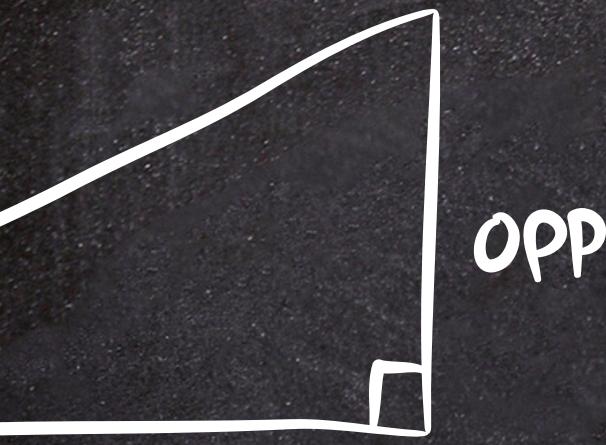


$$a + b = b + a$$

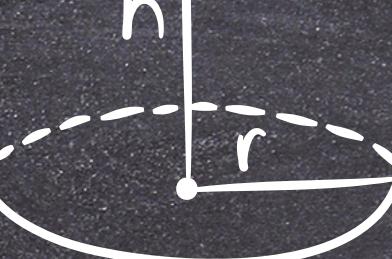
$$\frac{b \pm \sqrt{b^2 - 4ac}}{2a}$$



adj

$$\theta) = \frac{\text{OPP}}{\text{adj}}$$

$$ax^2 + bx + c = 0$$



$$V = \pi r^2 h$$

$$A = bh$$

# PROBABILIDAD

Integrantes:

Arath Yahir Montero Hernandez

Jesus Arturo resendiz Rosales

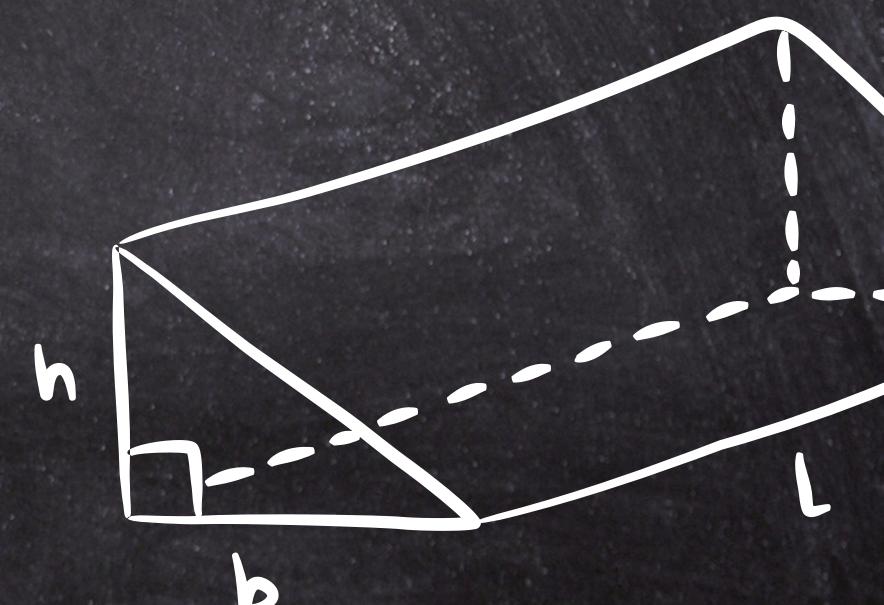
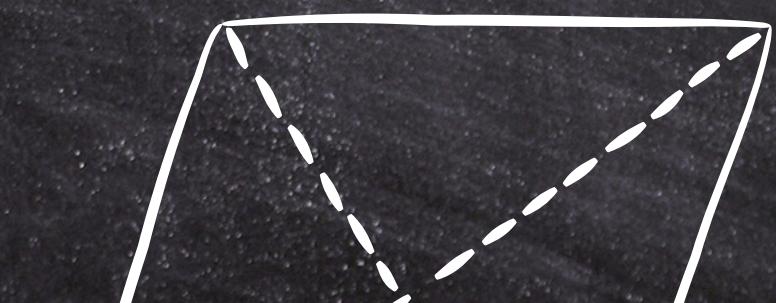
Pedro Ivan Landaver Mora

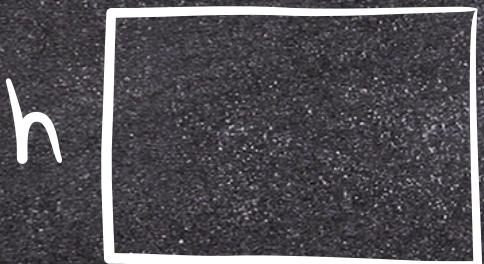
Santiago González Chávez

Victoria del Carmen Hernández Montes



$$V = \frac{4}{3} \pi r^3$$





$$A = bh$$

$$ax^2 + bx + c = 0$$

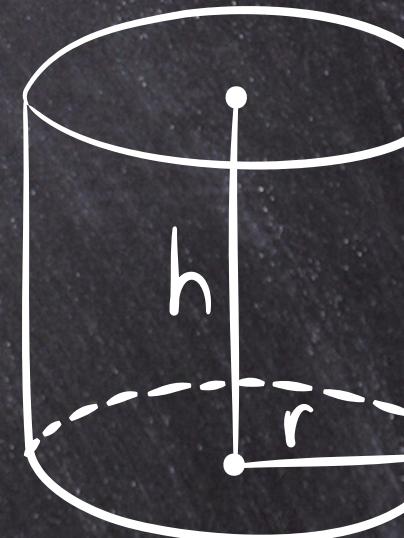
$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

# GENERADORES DE NÚMEROS PSEUDOALEATORIOS.

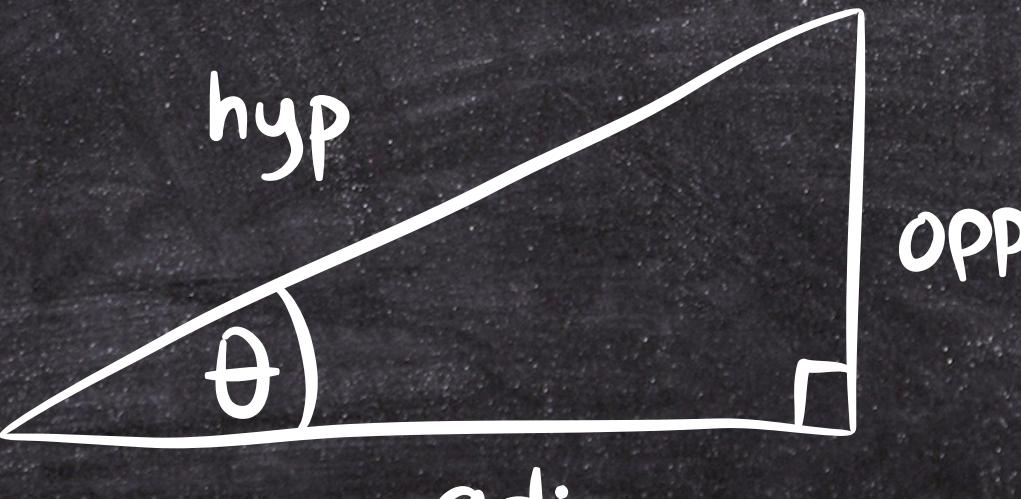


$$\pi r^3$$

$$a + b = b + a$$



$$V = \pi r^2 h$$



# FUNDAMENTOS DE LOS GENERADORES DE NÚMEROS SEUDOALEATORIOS

La estructura típica de un generador de números aleatorios es la siguiente: existe un conjunto finito  $S$  de estados, y una función  $f : S \rightarrow S$ .

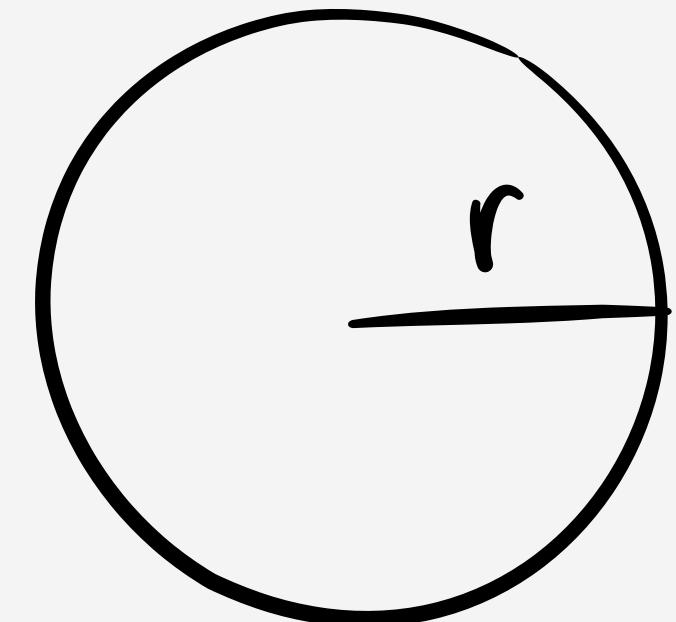
Existe un espacio de salida  $U$ , y una función de salida  $g : S \rightarrow U$ .  
Siempre tomaremos el espacio de salida como el intervalo  $(0, 1)$

El generador recibe un valor inicial  $S_0$  para el estado, llamado semilla (seed). La semilla suele ser proporcionada por el usuario. Luego, se genera una secuencia de números aleatorios definiendo:

$$\begin{aligned} S_n &= f(S_{n-1}), \quad n = 1, 2, 3, \dots \\ U_n &= g(S_n) \end{aligned} \quad (3.1)$$

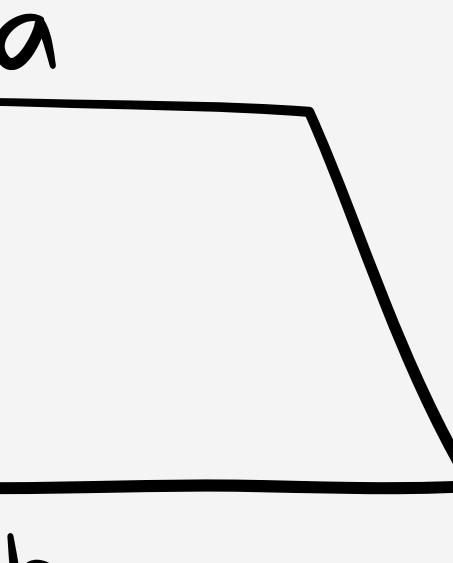
Si generamos una secuencia de números con este procedimiento y luego generamos otra secuencia usando la misma semilla, las dos secuencias serán idénticas.

Como el espacio de estados es finito,  $S_n$  eventualmente debe volver a un estado anterior. El entero positivo más pequeño  $p$  tal que, para algún estado, la función  $f$  regrese a ese estado después de  $p$  iteraciones se llama período del generador.



$$C = 2\pi r$$

Obviamente, los períodos largos son mejores que los cortos. Pero un período largo por sí solo no garantiza que el generador sea bueno.



$$a + (b + c) = (a + b) + c$$

# PASAR PRUEBAS ESTADÍSTICAS EMPÍRICAS

Estas pruebas consisten en generar una secuencia larga de números aleatorios y luego realizar diversas pruebas estadísticas para verificar la hipótesis de que los números están distribuidos uniformemente en  $[0, 1]$  y son independientes.

# EJEMPLOS

01

## Base matemática

Existe una teoría matemática considerable detrás de estos generadores (al menos de algunos de ellos), incluyendo propiedades que deberían garantizar buenos resultados.

02

## Rápido (y no mucha memoria)

La mayoría de las simulaciones de Monte Carlo requieren una gran cantidad de números aleatorios. A menudo se desean generar muchos ejemplos, y cada uno puede requerir múltiples llamadas al generador. Por lo tanto, debe ser rápido.

03

## Múltiples flujos (streams)

Es fácil usar computación paralela para Monte Carlo. Pero esto requiere ejecutar múltiples copias del generador de números aleatorios, por lo que estos diferentes flujos deben ser independientes entre sí.

04

## Aspectos prácticos

Facilidad de instalación y de asignar la semilla. Algunos generadores son muy simples y sólo requieren unas pocas líneas de código. Otros son bastante más complejos. Algunos, como el Mersenne Twister, requieren una semilla grande.

$$a + (b + c) = (a + b) + c$$

05

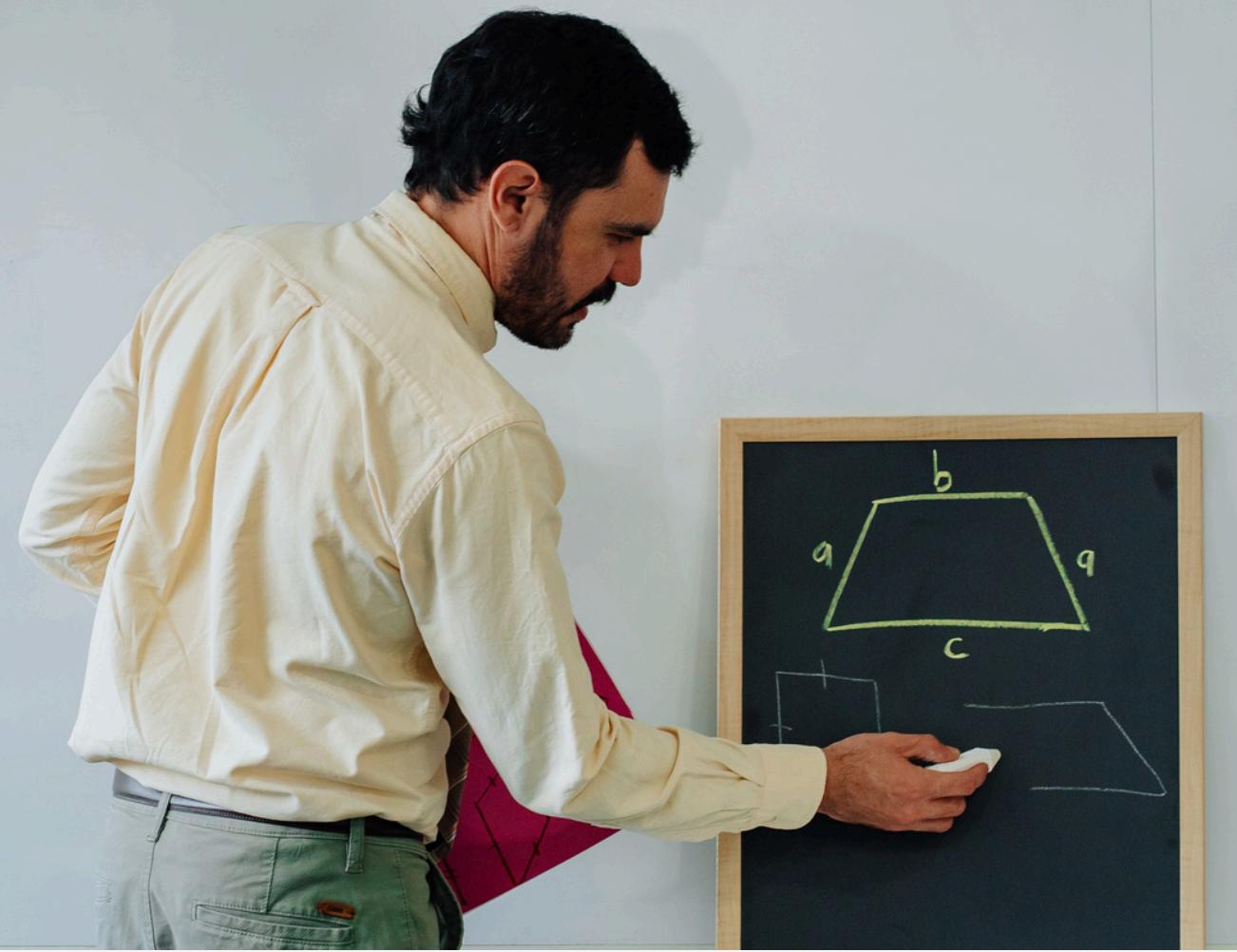
## Generadores congruenciales lineales

El espacio de estados es  $\{0, 1, 2, \dots, m - 1\}$ , donde  $m$  es un entero positivo.

$$f(X) = aX + c \bmod m \quad (3.2)$$

donde  $\bmod m$  indica que se hace aritmética módulo  $m$ . Las constantes  $a$  y  $c$  son enteros, y no hay pérdida de generalidad si se toman en  $\{0, \dots, m-1\}$ . Para la función de salida podemos tomar:

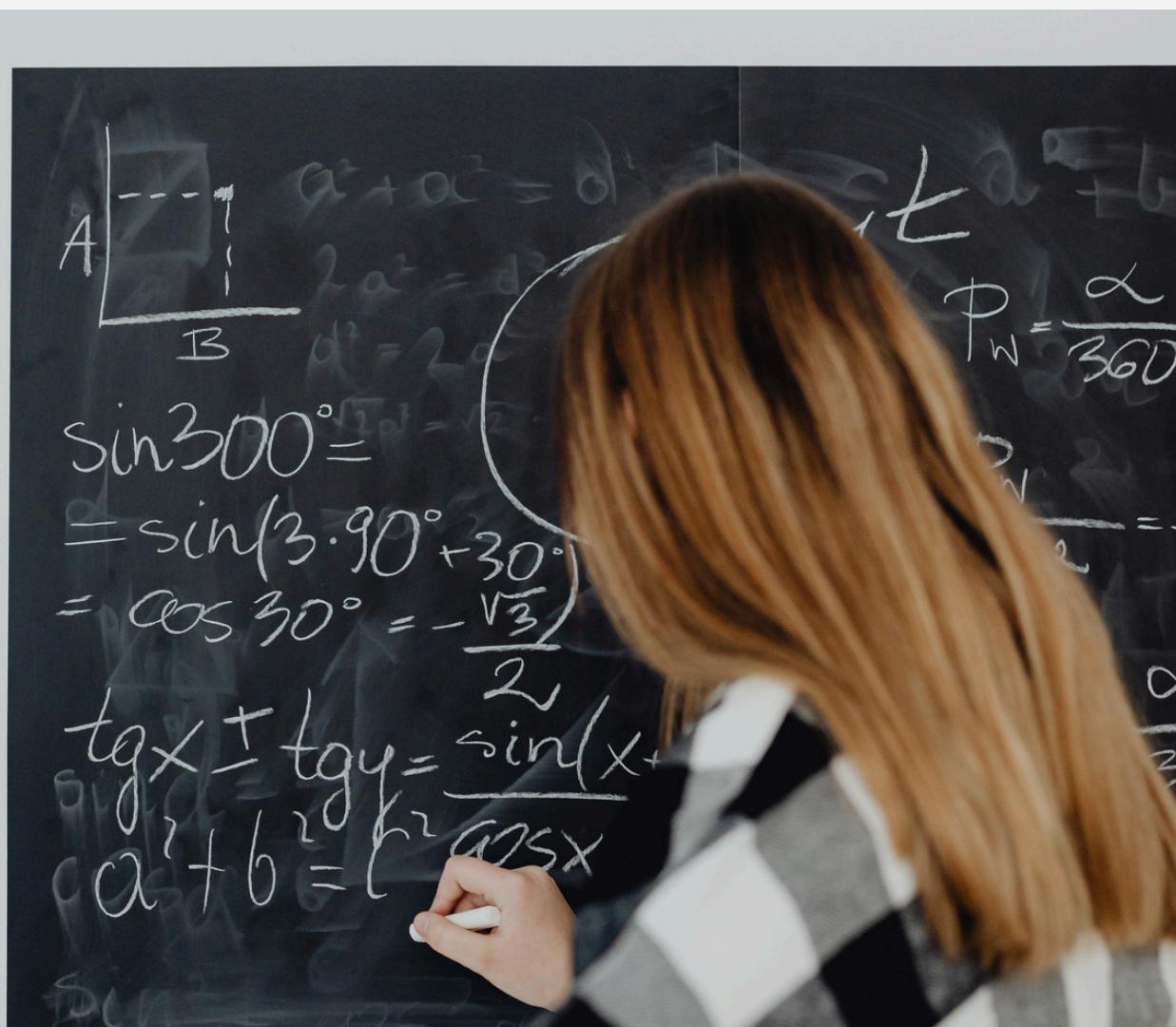
$$g(X) = X / m \quad (3.3)$$



La calidad de este generador depende de cómo se elijan las constantes  $a$  y  $c$ . Hay bastante teoría matemática detrás de estas elecciones, que no trataremos aquí.

Ejemplo: Lewis, Goodman y Miller propusieron

$a = 7^5 = 16807$ ,  $c = 0$  y  $m = 2^{31} - 1 = 2147483647$ . Tiene período  $2^{31} - 2$ .



drand48(): es un generador congruencial lineal con  $m = 2^{48}$ ,  $a = 0x5DEECE66D = 2736731631558$ ,  $c = 0xB = 138$ . Es parte de la biblioteca estándar de C. No se recomienda.

# MÉTODOS DE GENERACIÓN DE NÚMEROS ALEATORIOS RECTANGULARES

Las variables generadas deben cumplir con:

- Ser uniformemente distribuidas
- Estadísticamente independientes
- Reproducibles
- Tener periodo largo (sin una pronta repetición)
- Generadas por un método rápido
- Generadas por un método que no requiera mucha capacidad de almacenamiento en memoria

La prueba de Kolmogorov-Smirnov (K-S) es una prueba estadística no paramétrica utilizada para determinar si una muestra de datos sigue una distribución teórica específica, como la normal. Se basa en comparar la distribución acumulada empírica de los datos con la distribución acumulada teórica esperada.



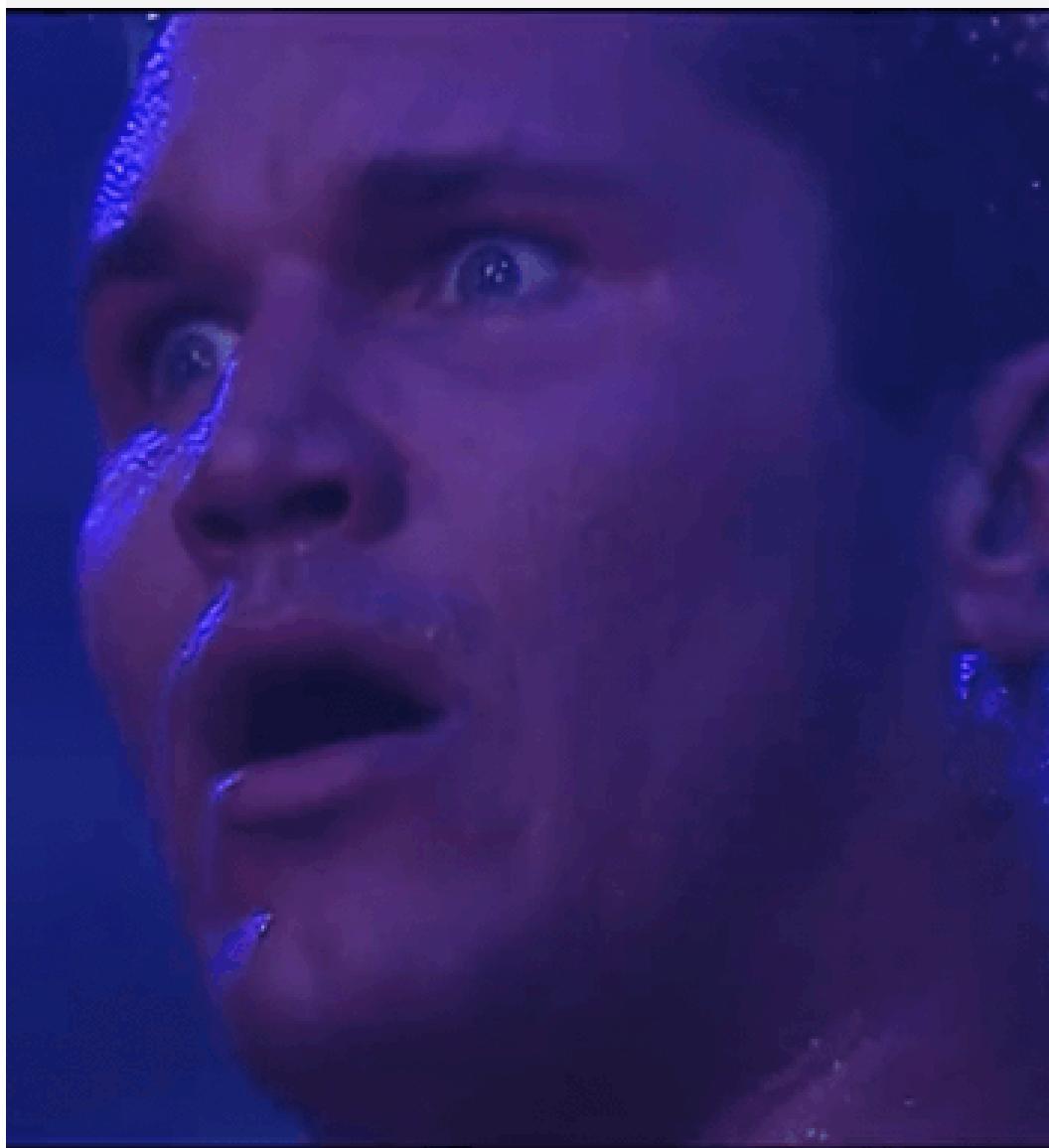
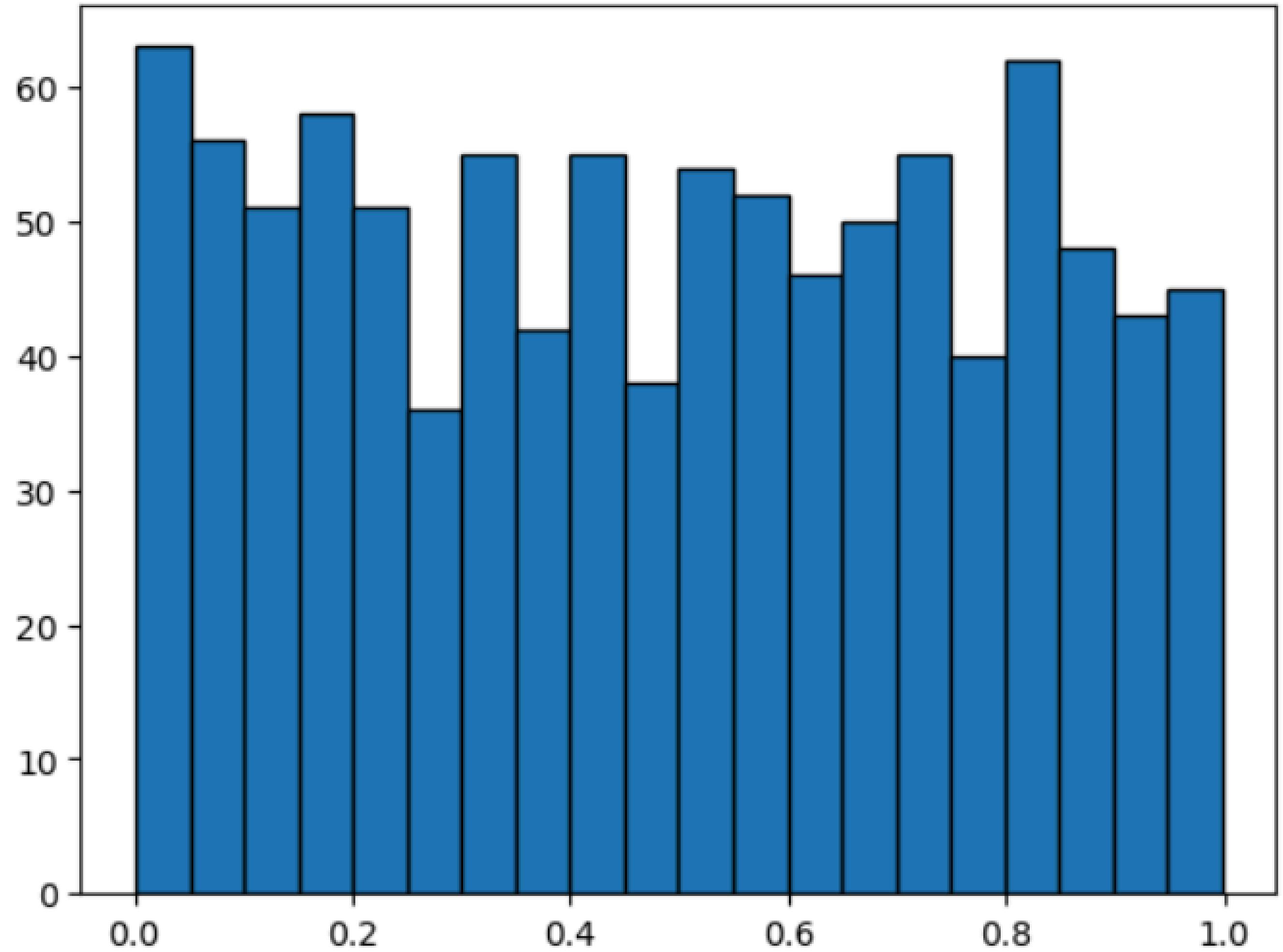
### Distribución Acumulada Empírica

Es la función que muestra la proporción de datos observados menores o iguales a un valor dado, basada en una muestra real.

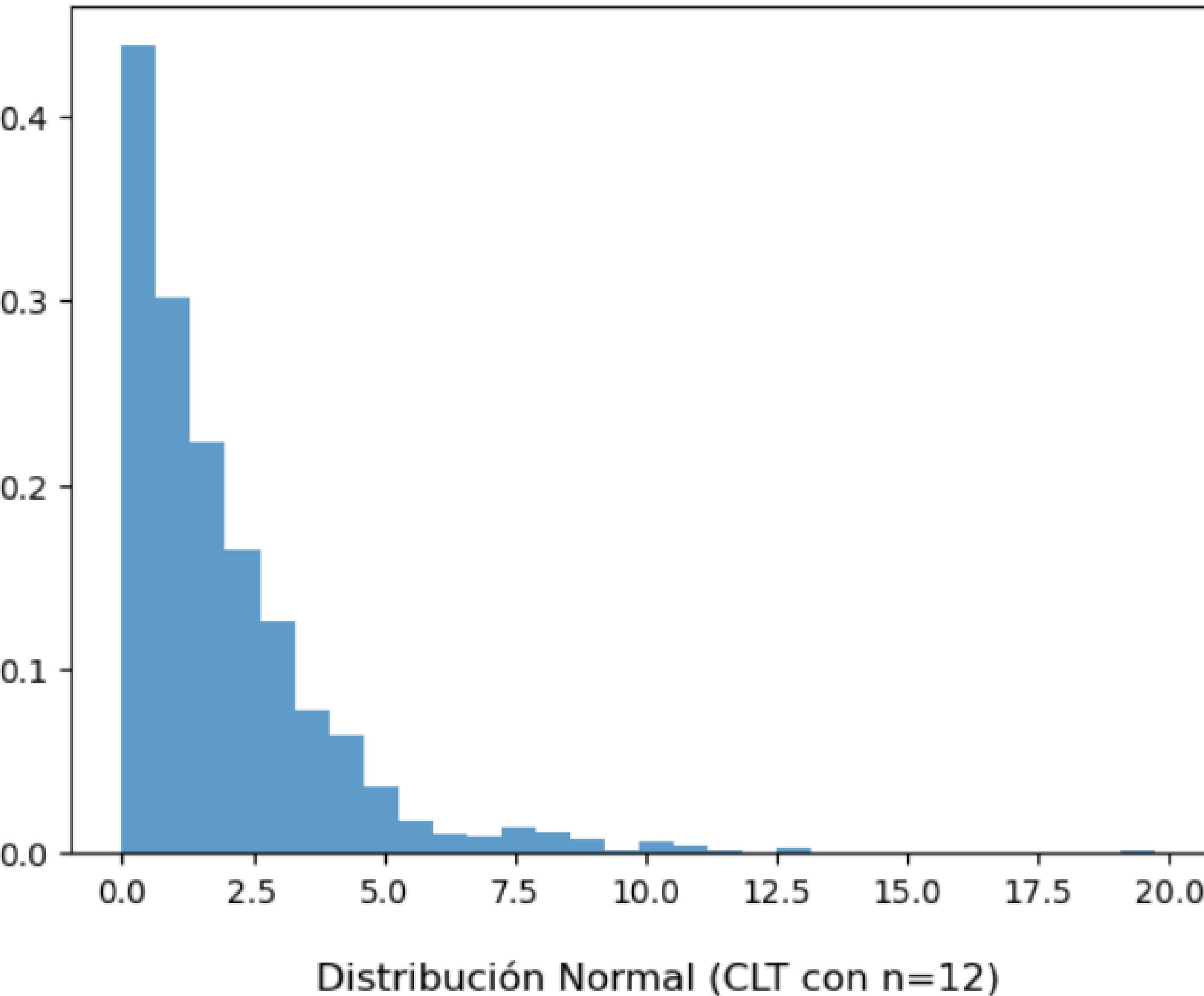
### Distribucion Acumulada Teorica Esperada

Es la función matemática que describe cómo deberían distribuirse los valores en una población ideal según un modelo probabilístico.

# Método Congruencial Multiplicativo



## Distribución Exponencial ( $\theta=2$ )



Distribución Normal (CLT con n=12)

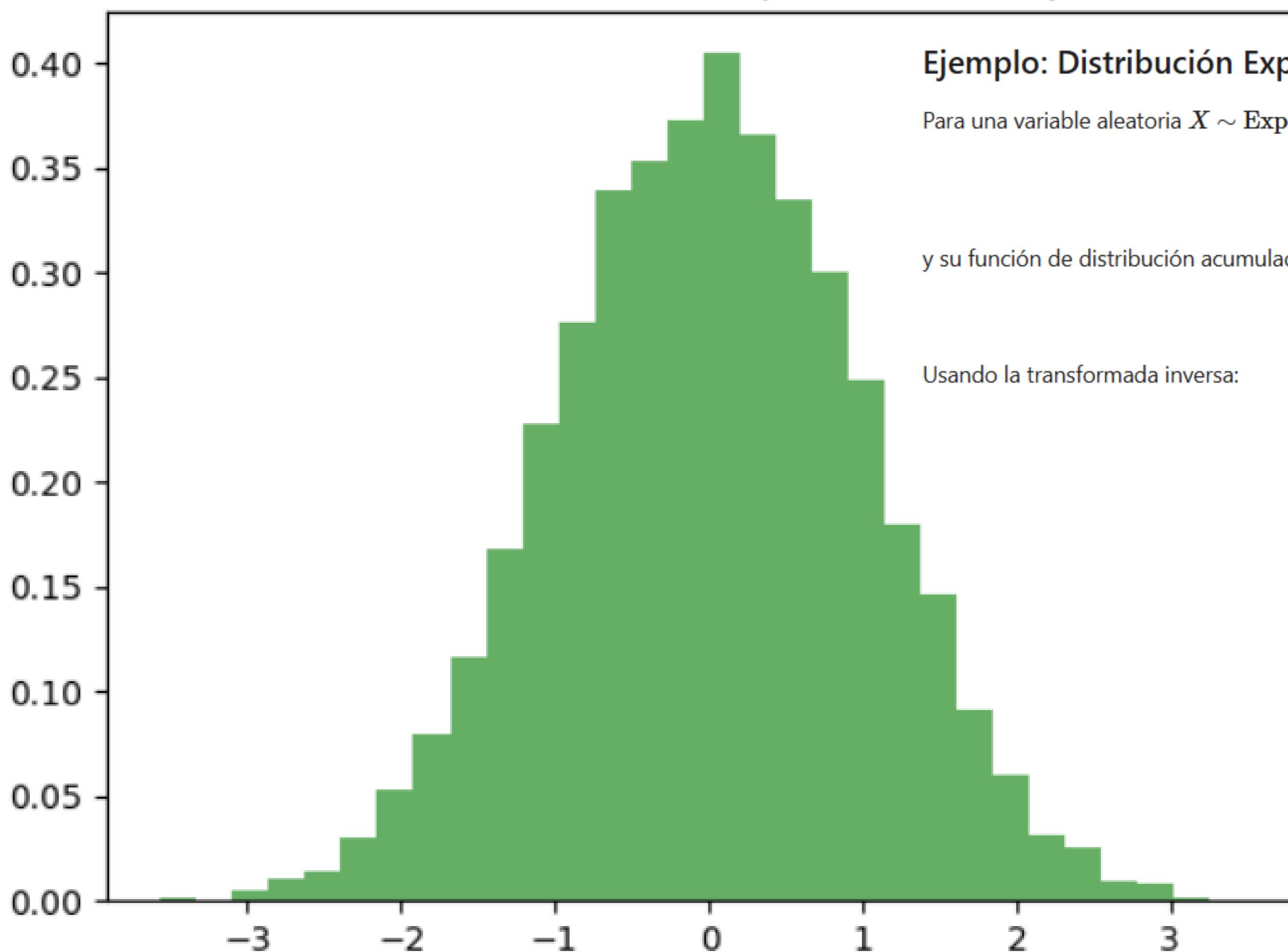
$$f(x) = \frac{1}{\theta} e^{-x/\theta}, \quad x \geq 0$$

$$F(x) = 1 - e^{-x/\theta}$$

$$x = -\theta \ln(R)$$



# Distribución Normal (CLT con n=12)



## Ejemplo: Distribución Exponencial

Para una variable aleatoria  $X \sim \text{Exp}(\lambda)$  con  $\lambda = 1/\theta$ , la función de densidad es:

$$f(x) = \frac{1}{\theta} e^{-x/\theta}, \quad x \geq 0$$

y su función de distribución acumulada es:

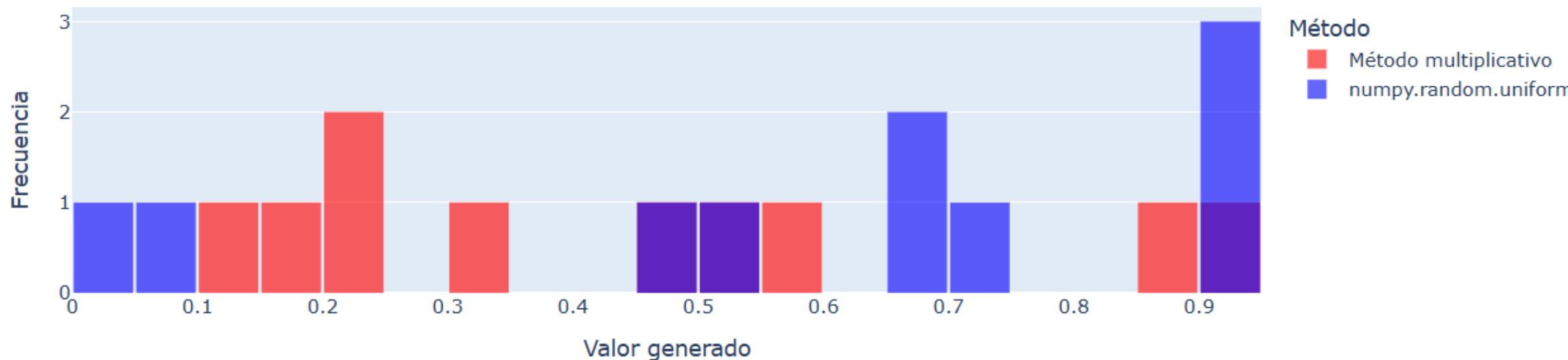
$$F(x) = 1 - e^{-x/\theta}$$

Usando la transformada inversa:

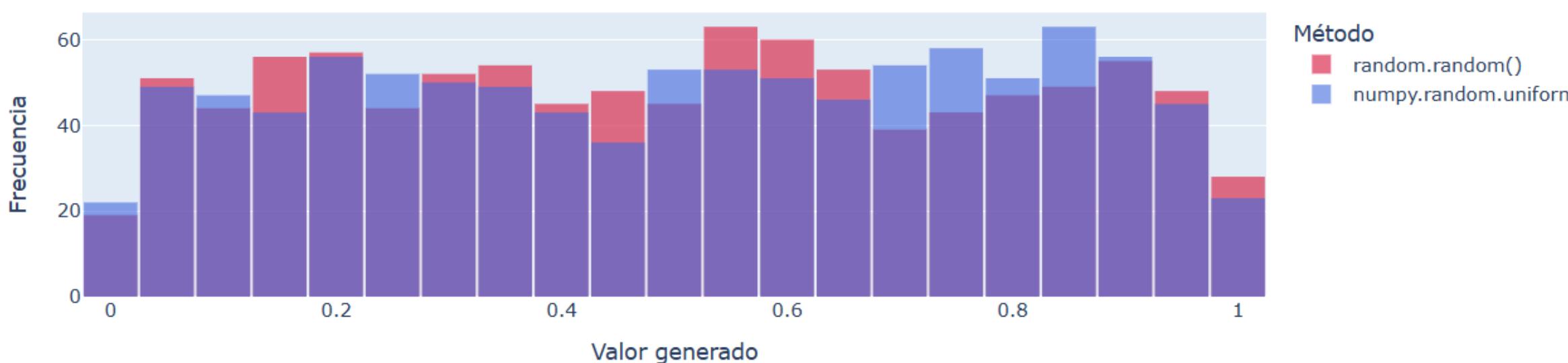
$$x = -\theta \ln(U), \quad U \sim \mathcal{U}(0, 1)$$



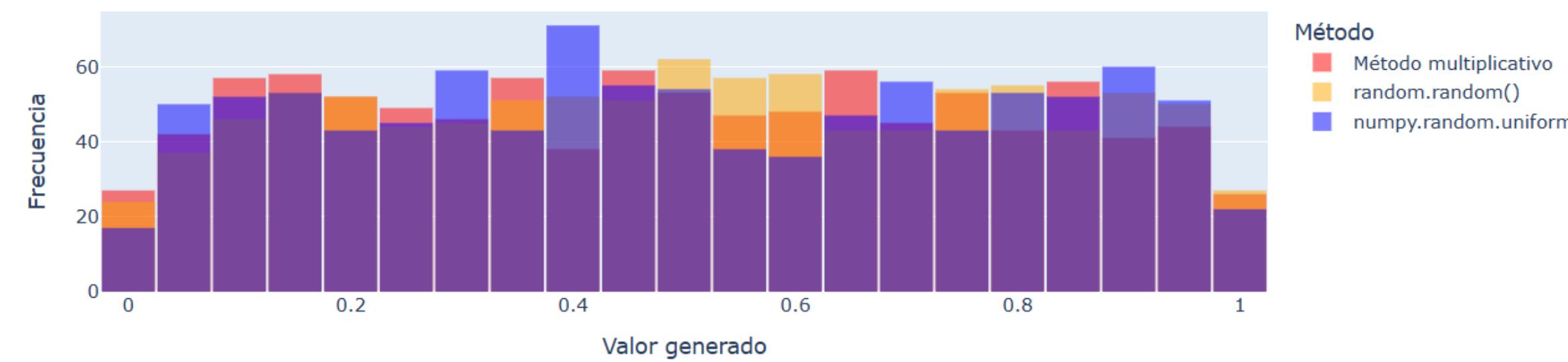
### Comparación: Método multiplicativo vs numpy.random.uniform



### Comparación: random.random() vs numpy.random.uniform



### Comparación: Multiplicativo vs random.random() vs numpy.random.uniform



## Ejemplo: Distribución Exponencial

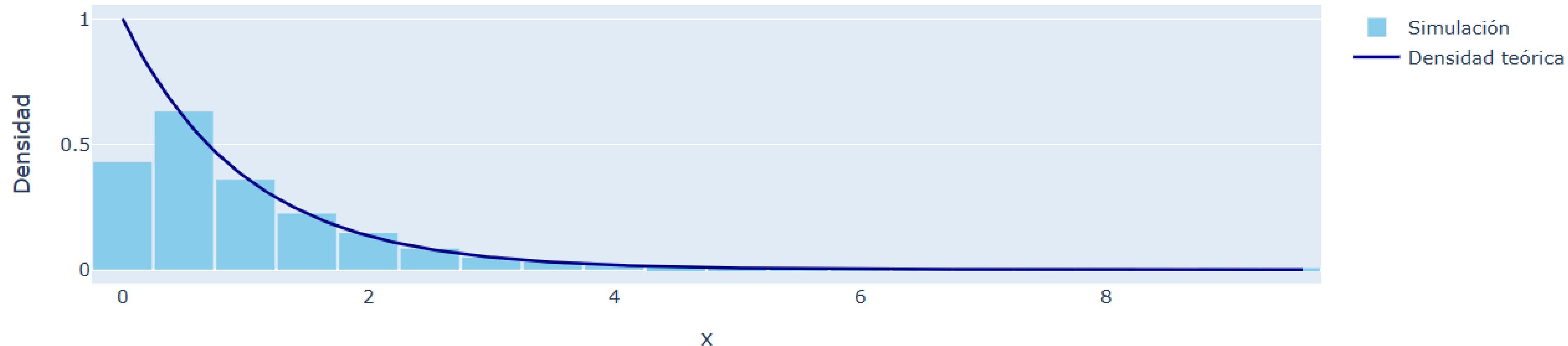
$$f(x) = \frac{1}{\theta} e^{-x/\theta}, \quad x \geq 0$$

$$F(x) = 1 - e^{-x/\theta}$$

Despejando:

$$x = -\theta \ln(R)$$

Distribución Exponencial simulada vs teórica ( $\lambda = 1$ )



## Ejemplo: Distribución Uniforme Continua

$$f(x) = \frac{1}{b-a}, \quad a \leq x \leq b$$

$$F(x) = \frac{x-a}{b-a} = R \Rightarrow x = a + (b-a)R$$

Distribución Uniforme Continua simulada vs teórica [2, 5]

