Generación de variables aleatorias discretas
Método de la Transformada Inversa

Georgina Flesia

FaMAF

9 de abril, 2013

Generación de v.a. discretas

Existen diversos métodos para generar v.a. discretas:

- Transformada Inversa
- De aceptación-rechazo, o método de rechazo.
- De composición.
- Métodos mejorados según la distribución.

Método de la Transformada Inversa

- X: una variable aleatoria discreta con probabilidad de masa

$$P(X = x_j) = p_j, \qquad j = 0, 1, \ldots.$$

- $U \sim \mathcal{U}(0, 1)$: simulación de una v.a. con distribución uniforme.
- Método de la transformada inversa:

$$X = \begin{cases} x_0 & \text{si } U < p_0 \\ x_1 & \text{si } p_0 \leq U < p_0 + p_1 \\ \vdots & \\ x_j & \text{si } p_0 + \cdots + p_{j-1} \leq U < p_0 + \cdots + p_{j-1} + p_j \\ \vdots & \end{cases}$$

$$\boxed{P(X = x_j) = P(p_0 + \cdots + p_{j-1} \leq U < p_0 + \cdots + p_{j-1} + p_j) = p_j.}$$

Método de la Transformada Inversa

$$F(x) = P(X \leq x): \text{ Función de distribución acumulada}$$

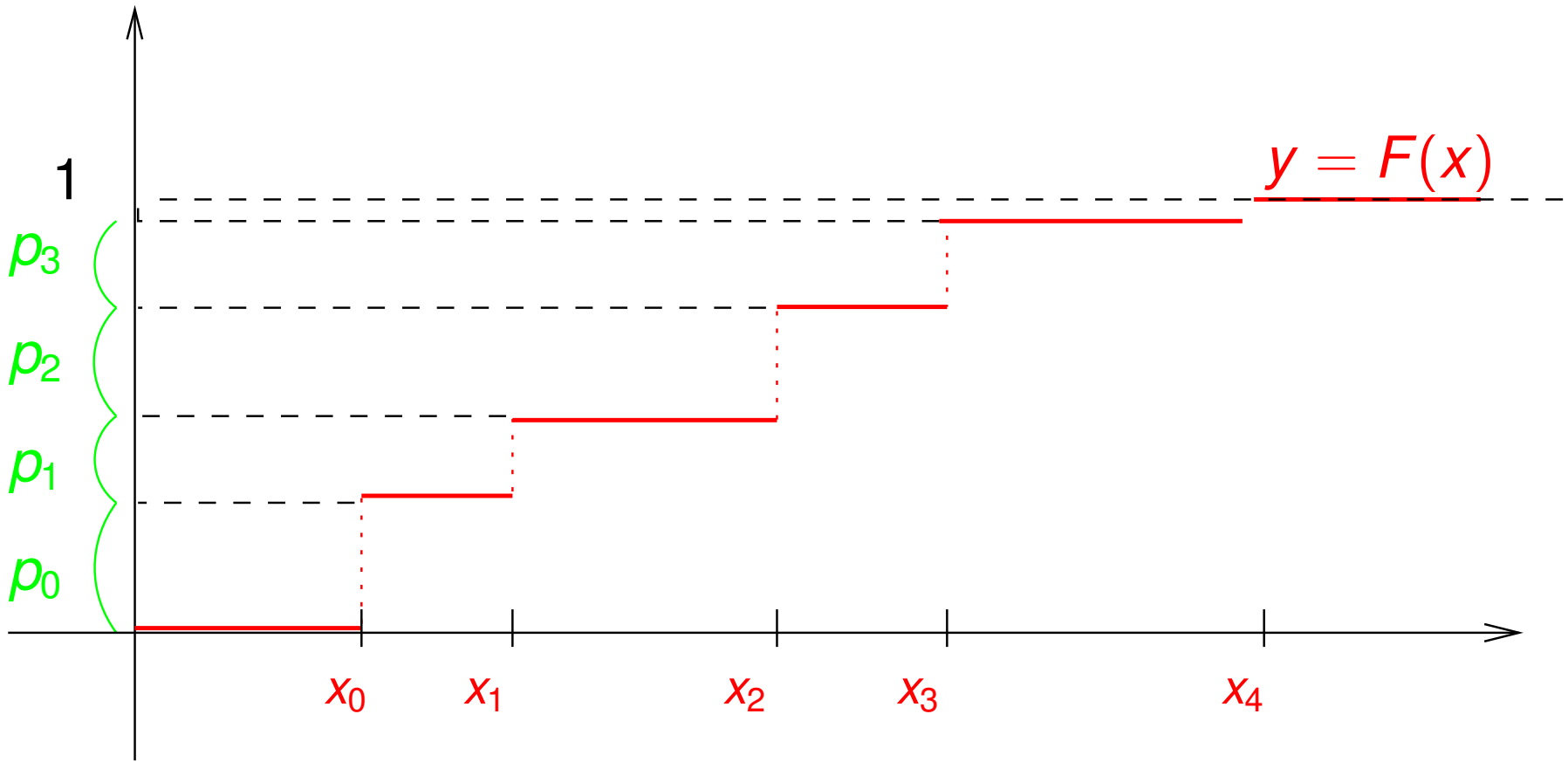
- F es una función creciente, escalonada, que toma valores entre 0 y 1.
- Si se ordenan los valores de la variable en forma creciente:

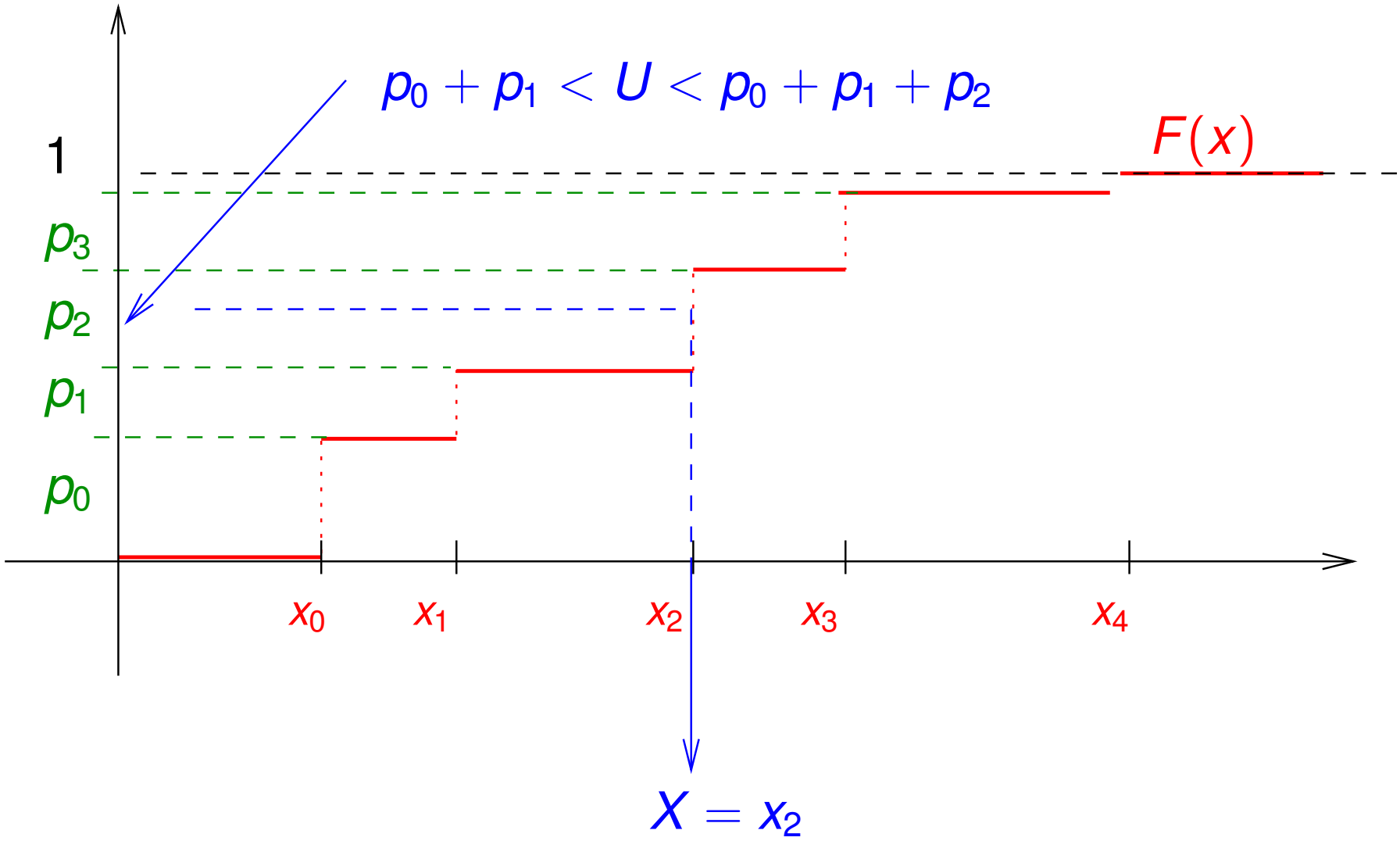
$$x_0 < x_1 < \cdots < x_n < \ldots,$$

entonces

$$F(x_j) = \sum_{k=0}^{j} p_k = p_0 + p_1 + \cdots + p_j.$$

Gráficamente




$p_0 + p_1 < U < p_0 + p_1 + p_2$
$F(x)$
1
p_3
p_2
p_1
p_0
x_0
x_1
x_2
x_3
x_4
$X = x_2$

Algoritmo

Si la v.a. toma un número finito de valores, el algoritmo es el siguiente:

Algorithm 1: Transformada Inversa

Generar $U \sim \mathcal{U}(0,1)$;
if $U < p_0$ **then**
 | $X \leftarrow x_0$ y terminar.
end
if $U < p_0 + p_1$ **then**
 | $X \leftarrow x_1$ y terminar.
end
if $U < p_0 + p_1 + p_2$ **then**
 | $X \leftarrow x_2$ y terminar.
end

$\vdots$

Algoritmo de la Transformada Inversa

Si $x_0 < x_1 < x_2 < \ldots$, entonces $F(x_j) = \sum_{i=0}^{j} p_i$, y por lo tanto

$$X \leftarrow x_0 \qquad \text{si} \qquad U < p_0 = F(x_0)$$
$$X \leftarrow x_j \qquad \text{si} \qquad F(x_{j-1}) \leq U < F(x_j)$$

Se trata de hallar el intervalo $[F(x_{j-1}), F(x_j))$ donde se ubica U:

$$U \in [F(x_{j-1}), F(x_j)) \qquad \Longleftarrow \qquad \text{Transformada Inversa}$$

Ejemplo

$X : \{1, 2, 3, 4\}$.

$$p_1 = 0.20, \quad p_2 = 0.15, \quad p_3 = 0.25, \quad p_4 = 0.40$$

Algorithm 2: Transformada Inversa

Generar U;
if $U < 0.2$ **then**
 | $X \leftarrow 1$ y terminar.
end
if $U < 0.35$ **then**
 | $X \leftarrow 2$ y terminar.
end
if $U < 0.6$ **then**
 | $X \leftarrow 3$
else
 | $X \leftarrow 4$
end

Orden decreciente de p_i

Si se ordenan de manera decreciente las probabilidades de masa $p_0, p_1, \ldots,$ se puede obtener un algoritmo más eficiente:

Algorithm 3: Ordenando p_i

Generar U;
if $U < 0.4$ **then**
  $X \leftarrow 4$ y terminar.
end
if $U < 0.65$ **then**
  $X \leftarrow 3$ y terminar.
end
if $U < 0.85$ **then**
  $X \leftarrow 1$
else
  $X \leftarrow 2$
end

Generación de una v.a. uniforme discreta

Si $X \sim \mathcal{U}[1, n]$, entonces $p_1 = p_2 = \cdots = p_n = \dfrac{1}{n}$.

$$X = j \qquad \text{si} \qquad \frac{j-1}{n} \leq U < \frac{j}{n}$$

$$j - 1 \leq nU < j$$

$$\boxed{X = \lfloor nU \rfloor + 1}$$

siendo $\lfloor x \rfloor$ la parte entera (inferior) de x.

Ejemplo

Generar una variable aleatoria discreta X, con distribución uniforme en $[5, 15]$.

- El intervalo entero $[5, 15]$ contiene 11 números enteros.
- $\lfloor 11\,U \rfloor$ es un entero en $[0, 10]$.
- $\lfloor 11\,U \rfloor + 5$ es un entero en $[5, 15]$.

Generar U;
$X \leftarrow \lfloor 11U \rfloor + 5$

Generación de una permutación aleatoria

Aplicación: Generación de permutaciones aleatorias equiprobables de un conjunto de n elementos.

Sea $P_1, P_2, \ldots, P_n$ una permutación de 1,2,. . . , n.

Algorithm 4: Permutación aleatoria de $P_1, P_2, \ldots, P_n$

$k \leftarrow n$;
while $k > 1$ **do**
    Generar U;
    $I \leftarrow [kU] + 1$;
    Intercambiar los valores de P_I y P_k;
    $k \leftarrow k - 1$
end

Ejemplo

- Conjunto de 5 elementos: a, b, c, d, e.
- k recorre el vector, de la posición $n = 5$ hasta 2.
- l: selecciona un elemento entre las posiciones 1 y k para intercambiar.

k	l
5	4
4	2
3	3
2	1

$\longrightarrow$	ae	
$\longrightarrow$	bea	$\longleftarrow$
$\longrightarrow$	c	$\longleftarrow$
$\longrightarrow$	deb	$\longleftarrow$
	ed	$\longleftarrow$

Subconjuntos aleatorios

- Si el algoritmo anterior se ejecuta para $k = n, n-1, \ldots, n-(r-1)$, se obtiene un subconjunto aleatorio de cardinal r.
- Si $r > n/2$, conviene tomar el complemento de un subconjunto aleatorio de $n - r$ elementos.
- Aplicaciones:
  - Técnica del doble ciego.
  - Simulación: Problema de las dos muestras.

Permutaciones aleatorias

Otro método para generar una permutación aleatoria de un conjunto de tamaño n consiste en

- generar n números aleatorios $U_1, U_2, \ldots, U_n$,
- ordenarlos: $U_{i_1} < U_{i_2} < \cdots < U_{i_n}$,
- utilizar los índices de los valores ordenados para hacer la permutación.

Permutaciones aleatorias

Ejemplo

Obtener una permutación aleatoria de (a, b, c, d)*.*

- $U_1 = 0.4$, $U_2 = 0.1$, $U_3 = 0.8$, $U_4 = 0.7$.
- $U_2 < U_1 < U_4 < U_3$.
- La permutación es (b, a, d, c).

Desventaja: Se requieren $O(n \log(n))$ comparaciones.

Cálculo de promedios

Ejemplo

Aproximación del valor de

$$\overline{a} = \sum_{i=1}^{n} \frac{a(i)}{n}$$

siendo que $n \gg 1$.

- Tomamos X uniforme discreta en $[1, n]$.
- $a(X)$ es una v.a. discreta con media

$$\overline{a} = E[a(X)] = \sum_{i=1}^{n} a(i)P(X = i) = \sum_{i=1}^{n} \frac{a(i)}{n}.$$

- Generamos $U_1, U_2, \ldots, U_k$ aleatorios en $(0, 1)$, $X_i = \lfloor nU_i \rfloor + 1$.
- Ley fuerte de los grandes números:

$$\overline{a} \approx \sum_{i=1}^{k} \frac{a(X_i)}{k} \qquad k \text{ grande}$$

Cálculo de promedios

Ejemplo

Utilizar 100 números aleatorios para aproximar

$$S = \sum_{i=1}^{10000} e^{\frac{i}{10000}}.$$

Algorithm 5: Promedio

$S \leftarrow 0$;
for $i = 1$ **to** 100 **do**
  Generar U;
  $I \leftarrow \lfloor 10000\, U \rfloor + 1$;
  $S \leftarrow S + \exp(I/10000)$
end
$S \leftarrow S * 100$

Variable aleatoria geométrica

$$\boxed{P(X = i) = pq^{i-1}, \qquad i \geq 1, \qquad q = (1-p).}$$

Tenemos que $F(j-1) = P(X \leq j-1) = 1 - P(X > j-1) = 1 - q^{j-1}$.

- Método de la Transformada Inversa:

$$X \leftarrow j \qquad \text{si} \qquad 1 - q^{j-1} \leq U < 1 - q^{j}$$

$$q^{j} < 1 - U \leq q^{j-1}$$

- Esto equivale a encontrar el menor j tal que $q^j < 1 - U$.

$$\boxed{X = \min\{j \ : \ q^j < 1 - U\}}$$

Variable aleatoria geométrica

Propiedades:

- El logaritmo es una función creciente,
- $\log(q)$ es negativo,
- $V = 1 - U$ también es uniforme en $(0, 1)$:

$$
\begin{aligned}
X &= \min\{j \ : \ q^j < 1 - U\} \\
&= \min\{j \ : \ j \log(q) < \log(1 - U)\} \\
&= \min\left\{j \ : \ j > \frac{\log(1 - U)}{\log(q)}\right\} \\
&= \left\lfloor \frac{\log(1 - U)}{\log(q)} \right\rfloor + 1 \\
X &= \left\lfloor \frac{\log(V)}{\log(q)} \right\rfloor + 1
\end{aligned}
$$

Variable Aleatoria Poisson

$$p_i = P(X = i) = e^{-\lambda}\frac{\lambda^i}{i!}, \qquad i = 0, 1, \ldots$$

Algorithm 6: Generación de una v.a. Poisson $P(\lambda)$

Generar $U \sim \mathcal{U}(0, 1)$;
$i \leftarrow 0$, $p \leftarrow e^{-\lambda}$, $F \leftarrow p$;
while $U \geq F$ **do**
    $p = (\lambda \cdot p)/(i + 1)$;
    $F = F + p$;
    $i = i + 1$
end
$X \leftarrow i$

Inconvenientes del Algoritmo 1

- El algoritmo chequea desde 0 en adelante hasta obtener el valor deseado.
- El número de comparaciones es 1 más que el valor generado.
- El valor generado esperado es λ. Luego, el promedio de comparaciones es $\lambda + 1$.
- Si $\lambda \gg 1$, el algoritmo anterior realiza muchas comparaciones.

Mejora al Algoritmo 1

Para mejorar el algoritmo se utilizan las siguientes propiedades de la distribución de Poisson:

- Definición recursiva de p_i: $p_{i+1} = \frac{\lambda}{i+1} p_i, \quad i \geq 0.$
- $p_{i+1} \geq p_i$ si y sólo si $\frac{\lambda}{i+1} \geq 1$, es decir, $i + 1 \leq \lambda$.
- El valor máximo de p_i es $p_{\lfloor \lambda \rfloor}$, es decir, valores de i cercanos a λ.

Algoritmo mejorado

Algorithm 7: Generación de una v.a. Poisson $P(\lambda)$

$I \leftarrow \lfloor \lambda \rfloor$;
Calcular $F(I)$ usando la definición recursiva de p_i;
Generar $U \sim \mathcal{U}(0,1)$;
if $U \leq F(I)$ **then**
 | generar X haciendo búsqueda descendente.
end
if $U > F(I)$ **then**
 | generar X haciendo búsqueda ascendente.
end

Algoritmo mejorado

- El promedio de búsquedas es

$$1 + E\left[|X - \lambda|\right].$$

- Si $\lambda \gg 1$, X aproxima a una normal, $N(\lambda, \sqrt{\lambda})$.

$$\frac{X - \lambda}{\sqrt{\lambda}} \sim N(0, 1).$$

- Promedio de búsquedas:

$$\begin{aligned} 1 + \sqrt{\lambda} E\left[\frac{|X - \lambda|}{\sqrt{\lambda}}\right] &= 1 + \sqrt{\lambda} E[|Z|] \\ &= 1 + 0.798\sqrt{\lambda}. \end{aligned}$$

Variable Aleatoria Binomial $B(n, p)$

$$p_i = P(X = i) = \frac{n!}{i!(n-i)!} p^i (1-p)^{n-i}, \qquad i = 0, 1, \ldots, n$$

- Caso similar al de generación de una v. a. Poisson.
- $$\mathrm{E}[X] = np \qquad\qquad \mathrm{Var}[X] = np(1-p).$$
- Conviene utilizar la fórmula recursiva:

$$p_{i+1} = \frac{n-i}{i+1} \frac{p}{1-p} p_i, \quad 0 \leq i < n.$$

Algoritmo para una Binomial

Algorithm 8: Generación de una v.a. Binomial $B(n, p)$

Generar $U \sim \mathcal{U}(0, 1)$;
$i \leftarrow 0$;
$c \leftarrow p/(1-p)$;
$Pr \leftarrow (1-p)^n$;
$F \leftarrow Pr$;
while $U \geq F$ **do**
    $Pr \leftarrow [c(n-i)/(i+1)]Pr$;
    $F \leftarrow F + Pr$;
    $i \leftarrow i + 1$
end
$X \leftarrow i$

Inconvenientes del algoritmo

- El algoritmo chequea desde 0 en adelante hasta obtener el valor deseado.
- El número de comparaciones es 1 más que el valor generado.
- El promedio de comparaciones es $np + 1$.
- Existen métodos alternativos más eficientes.

Métodos alternativos para generar una Binomial $B(n, p)$

- Si $p > 1/2$, generar $n - B(n, 1-p)$. (menor número de comparaciones).
- Generar $U_1, \ldots, U_n$, y contabilizar cuántas son menores que p. Es menos eficiente pues requiere n comparaciones.
- Utilizar la sugerencia análoga para Poisson. (menor número de comparaciones).
  - Valor más probable: $i = \lfloor np \rfloor$.
  - Promedio de búsquedas: $1 + \sqrt{np(1-p)}\, 0.798$.