

# Generación de variables aleatorias discretas

## Método de la Transformada Inversa

**Patricia Kisbye**

FaMAF

30 de marzo, 2010

# 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

$$P(X = x_j) = p_j, \quad j = 0, 1, \dots$$

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

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

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

# Método de la Transformada Inversa

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

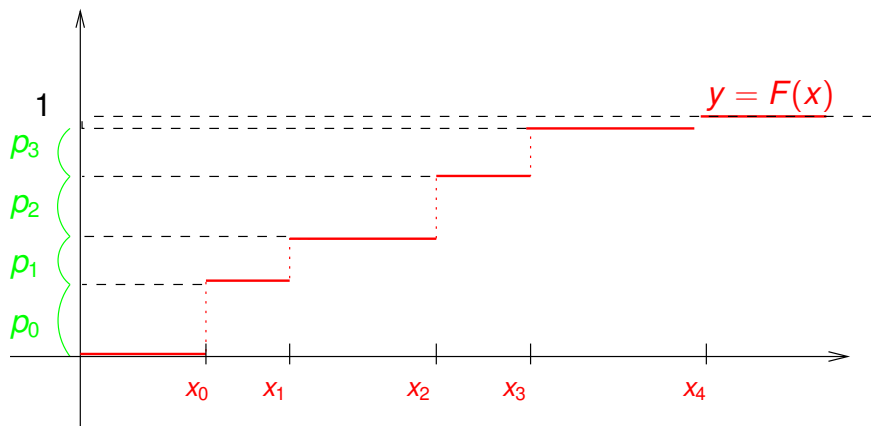
- ▶  $F$  es una función creciente, escalonada, que toma valores entre 0 y 1.
- ▶ Si se ordenan los valores de la variable en forma creciente:

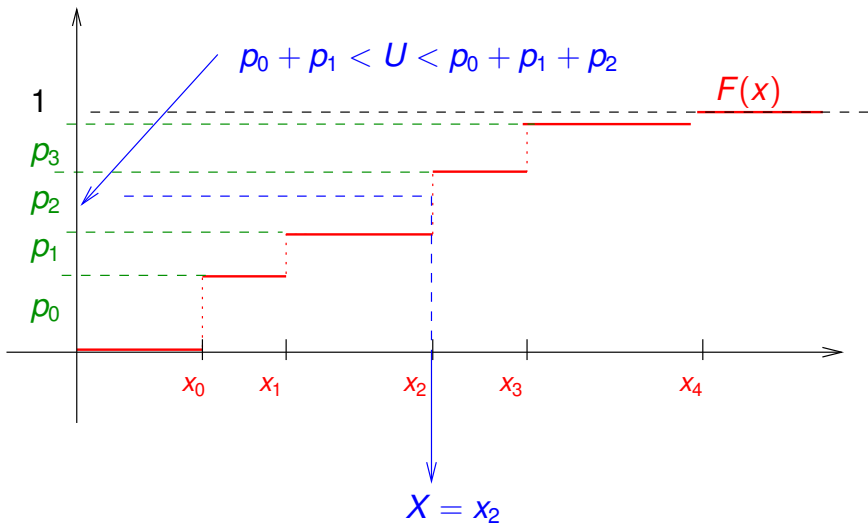
$$x_0 < x_1 < \cdots < x_n < \dots,$$

entonces

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

# Gráficamente





# Algoritmo

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

---

Algoritmo: 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**

⋮

---

# Algoritmo de la Transformada Inversa

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

$$X \leftarrow x_0 \quad \text{si} \quad U < p_0 = F(x_0)$$

$$X \leftarrow x_j \quad \text{si} \quad F(x_{j-1}) \leq U < F(x_j)$$

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

|                              |              |                      |
|------------------------------|--------------|----------------------|
| $U \in [F(x_{j-1}), F(x_j))$ | $\Leftarrow$ | Transformada Inversa |
|------------------------------|--------------|----------------------|



## Ejemplo

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

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

---

Algoritmo

---

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  $p_0, p_1, \dots$ , se puede obtener un algoritmo más eficiente:

---

Algoritmo: 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  $p_1 = p_2 = \dots = p_n = \frac{1}{n}$ .

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

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

$$X = \lfloor nU \rfloor + 1$$

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.
- ▶  $\lfloor 11U \rfloor$  es un entero en  $[0, 10]$ .
- ▶  $\lfloor 11U \rfloor + 5$  es un entero en  $[5, 15]$ .

---

Algoritmo:  $X \sim \mathcal{U}[5, 15]$

---

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

---

Algoritmo: Permutación aleatoria de  $P_1, P_2, \dots, P_n$

---

$k \leftarrow n$ ;

**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.
- ▶ La tabla se lee por columnas de izquierda a derecha.

| $l$ |   | 4 | 2 | 3 | 1 |
|-----|---|---|---|---|---|
| $k$ |   | 5 | 4 | 3 | 2 |
|     | a | a | a | a | e |
|     | b | b | e | e | a |
|     | c | c | c | c | c |
|     | d | e | b | b | b |
|     | e | d | d | d | d |

# Subconjuntos aleatorios

- ▶ Si el algoritmo anterior se ejecuta para  $k = n, n - 1, \dots, n - (r - 1)$ , 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 no equiprobables

Si consideramos el algoritmo

---

Algoritmo v2: Permutación aleatoria de  $P_1, P_2, \dots, P_n$

---

$k \leftarrow n$ ;

**while**  $k > 1$  **do**

    Generar  $U$ ;

$l \leftarrow [nU] + 1$ ;

    Intercambiar los valores de  $P_l$  y  $P_k$ ;

$k \leftarrow k - 1$

**end**

---

¿Son equiprobables las permutaciones?

- No, a menos que  $n = 2$ .

$$\frac{n^{n-1}}{n!} \quad \text{no es un número entero.}$$



# Permutaciones aleatorias

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

- ▶ generar  $n$  números aleatorios  $U_1, U_2, \dots, U_n$ ,
- ▶ ordenarlos:  $U_{i_1} < U_{i_2} < \dots < U_{i_n}$ ,
- ▶ 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*

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

*siendo que  $n \gg 1$ .*

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

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

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

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

# Cálculo de promedios

## Ejemplo

*Utilizar 100 números aleatorios para aproximar*

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

---

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

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

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

- ▶ Método de la Transformada Inversa:

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

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

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

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

# 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

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

---

Algoritmo 1: 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  $\lambda$ . 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  $p_i$ :  $p_{i+1} = \frac{\lambda}{i+1} p_i$ ,  $i \geq 0$ .
- ▶  $p_{i+1} \geq p_i$  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  $\lambda$ .

## Algoritmo mejorado

---

Algoritmo 2: 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

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

- ▶ 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 $B(n, p)$

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

- ▶ Caso similar al de generación de una v. a. Poisson.



$$E[X] = np \quad \text{Var}[X] = np(1-p).$$

- ▶ 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

---

Algoritmo A: 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  $U_1, \dots, U_n$ , 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.