Generadores pseudoaleatorios de distribuciones de probabilidad.

Delmonti Agustín
Universidad Tecnológica Nacional Rosario
Zeballos 1341, S2000
agus.delmonti@gmail.com

Trilla Melody
Universidad Tecnológica Nacional Rosario
Zeballos 1341, S2000
trillamelo@gmail.com

Abril 2020

Resumen

En el siguiente trabajo se presentan métodos para generar distribuciones aleatorias a partir de un generador de números pseudoaleatorios. Se exploran el método de la transformada inversa, y de simulación de procesos de Bernoulli y de Poisson.

1. Introducción

La generación de variables aleatorias es un tópico muy importante en muchas áreas de las ciencias de la computación, en algoritmos estocásticos y simulaciones. En todo modelo de simulación se debe poder generar bajo demanda una o más variables aleatorias de todo tipo, fuese cual fuere su distribución de probabilidad. Surge entonces la necesidad de encontrar métodos numéricos o matemáticos para generar variables aleatorias uniformes y, para transformar esas variables en otras distribuciones. Puesto que en el curso de una simulación, los algoritmos de generación de variables aleatorias se pueden ejecutar millones de veces, una implementación computacionalmente eficiente es especialmente importante. Existen varios procedimientos para lograr este objetivo, entre ellos, el método de la transformada inversa, método de Marsaglia y métodos de simulación de procesos de Bernoulli y Poisson, los cuales desarrollaremos en el siguiente informe.

2. Metodo transformada inversa

Si se desea generar valores aleatorios x_i de una determinada variable aleatoria X cuya función de densidad esta dada por $f(x)$ se puede utilizar el método de muestreo por transformación inversa mediante el uso de su **distribución acumulada inversa** $F^{-1}(x)$.[1, 2]

Según la definición de función de densidad de probabilidad, el área total debajo de la curva es igual a 1. Esta correspondencia entre área y probabilidad queda definida para cada x_i según la distribución acumulada $F : \mathbb{R} \to [0, 1]$,

$$F(x) = P(X \leq x) \tag{1}$$

Puesto que $F(x)$ se define sobre el rango de 0 a 1, se puede utilizar un generador de números pseudoaleatorios para generar realizaciones independientes de una variable aleatoria U distribuida uniformemente en $[0, 1]$ de modo que

$$F(x) = U = \int_{-\infty}^{x} f(t)dt \tag{2}$$

Como $F(x)$ es inyectiva, x queda determinada unívocamente, es decir que es posible encontrar un valor de x_i dado un valor U. Esto es,

$$x = F^{-1}(U) \tag{3}$$

2.1. Algoritmo.

El método de la transformada inversa se basa en que se puede generar una distribución U uniforme del cual podemos muestrear valores y luego mapearlos según una distribución acumulada. En términos de implementación, esta distribución se puede generar con un **generador de números pseudoaleatorios** (PRNG) eficientemente, el cual cada lenguaje de programación provee en sus librerías. En este trabajo, para cada distribución presentada se provee su implementación en *Python 3.6* y se utiliza el *generador Mersenne-Twister* de su librería *random* por su alta calidad en la generación de números pseudoaleatorios.

2.1.1. Distribuciones continuas.

Supongamos se desea generar una variable aleatoria X con una función de distribución acumulativa F_X. El algoritmo de muestreo de transformación inversa es simple:

1. Generar $U \sim \text{Unif}(0, 1)$
2. Calcular $X = F_X^{-1}(U)$

Este algoritmo no siempre es práctico puesto que al tratar de invertir $F(x)$ en muchas distribuciones la integral es analíticamente imposible de calcular. Por ejemplo, invertir $F(x)$ es fácil si X es una variable aleatoria exponencial, pero es más difícil si X es una variable aleatoria normal.

2.1.2. Distribuciones discretas.

Ahora consideraremos la versión discreta del método de transformación inversa que solo toma valores enteros no negativos. Supongamos que X es una variable aleatoria discreta tal que $P(X = x_i) = p_i$, es decir puede tomar cualquiera de los valores de K con probabilidades $\{p1, ..., p_k\}$.

La distribución acumulada de probabilidad de una variable aleatoria discreta X se define de manera muy similar a la de una variable continua

$$F(x) = P(X \leq x) = \sum_{x_i \leq x} P(X = x_i) = \sum_{x_i \leq x} f(x_i) \tag{4}$$

El algoritmo procede como sigue:

1. Generar $U \sim \text{Unif}(0, 1)$
2. Determinar el índice k tal que $\sum_{j=1}^{k-1} p_j \leq U < \sum_{j=1}^{k} p_j$ y retornar $X = x_k$

3. Distribuciones continuas

3.1. Uniforme

Si X es una variable aleatoria que se distribuye uniformemente en un intervalo (a, b) entonces todo intervalo de una misma longitud tomado dentro de (a, b) tiene la misma probabilidad.[3]
Se nota $X \sim U(a, b)$ y su función de densidad está dada por

$$f(x) = \begin{cases} \frac{1}{b-a} & \text{para } a \leq x \leq b, \\ 0 & \textit{en otro caso} \end{cases} \tag{5}$$

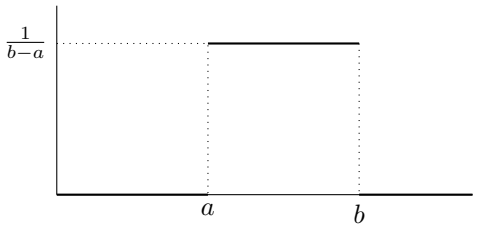


Figura 1: Función de densidad uniforme.

La función de distribución acumulada para algún $a \leq x \leq b$ se obtiene analíticamente con la integral

$$F(x) = P(X \leq x) = \int_a^x \frac{1}{b-a} dt = \frac{x-a}{b-a} \tag{6}$$

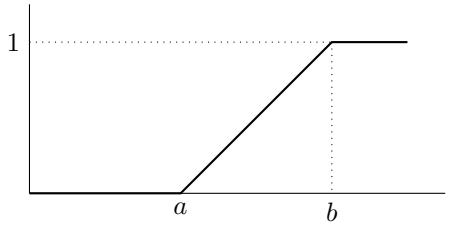


Figura 2: Función acumulada uniforme.

Luego, el generador de la distribución uniforme es $x = F^{-1}(U)$

$$x = (b - a)U + a \tag{7}$$

3.1.1. Implementación

A continuación se presenta la implementación en Python con un generador Mersenne-Twister

```
import random as rn

def uniform(a, b):
    r = rn.random()
    return (b-a)*r + a
```

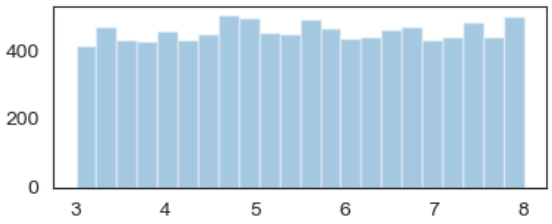


Figura 3: generación de 10k números con una distribución uniforme con $a = 3, b = 8$.

3.2. Triangular

Proporciona una primera aproximación cuando hay poca información disponible, de forma que sólo se necesita conocer el mínimo (valor pesimista), el máximo (valor optimista) y la moda (valor más probable). Estos tres valores son los parámetros que caracterizan a la distribución y se denotan por a, b y c respectivamente, con $a \leq c \leq b$.[4]

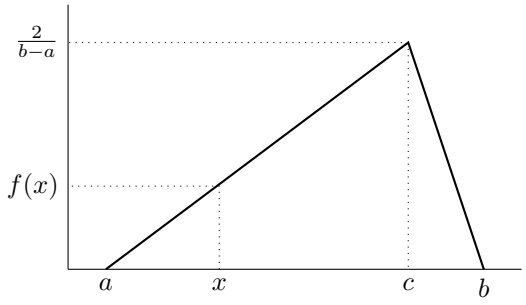


Figura 4: Función de densidad triangular.

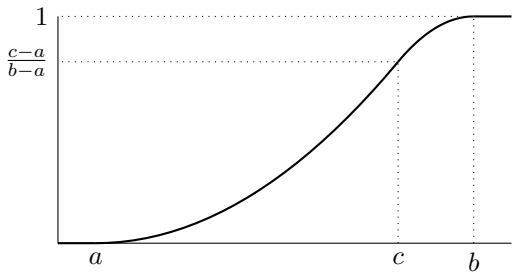


Figura 5: Función acumulada triangular.

Una particularidad de la distribución es que el valor más probable ocurre en la moda c. Esto se puede observar en el pico de la función de densidad en la Figura, donde $f(c) = \frac{2}{b-a}$ y que determina una función de densidad a tramos.
Aplicando semejanza de triángulos se puede deducir una expresión para la función de densidad

$$f(x) = \begin{cases} \frac{2(x-a)}{(b-a)(c-a)} & \text{para } a \leq x \leq c, \\ \frac{2(b-x)}{(b-a)(b-c)} & \text{para } c < x \leq b, \\ 0 & \text{en otros casos} \end{cases} \tag{8}$$

Demostración. Sea x, con $a \leq x \leq c$, por semejanza de triángulos la proporción

$$\frac{f(x)}{x-a} = \frac{2}{(b-a)(c-a)} \tag{9}$$

se mantiene. Así mismo, si $c < x \leq b$, la proporción

$$\frac{f(x)}{b-x} = \frac{2}{(b-a)(b-c)} \tag{10}$$

también se mantiene. Reescribiendo se obtiene la función a tramos de la Ecuación 8. □

La función de distribución acumulada de la distribución triangular está dada por el área bajo la curva

$$F(x) = \begin{cases} 0 & x < a \\ \frac{(x-a)^2}{(b-a)(c-a)} & \text{para } a \leq x \leq c, \\ 1 - \frac{(b-x)^2}{(b-a)(b-c)} & \text{para } c < x \leq b, \\ 1 & x > b \end{cases} \tag{11}$$

Demostración. Sea x, con $a \leq x \leq c$, el área bajo la curva es el área del triangulo cuya base es $(x-a)$ y su altura $f(x)$. Luego,

$$F(x) = \frac{(x-a)f(x)}{2} = \frac{(x-a)^2}{(b-a)(c-a)} \tag{12}$$

Dado que el área bajo la curva es 1, por la definición de función de densidad, $F(x) = P(X < x) = 1 - P(X > x)$. Si $c < x \leq b$ entonces la función acumulada se puede escribir con respecto al área del triangulo cuya base es $(b-x)$ y su altura $f(x)$ como sigue:

$$F(x) = 1 - \frac{(b-x)f(x)}{2} = 1 - \frac{(b-x)^2}{(b-a)(b-c)} \tag{13}$$

Reescribiendo se obtiene la función a tramos de la Ecuación 11. □

Luego el generador de la distribución es $x = F^{-1}(U)$

$$x = \begin{cases} \sqrt{U(c-a)(b-a)} + a & , U < \frac{c-a}{b-a} \\ -\sqrt{(1-U)(b-c)(b-a)} + b & , U \geq \frac{c-a}{b-a} \end{cases} \tag{14}$$

3.2.1. Implementación

```
import random as rn
import math

def triangular(a, b, c):
    prop = (c - a) / (b - a)
    r = rn.random()

    if r < prop:
        n = r * (c - a) * (b - a)
        return math.sqrt(n) + a
    else:
        n = (1 - r) * (b - c) * (b - a)
        return -math.sqrt(n) + b
```

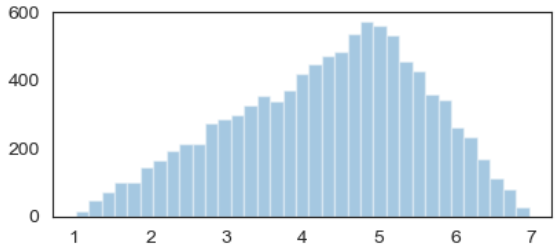


Figura 6: generación de 10k números con una distribución triangular con $a = 1, b = 7, c = 5$.

3.3. Exponencial

Una variable aleatoria X se distribuye de forma exponencial con un parámetro λ si su función de densidad esta dada por

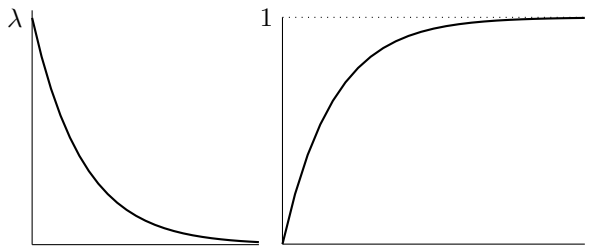
$$f(x) = \alpha e^{-\alpha x}, \textit{con } x > 0 \tag{15}$$

Se nota $X \sim exp(\lambda = \alpha)$ donde λ es el recíproco de la media de la distribución.

La función de distribución acumulada de la distribución exponencial es la integral

$$F(x) = P(X \leq x) = \int_0^x \alpha e^{-\alpha t} dt = 1 - e^{-\alpha x} \tag{16}$$

Luego, aplicando la Ecuación (2) el generador de la distribución es $x = F^{-1}(U)$



(a) Función de densidad exponencial. (b) Función acumulada exponencial.

$$x = \frac{-\ln(1-U)}{\alpha} \tag{17}$$

Debido a la simetría de la distribución uniforme sigue la posibilidad de intercambiar $1-U$ por U por motivos prácticos[1] y consecuentemente:

$$x = \frac{-\ln U}{\alpha} \tag{18}$$

3.3.1. Implementación

```
import random as rn
import numpy as np

def exp(alpha):
    r = rn.random()
    return -np.log(r)/alpha
```

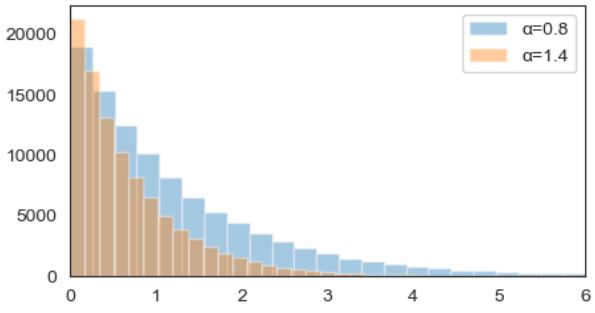


Figura 8: generación de 2 distribuciones exponenciales (n = 10k) variando el parámetro α.

3.4. Normal

Si X es una variable aleatoria que se distribuye de forma normal su función de densidad esta dada por

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \tag{19}$$

Los parámetros de la distribución μ y σ son la media y desviación estándar de la distribución respectivamente.

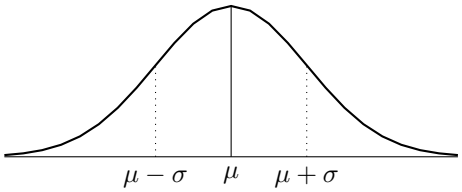


Figura 9: Función de densidad normal.

La función de distribución acumulada de la distribución normal estándar, generalmente denotada con la letra mayúscula griega Φ, es la integral

$$F(x) = \phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-t^2/2}\, dt \tag{20}$$

Puesto a que esta integral *no puede expresarse en términos de funciones elementales*, no se puede hallar una expresión de la acumulada inversa que permita utilizar el método de la transformada inversa. A continuación se plantea el **método polar de Marsaglia** (una versión más eficiente que el método de Box-Muller).

3.4.1. Marsaglia

El método polar de Marsaglia produce correctamente valores con una distribución normal.

Se toman dos valores independiente y uniformemente distribuidos en el intervalo cerrado $[-1, +1]$, y se producen tuplas (u,v) pertenecientes al plano cartesiano $[-1, 1] \times [-1, 1]$ como se muestra en la Figura 10.

Puesto que queremos quedarnos con los pares (u,v) dentro del circulo unitario, se deben generar s hasta que

$$0 < s = R^2 = u^2 + v^2 < 1 \tag{21}$$

Como u y v están distribuidos uniformemente y porque solo se han admitido puntos dentro del círculo unitario, los valores de s se distribuirán uniformemente en el intervalo abierto $(0, 1)$.[5]

Luego devuelve el par requerido de variables aleatorias normales como

$$\left(u\sqrt{\frac{-2\ln(s)}{s}}, v\sqrt{\frac{-2\ln(s)}{s}}\right) \tag{22}$$

donde $u/\sqrt{s}$ y $v/\sqrt{s}$ representan el coseno y el seno del ángulo que forma el vector (u, v) con el eje x.

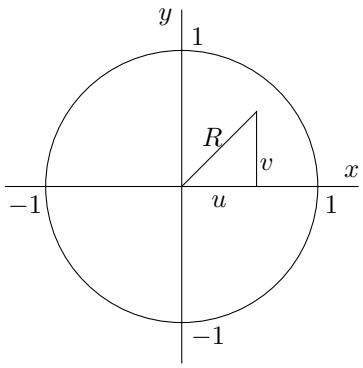


Figura 10: circulo unitario para el criterio de rechazo de pares de Marsaglia.

Como el método genera dos distribuciones normales independientes, al generar un valor se puede retornar uno y guardar el otro para la próxima solicitud de un número aleatorio.

3.4.2. Implementación

```
import math

def normal(mu=0, sigma=1):
    s = 0
    while s == 0 or s >= 1:
        u = uniform(-1, 1)
        v = uniform(-1, 1)
        s = u ** 2 + v ** 2

    k = math.sqrt(-2 * np.log(s)/ s)
    z1 = u * k * sigma + mu
```

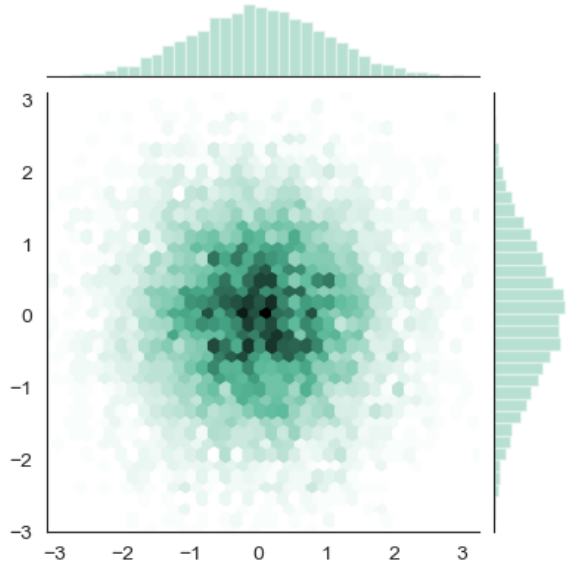


Figura 11: distribución conjunta de pares generados por el metodo Marsaglia (n = 10k) con $\mu = 0, \sigma = 1$.

```
    z2 = v * k * sigma + mu
    return z1, z2
```

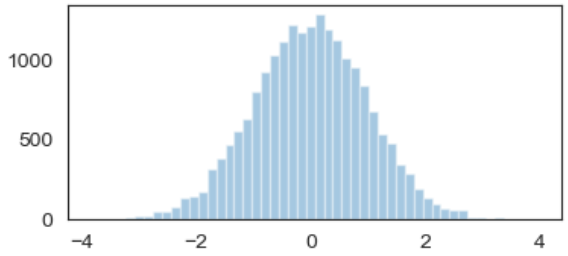


Figura 12: generación una distribución normal (n = 20k) con $\mu = 0, \sigma = 1$.

4. Distribuciones discretas

4.1. Empírica

Una distribución empírica es aquella para la cual cada posible evento se le asigna una probabilidad derivada de la observación experimental de una mues-

tra. Por ejemplo, si se toma una muestra $x_m = \{x_1, x_2, .., x_n\}$ de n observaciones, la distribución de la muestra esta dada por la frecuencia relativa $f(x_i)$ de cada observación.

La frecuencia acumulada $F(x)$ es la suma de las frecuencias relativas hasta x inclusive

$$F(x) = P(X \leq x) = \sum_{i=1}^{n} f(x_i) \tag{23}$$

Las observaciones de las muestras pueden ser resumidas en alguna tabla como la Tabla 1.

x	1	2	3	4	5	6
f(x)	0.2	0.18	0.1	0.1	0.12	0.3
F(x)	0.2	0.38	0.48	0.58	0.7	1.0

Cuadro 1: función de densidad empírica discreta.

El generador de la distribución consiste en aplicar el método de la transformada inversa para variables discretas descrito previamente en la Sección 2.1.2. Se genera un número $U \sim (0, 1)$ y luego se encuentra $X = k$ tal que $P(X \leq x_{k-1}) < U \leq P(X \leq x_k)$. Por ejemplo, según el ejemplo de la tabla presentada anteriormente, si $u = 0{,}61$ entonces $x = 5$ puesto que $P(X \leq 4) = 0{,}58 < u \leq P(X \leq 5) = 0{,}7$.

Nótese que como la transformada inversa es un mapeo directo, las variables con más probabilidad abarcan una proporción mayor del intervalo $[0; 1]$ generado por la distribución uniforme y naturalmente son más frecuentes.

4.1.1. Implementación

```
import random as rn
import numpy as np

def empirical_discrete(fx):
    cum = np.cumsum(fx)
    r = rn.random()

    for k in range(len(cum)):
        if r < cum[k]:
            return k+1
```

La función recibe como parámetro una lista con las probabilidades $P(X = x_i) = p_i$ de cada categoría i. Luego la función calculará la distribución acumulada en cada categoría. Algunos ejemplos de funciones de densidades son

```
fx0 = [.2, .3, .4, .1]
fx1 = [.11, .12, .09, .08, .12, .1,
       .09, .09, .1, .1]
```

A continuación se muestra la generación de 10k números siguiendo la distribución $f_1(x)$ de la Tabla 1.

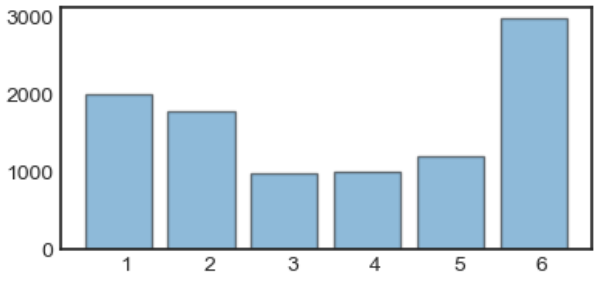


Figura 13: generación de 10k números según distribución empírica con densidad $f(x)$.

4.2. Geométrica

Sea $X \sim Geom(p)$, X representa el numero de ensayos de Bernoulli independientes necesarios para alcanzar el primer éxito, donde p es la probabilidad de éxito de un ensayo.[6]

Si la probabilidad de éxito en cada suceso es p, entonces la probabilidad de que el k-ésimo ensayo sea el primer éxito es:

$$P(X = k) = (1 - p)^{k-1} p \tag{24}$$

Se deduce entonces la función de distribución acu-

mulada $F(x)$,

$$\begin{aligned} F(x) &= P(X \leq x) \\ &= \sum_{j=1}^{x} P(X = j) \\ &= \sum_{j=1}^{x} (1-p)^{j-1} p \\ &= \frac{(1-p)^x - 1}{(1-p) - 1} p \\ &= 1 - (1-p)^x \ \forall x \geq 1 \end{aligned} \tag{25}$$

Podemos utilizar el método de la transformada inversa para generar valores de la distribución geométrica utilizando la definición de la Sección 2.1.2,

$$\begin{aligned} & F(x-1) \leq U < F(x) \Rightarrow \\ & \quad \Rightarrow 1 - (1-p)^{x-1} \leq U < 1 - (1-p)^x \\ & \quad \Rightarrow (1-p)^{x-1} \geq 1 - U > (1-p)^x \\ & \quad \Rightarrow (x-1)\ln(1-p) \geq \ln(1-U) > x \ln(1-p) \\ & \quad \Rightarrow x - 1 \leq \frac{\ln(1-U)}{\ln(1-p)} < x \end{aligned} \tag{26}$$

Puesto que $x \in \mathbb{Z}$, solo existe un número que satisfaga la ecuación

$$\therefore x = 1 + \left\lfloor \frac{\ln 1 - U}{\ln(1-p)} \right\rfloor = \left\lfloor \frac{\ln U}{\ln(1-p)} \right\rfloor \tag{27}$$

Otra posibilidad para generar valores de una distribución geométrica es utilizar la propia definición de la misma. Es decir, realizar ensayos de Bernoulli hasta obtener un éxito y retornar el índice k de la iteración.

4.2.1. Implementación

Se presenta la generación de variables con distribución geométrica a través del método de ensayos sucesivos de Bernoulli

```
import random as rn

def bernoulli(p):
    r = rn.random()
    if r < p:
        return 1
    else:
        return 0

def geometric(p):
    x = 1
    while bernoulli(p) != 1:
        x += 1
    return x
```

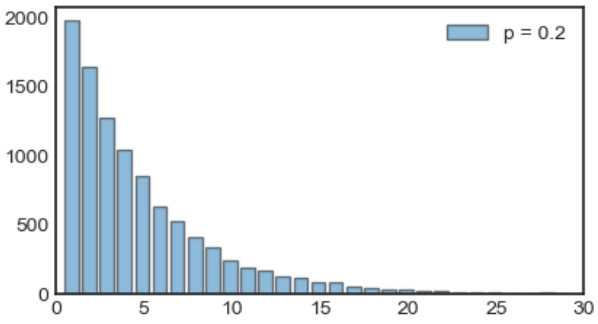


Figura 14: generación de una distribución geométrica (n = 10k) con parámetro $p = 0{,}2$.

4.3. Binomial

Cuenta el número de éxitos en una secuencia de n ensayos de Bernoulli independientes entre sí, con una probabilidad fija p de ocurrencia del éxito entre los ensayos.[7] Para representar que una variable aleatoria X sigue una distribución binomial de parámetros n y p, se escribe: $X \sim B(n, p)$.
Su función de probabilidad es:

$$f(x) = \binom{n}{x} p^x (1-p)^{n-x}, \ 0 \leq p \leq 1 \tag{28}$$

La función de distribución acumulada se puede expresar como:

$$F(x) = \Pr(X \leq x) = \sum_{i=0}^{x} \binom{n}{i} p^i (1-p)^{n-i} \tag{29}$$

Puesto a que repetir el calculo de la combinatoria no es muy eficiente, para generar números según una distribución binomial podemos recurrir a la definición de la misma y simular n eventos de Bernoulli.

4.3.1. Implementación

```
import random as rn

def bernoulli(p):
    r = rn.random()
    if r < p:
        return 1
    else:
        return 0


def binomial(n, p):
    x = 0
    for i in range(n):
        x += bernoulli(p)
    return x
```

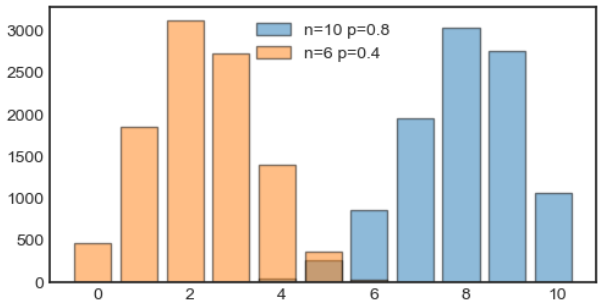


Figura 15: generación de 2 muestras con distribuciones binomiales (n = 10k) variando los parámetros $n = \{6; 10\}$ y $p = \{0{,}4; 0{,}8\}$.

4.4. Hipergeométrica

La distribución hipergeométrica se representa con la notación $X \sim H(N, K, n)$ y describe la probabilidad de obtener k éxitos (muestras aleatorias para los cuales el objeto tomado tiene una característica específica) en muestras tomadas de tamaño n *sin reposición* de una población finita de tamaño N que contiene exactamente K objetos con la característica especifica. La función de densidad de una variable aleatoria que sigue una distribución hipergeometrica esta dada por

$$P(x) = P(X = x) = \frac{\binom{K}{x}\binom{N-K}{n-x}}{\binom{N}{n}} \tag{30}$$

Puesto que es un muestreo sin reposición, cuando se elige un elemento de la población, no se lo puede volver a elegir y la probabilidad de que un elemento sea seleccionado aumenta con cada ensayo.[8]

Su función acumulada está dada por la siguiente suma

$$P(X = x) = \binom{N}{n}^{-1} \sum_{k=0}^{[x]} \binom{d}{k}\binom{N-d}{n-k} \tag{31}$$

Como fue en el caso de la distribución Binomial, calcular la combinatoria no es muy eficiente. Se puede recurrir a la definición de la distribución Hipergeometrica y simular los eventos individuales.

4.4.1. Implementación

```
import random as rn
import numpy as np

def hypergeometric(N,K,n):
    x = 0
    c = N - K
    k = K

    for i in range(n-1):
        r = rn.random()
        if r <= k/N:
            x += 1
            k -= 1
        else:
            c -= 1
        N -= 1
```

```
    return x
```

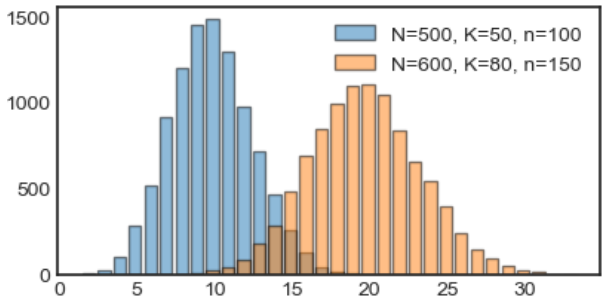


Figura 16: generación de dos distribuciones hipérgeometricas (n = 10k) variando los parámetros $\{N = 500, K = 50, n = 100\}$ y $\{N = 600, K = 80, n = 150\}$.

4.5. Poisson

Expresa, a partir de una frecuencia de ocurrencia media, la probabilidad de que ocurra un determinado número de eventos durante cierto período de tiempo. [9] Se nota $X \sim P(\lambda)$,donde λ es la media o el número de veces que se espera que ocurra el fenómeno durante un intervalo de tiempo. La función de densidad de probabilidad es

$$f(x) = P(X = x) = \frac{e^{-\lambda}\lambda^x}{x!} \tag{32}$$

La función de distribución acumulada de Poisson tiene la siguiente forma:

$$F(X) = P(X \leq x) = \sum_{i=0}^{x} \frac{e^{-\lambda}\lambda^i}{i!} \tag{33}$$

La distribución de Poisson esta relacionada con la distribución exponencial, de modo que el numero de veces que ocurre un determinado suceso en un intervalo de tiempo de longitud unidad sigue una distribución de Poisson de parámetro λ si y solo si los tiempos entre sucesos son independientes y se distribuyen según una distribución $Exp(\lambda)$. Haciendo uso de esta propiedad, podemos generar valores de una distribución $P(\lambda)$ generando valores $Y \sim Exp(\lambda)$ y contando la cantidad de sucesos i hasta que $\sum_i Y_i > 1$. Esto se puede simplificar utilizando la expresión hallada para una variable exponencial donde $Y = -\ln U/\lambda$:

$$\begin{aligned} \sum_i Y_i > 1 &\Leftrightarrow -\frac{1}{\lambda}\sum_i \ln U_i > 1 \\ &\Leftrightarrow \ln \prod_i U_i < -\lambda \\ &\Leftrightarrow \ln \prod_i U_i < -\lambda \\ &\Leftrightarrow \prod_i U_i < e^{-\lambda} \end{aligned} \tag{34}$$

4.5.1. Implementación

```
import matplotlib.pyplot as plt
import random as rn
import math

def poisson(alpha):
    x = 0
    p = 1

    while p >= math.exp(-alpha):
        r = rn.random()
        p *= r
        x += 1
    return x
```

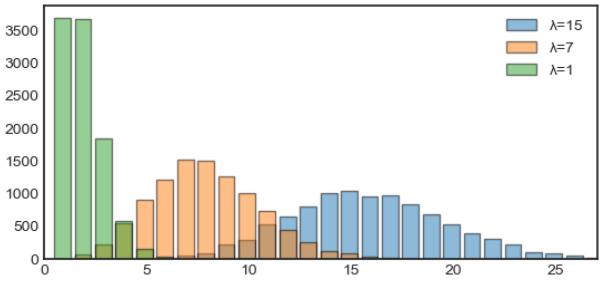


Figura 17: generación de 3 muestras con distribuciones de Poisson (n = 10k) variando el parámetro $\lambda = \{1, 7, 15\}$.

Distribucion	Metodo	Code	Test
	Continua		
Uniforme	T. inversa	Si	Si
Triangular	T. inversa	Si	Si
Exponencial	T. inversa	Si	Si
Normal	Polar Marsaglia	Si	Si
	Discreta		
Empirica	T. inversa	Si	Si
Geometrica	Composicion (Bernoulli)	Si	Si
Binomial	Composicion (Bernoulli)	Si	Si
Hipergeometrica	Composicion (Bernoulli)	Si	Si
Poisson	Composicion (Poisson)	Si	Si

Cuadro 2: resumen de distribuciones realizadas el informe.

5. Conclusión

En este trabajo se emplearon distintos métodos para generar valores según distribuciones de probabilidad, de forma rápida y eficiente ,a partir de un generador de números pseudoaleatorios uniforme. Exploramos las distintas distribuciones y los métodos de generación mas convenientes para estas, lo cual se resume en la Tabla 2.

Referencias

[1] Thomas H. Naylor, Joseph L. Balintfy, and Donald S. Burdick. *Computer simulation techniques*. John Wiley Sons, 1968.

[2] Wikipedia contributors. Inverse transform sampling — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Inverse_transform_sampling&oldid=955212807, 2020. [Online; accessed 7-June-2020].

[3] http://www.ub.edu/stat/GrupsInnovacio/Statmedia/demo/Temas/Capitulo4/B0C4m1t1.htm, 2020. [Online; accessed 7-June-2020].

[4] Distribución triangular. https://www.sergas.es/Saude-publica/Documents/1899/Ayuda_Epidat_4_Distribuciones_de_probabilidad_Octubre2014.pdf, 2020. [Online; accessed 8-June-2020].

[5] Wikipedia contributors. Boxmuller transform — Wikipedia, the free encyclopedia, 2020. [Online; accessed 10-June-2020].

[6] Distribución geométrica. https://webs.um.es/mpulido/miwiki/lib/exe/fetch.php?id=amio&cache=cache&media=wiki:simt3.pdf, 2020. [Online; accessed 9-June-2020].

[7] Wikipedia contributors. Distribución binomial — Wikipedia, the free encyclopedia. https://es.wikipedia.org/wiki/Distribuci%C3%B3n_binomial, 2020. [Online; accessed 8-June-2020].

[8] Distribución hipergeométrica. https://support.minitab.com/es-mx/minitab/18/help-and-how-to/probability-distributions-and-random-data/supporting-topics/distributions/hypergeometric-distribution/, 2020. [Online; accessed 8-June-2020].

[9] Wikipedia contributors. Distribución de poisson — Wikipedia, the free encyclopedia. https://es.wikipedia.org/wiki/Distribuci%C3%B3n_de_Poisson, 2020. [Online; accessed 9-June-2020].