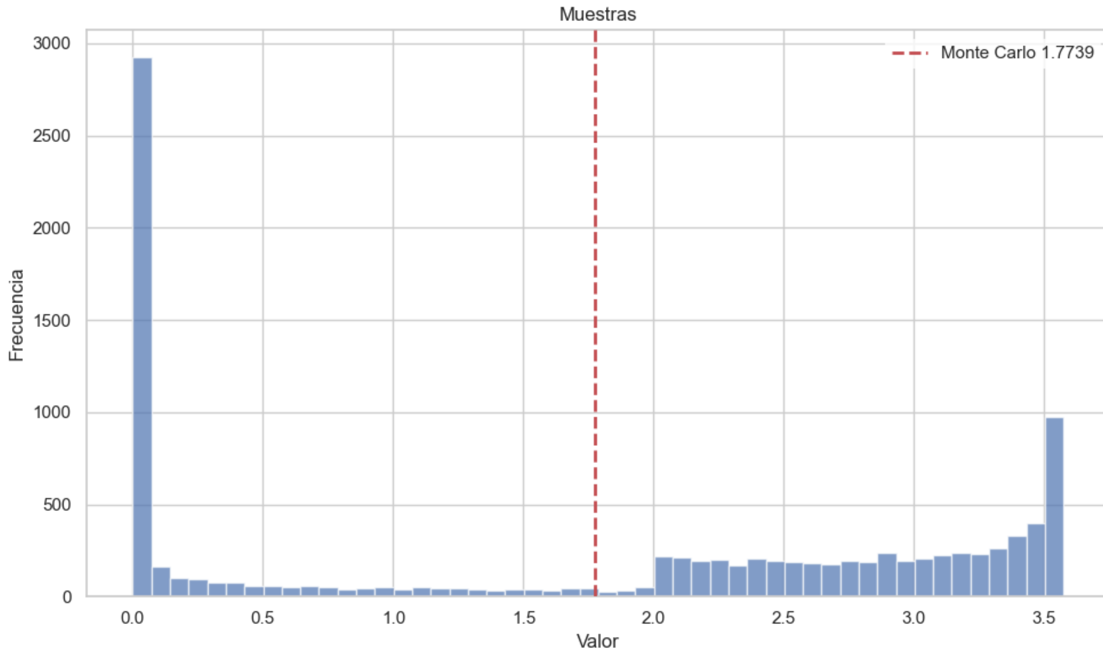
[33]:
```python
def g(x):
    return np.exp(-x**2)

def h(u):
    return 2* g(1/u-1)/u**2

k = 10_000

u = np.random.random(k)
muestras = h(u)
montecarlo = muestras.mean()
montecarlo
```

[33]:
```
np.float64(1.7739489326465634)
```

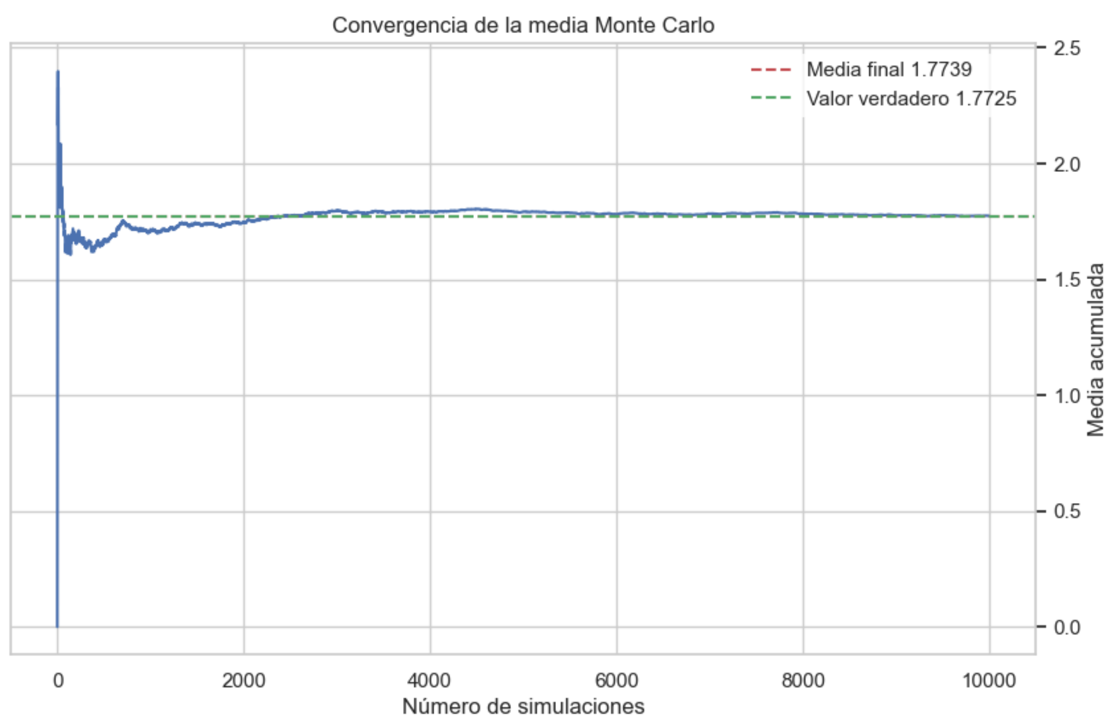
[34]:
```python
verdadero = np.sqrt(np.pi)
verdadero
```

[34]:
```
np.float64(1.7724538509055159)
```

[35]:
```python
histograma(muestras, montecarlo)
```



[36]:
```
tlc(muestras, verdadero)
```



Ejercicio 10

$$\int_0^{\infty} \int_0^{x} e^{-(x+y)}\, dy\, dx.$$

Solución analítica

$$\int_0^{\infty}\int_0^{x} e^{-(x+y)}\, dy\, dx = \int_0^{\infty} e^{-x}\left(\int_0^{x} e^{-y}\, dy\right) dx.$$

Integrando a y con x fijo:

$$\int_0^{x} e^{-y}\, dy = \big[-e^{-y}\big]_0^{x} = 1 - e^{-x}.$$

Sustituyendo:

$$\int_0^{\infty} e^{-x}\big(1 - e^{-x}\big)\, dx = \int_0^{\infty} e^{-x}\, dx - \int_0^{\infty} e^{-2x}\, dx.$$

Evaluando ambas integrales impropias:

$$\int_0^{\infty} e^{-x}\, dx = \big[-e^{-x}\big]_0^{\infty} = 1, \qquad \int_0^{\infty} e^{-2x}\, dx = \big[-\tfrac{1}{2}e^{-2x}\big]_0^{\infty} = \tfrac{1}{2}.$$

Restando:

$$1 - \tfrac{1}{2} = \tfrac{1}{2}.$$

Estimación Monte Carlo

Sea:

$$\theta = \int_0^\infty \int_0^x e^{-(x+y)}\, dy\, dx.$$

Entonces:

$$\theta = \int_0^\infty \int_0^\infty \mathbf{1}\{x_2 \le x_1\}\, e^{-(x_1+x_2)}\, dx_2\, dx_1, \qquad g(x_1,x_2) = \mathbf{1}\{x_2 \le x_1\}.$$

Sabemos que:

$$\theta = \mathbb{E}\big[g(X_1,X_2)\big], \quad X_1, X_2 \overset{iid}{\sim} \mathrm{Exp}(1).$$

Estimador Monte Carlo:

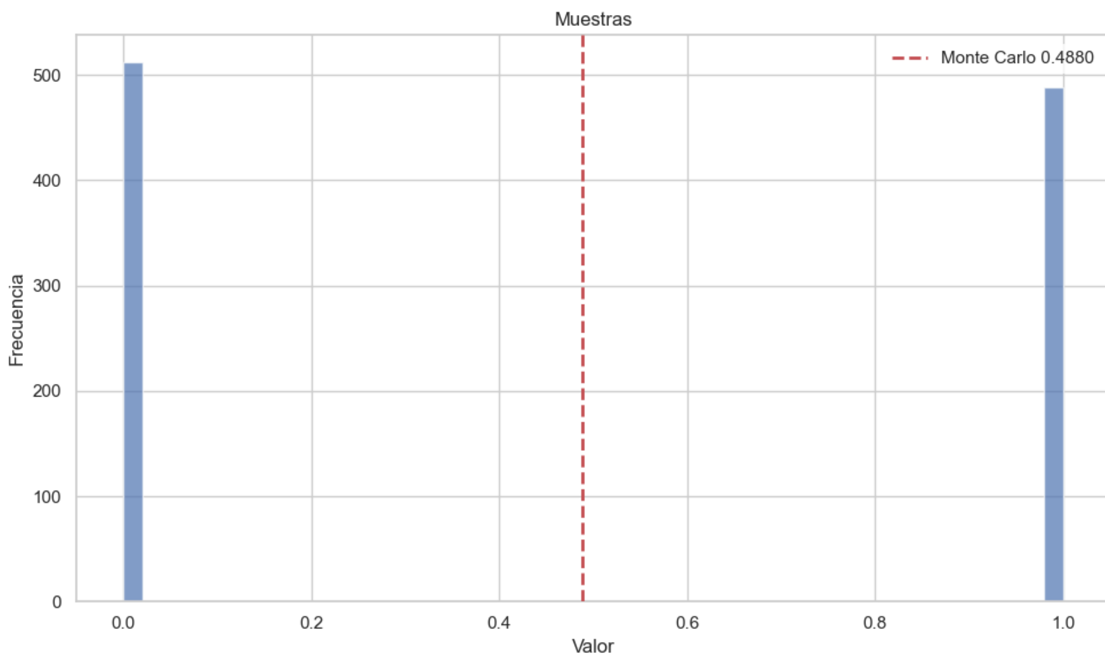
$$\widehat{\theta}_k = \frac{1}{k}\sum_{i=1}^{k} g(x_{i1},x_{i2}) = \frac{1}{k}\sum_{i=1}^{k} \mathbf{1}\{x_{i2} \le x_{i1}\}, \quad (x_{i1},x_{i2}) \overset{iid}{\sim} \mathrm{Exp}(1).$$

(Para simular x_{ij}: $x_{ij} = -\ln(1-u_{ij})$, $u_{ij} \overset{iid}{\sim} \mathrm{Unif}(0,1)$, con el método de la transformada inversa).

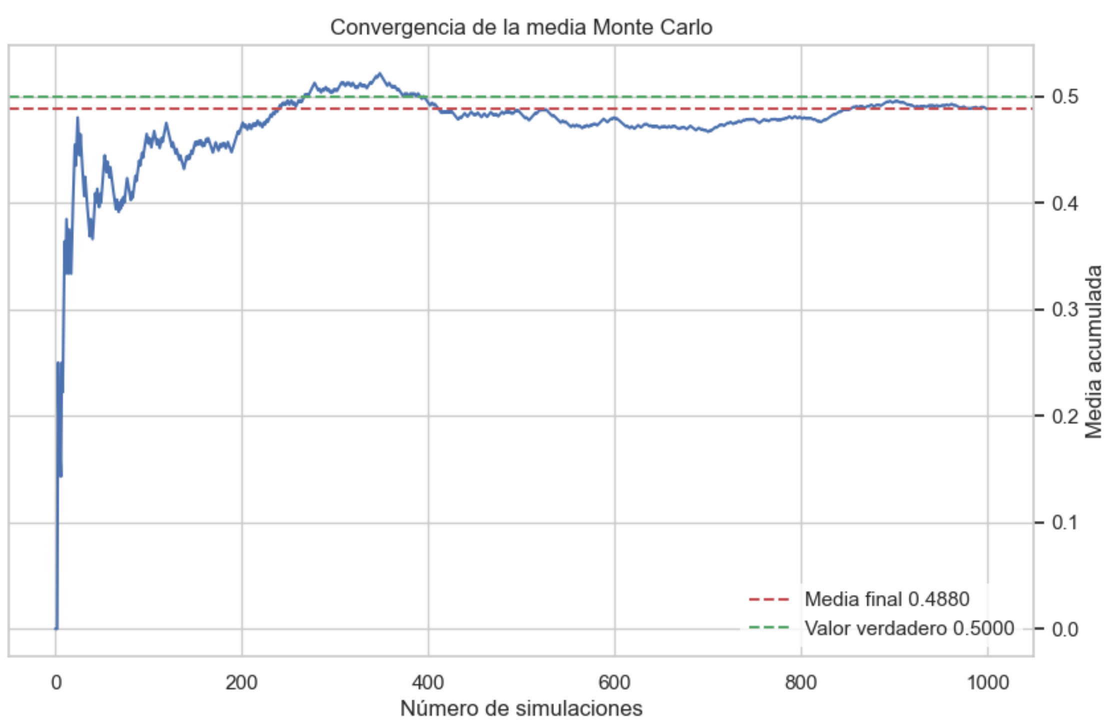
[37]:
```
k = 1000
u1 = np.random.random(k)
u2 = np.random.random(k)

e1 = -np.log(u1)
e2 = -np.log(u2)

def g(e1, e2):
    if e1 < e2:
        return 1
    else:
        return 0

muestras = [g(e1[i], e2[i]) for i in range(k)]

montecarlo = np.mean(muestras)
montecarlo
```

[37]:
```
np.float64(0.488)
```

[38]:
```
histograma(muestras, montecarlo)
```



[39]:
```
tlc(muestras, 1/2)
```



Ejercicio 12

Sea $U \sim \mathcal{U}(0,1)$. Aproximar por simulación:

(a) $\operatorname{Corr}\left(U, \sqrt{1-U^2}\right)$,

(b) $\operatorname{Corr}\left(U^2, \sqrt{1-U^2}\right)$.

Covarianza

$$\operatorname{Cov}(X,Y) = \mathbb{E}\big[(X-\mathbb{E}X)(Y-\mathbb{E}Y)\big].$$

Expansión lineal:

$$\operatorname{Cov}(X,Y) = \mathbb{E}[XY] - \mathbb{E}[X]\,\mathbb{E}[Y].$$

Aplicando a $X = U$ y $Y = e^U$:

$$\operatorname{Cov}(U, \sqrt{1-U^2}) = \mathbb{E}\big[U\sqrt{1-U^2}\big] - \mathbb{E}[U]\,\mathbb{E}[\sqrt{1-U^2}].$$

Correlación

$$\widehat{\rho} = \frac{\widehat{\operatorname{Cov}}}{\sqrt{s_U^2\, s_Y^2}}.$$

Varianza

$$\operatorname{Var}(X) = \mathbb{E}[X^2] - (\mathbb{E}[X])^2.$$

Estimación Monte Carlo

Sea $u_1, \ldots, u_K \overset{iid}{\sim} \mathrm{Unif}(0,1)$. Entonces

$$\widehat{\mu}_U = \frac{1}{K}\sum_{i=1}^{K} u_i, \qquad \widehat{\mu}_{\sqrt{1-U^2}} = \frac{1}{K}\sum_{i=1}^{K} \sqrt{1-U^2}, \qquad \widehat{m} = \frac{1}{K}\sum_{i=1}^{K} u_i\sqrt{1-U^2}.$$

Entonces:

$$\widehat{\mathrm{Cov}}^{(MC)} = \widehat{m} - \widehat{\mu}_U\,\widehat{\mu}_e$$

A

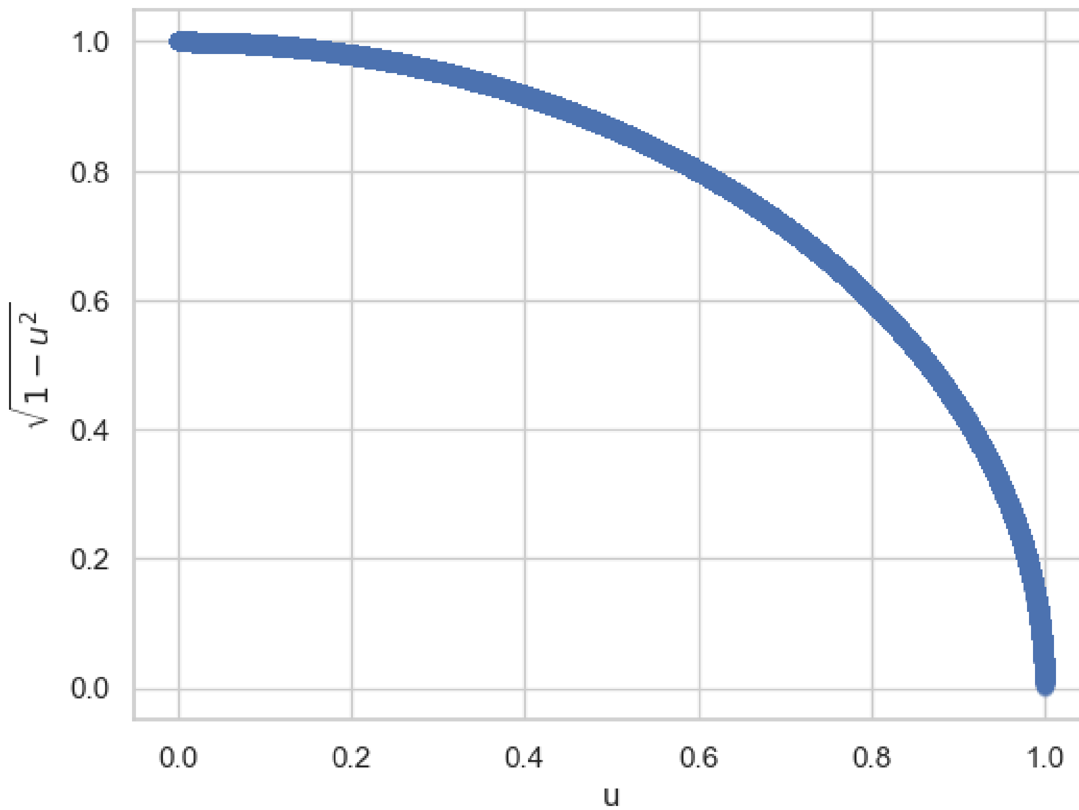
```
[40]: def valor_esperado_1(u):
          return u * np.sqrt(1-u**2)

      def valor_esperado_2(u):
          return u

      def valor_esperado_3(u):
          return np.sqrt(1-u**2)

      k = 1000000

      u = np.random.random(k)

      cov = valor_esperado_1(u).mean() - valor_esperado_2(u).mean() *␣
       ↪valor_esperado_3(u).mean()
      cov
```

```
[40]: np.float64(-0.0593370812703804)
```

```
[41]: def varianza(u):
          return (u**2).mean() - u.mean()**2

      corr = cov / np.sqrt(varianza(u) * varianza(np.sqrt(1-u**2)))
      corr
```

```
[41]: np.float64(-0.9213938620284924)
```

```
[42]: plt.scatter(u, np.sqrt(1-u**2), alpha=0.1)
      plt.xlabel("u")
      plt.ylabel("$\\sqrt{1-u^2}$")
      plt.show()
```



1.0
0.8
0.6
0.4
0.2
0.0
$\sqrt{1-u^2}$
0.0
0.2
0.4
0.6
0.8
1.0
u

B

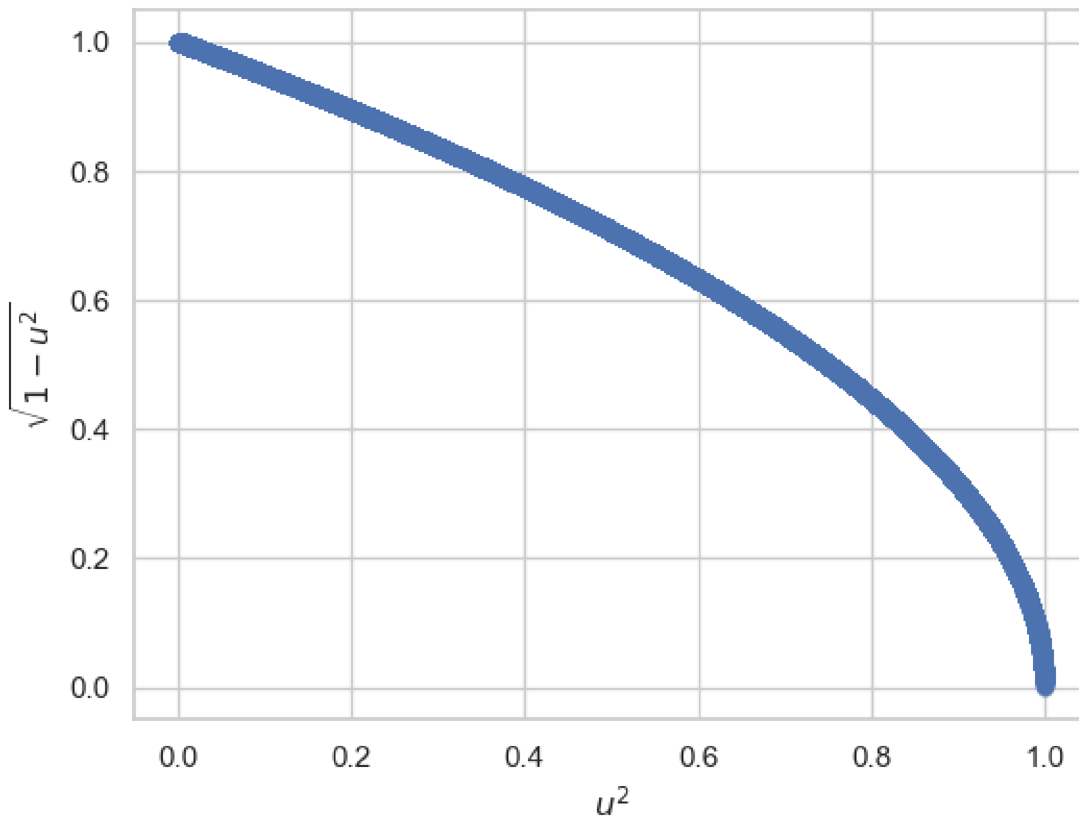
[43]:
```
def valor_esperado_1(u):
    return u**2 * np.sqrt(1-u**2)

def valor_esperado_2(u):
    return u**2

def valor_esperado_3(u):
    return np.sqrt(1-u**2)

k = 1000000

u = np.random.random(k)

cov = valor_esperado_1(u).mean() - valor_esperado_2(u).mean() *␣
↪valor_esperado_3(u).mean()
cov
```

[43]: np.float64(-0.06548312826910846)

[44]:
```
def varianza(u):
    return (u**2).mean() - u.mean()**2

corr = cov / np.sqrt(varianza(u**2) * varianza(np.sqrt(1-u**2)))
corr
```

[44]: np.float64(-0.9835397769788468)

[45]:
```
plt.scatter(u**2, np.sqrt(1-u**2), alpha=0.1)
plt.xlabel("$u^2$")
plt.ylabel("$\\sqrt{1-u^2}$")
plt.show()
```



1.0
0.8
0.6
0.4
0.2
0.0
$\sqrt{1-u^2}$
0.0
0.2
0.4
0.6
0.8
1.0
u^2

Ejercicio 14

Sea $U_i \sim \mathcal{U}(0,1)$ i.i.d. Definir

$$N = \max\left\{n : \prod_{i=1}^{n} U_i \geq e^{-3}\right\}, \quad \text{con } \prod_{i=0}^{0} U_i = 1.$$

a) Estimar $\mathbb{E}[N]$ por simulación.
b) Estimar $\mathbb{P}[N = i]$ para $i = 0, 1, 2, 3, 4, 5, 6$.

A

```
ENTRADA: k
SALIDA: E_hat

acumulado ← 0
PARA r ← 1 HASTA k HACER:
    S ← 0
    n ← 0
    MIENTRAS S  3 HACER:
        u ← UNIFORME(0,1)
        S ← S + (log u)
        SI S  3 ENTONCES:
            n ← n + 1
    acumulado ← acumulado + n
E_hat ← acumulado / k
RETORNAR E_hat
```

[46]:
```python
def estimar_E_N(k, seed=None):
    rng = np.random.default_rng(seed)
    total = 0
    for _ in range(k):
        S = 0.0
        n = 0
        while S <= 3.0:
            S += -np.log(rng.random())
            if S <= 3.0:
                n += 1
        total += n
    return total / k
```

[47]:
```python
estimar_E_N(10000)
```

[47]: 3.0137

B

```
ENTRADA: k
SALIDA: p_hat[0..6]

p_hat[0..6] ← 0
PARA r ← 1 HASTA k HACER:
    S ← 0
    n ← 0
    MIENTRAS S  3 HACER:
        u ← UNIFORME(0,1)
        S ← S + (log u)
        SI S  3 ENTONCES:
            n ← n + 1
    SI 0  n  6 ENTONCES:
        p_hat[n] ← p_hat[n] + 1
PARA i ← 0 HASTA 6 HACER:
    p_hat[i] ← p_hat[i] / k
RETORNAR p_hat
```

[48]:
```python
def estimar_pmf_N(k, seed=None):
    rng = np.random.default_rng(seed)
    counts = np.zeros(7, dtype=int)
    for _ in range(k):
        S = 0.0
        n = 0
        while S <= 3.0:
            S += -np.log(rng.random())
            if S <= 3.0:
                n += 1
        if 0 <= n <= 6:
            counts[n] += 1
    return counts / k
```

[49]:
```python
for i in range(7):
    print(f"Estimación de P(N={i}): {estimar_pmf_N(10000)[i]:.4f}")
```

```
Estimación de P(N=0): 0.0479
Estimación de P(N=1): 0.1475
Estimación de P(N=2): 0.2253
Estimación de P(N=3): 0.2321
Estimación de P(N=4): 0.1740
Estimación de P(N=5): 0.0974
Estimación de P(N=6): 0.0493
```