

Generación de variables aleatorias discretas

Método de la Transformada Inversa

Patricia Kisbye

FaMAF

3 de abril, 2008

mzran13()

El generador mzran13 de números aleatorios combina los siguientes generadores: Combina los generadores:

$$x_n = 69069x_{n-1} + 1013904243 \pmod{2^{32}}$$

$$y_n = y_{n-2} - y_{n-3} - c' \pmod{2^{32} - 18}$$

donde c es un "bit de acarreo".

$$x_n + y_n \pmod{2^{32}}.$$

Analizamos el algoritmo en C propuesto por Zaman y Marsaglia.

Rutina propuesta

```
typedef unsigned long int unlong;
unlong x=521288629, y=362436069, z=16163801,
      c=1, n=1131199209;
unlong mzran13()
{ long int s;
  if (y>x+c) (s=y-(x+c); c=0; }
  else { s=y-(x+c)-18; c=1; }
  x=y; y=z;
  return (z=s) + (n=69069*n+1013904243);
};
void ran13set(unlong xx, unlong yy, unlong zz, long nn)
  { x=xx; y=yy; z=zz; n=nn; c=y>z; }
```

Generación de v.a. discretas

- ▶ Asumimos un generador de valores de una v.a. uniforme $U \sim \mathcal{U}(0, 1)$.
- ▶ Métodos:
 - ▶ Transformada Inversa
 - ▶ de aceptación-rechazo o método de rechazo.
 - ▶ de composición.

Método de la Transformada Inversa

Consideramos una variable aleatoria X con distribución de masa

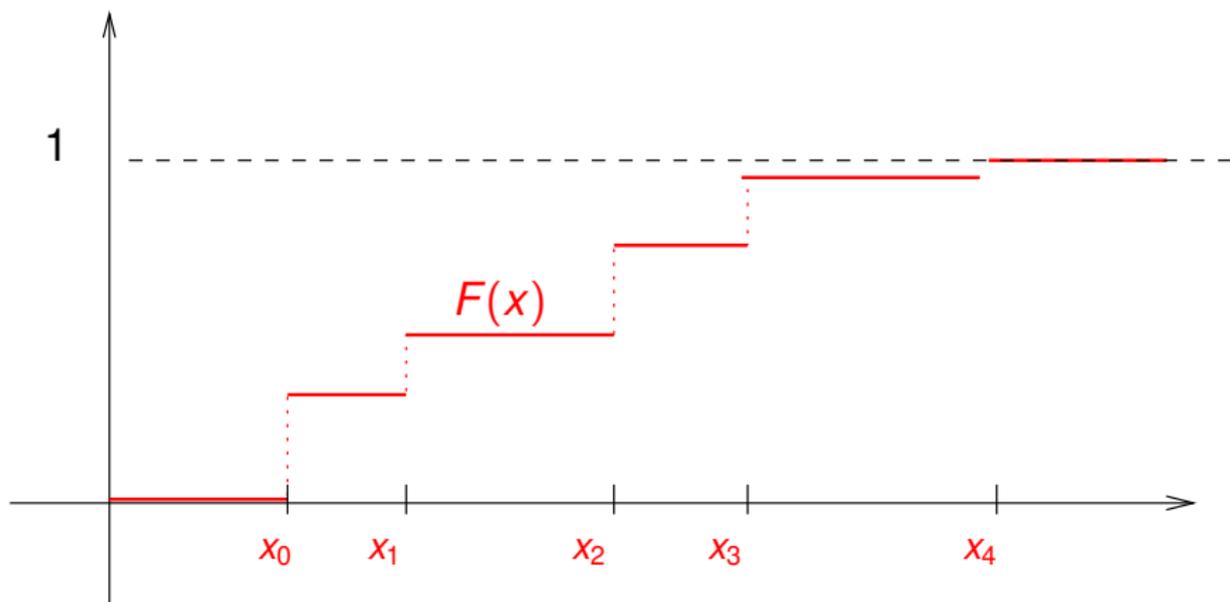
$$P(X = x_j) = p_j$$

$$j = 0, 1, \dots, \sum_j p_j = 1.$$

Su función de distribución acumulada es una función creciente, escalonada, que toma valores entre 0 y 1.

Asumimos que los valores de la variable están ordenados en forma creciente:

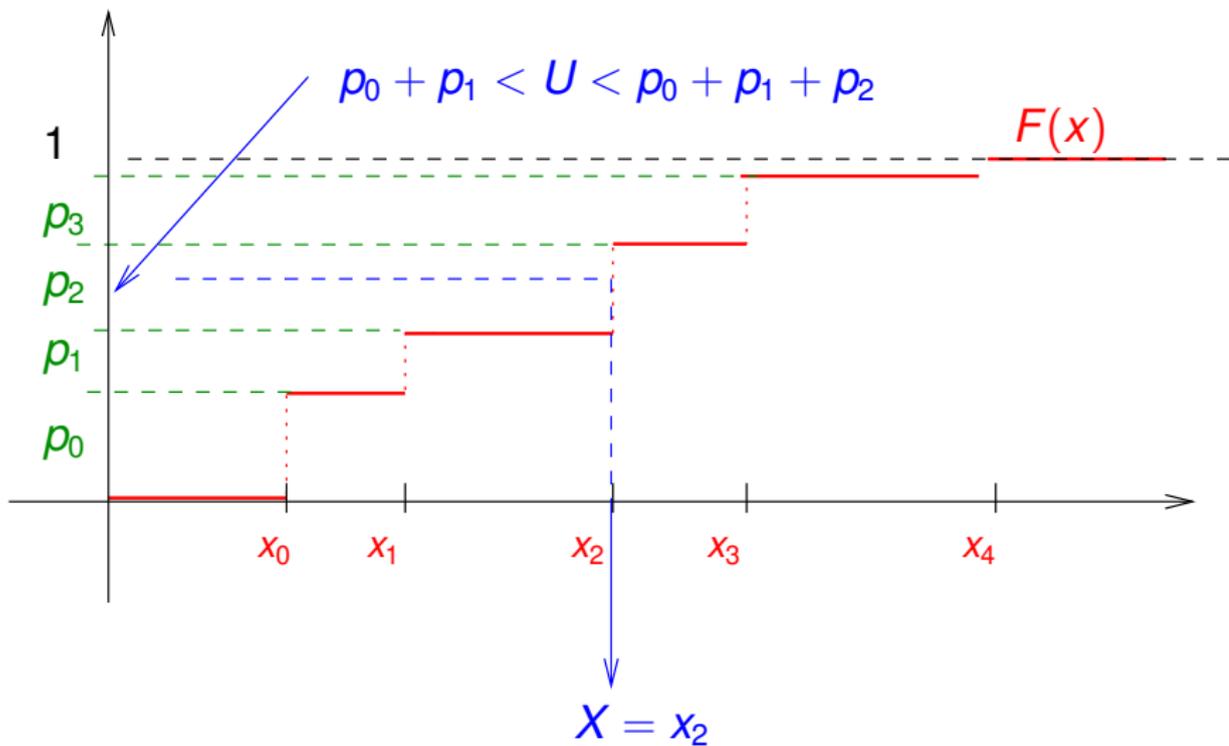
$$x_0 < x_1 < \dots < x_n < \dots$$



Para generar un valor de la variable aleatoria X , generamos un número aleatorio $U \sim \mathcal{U}(0, 1)$ y hacemos

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

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



Algoritmo:

- ▶ Generar $U \sim \mathcal{U}(0, 1)$.
- ▶ Si $U < p_0$ hacer $X = x_0$ y terminar.
- ▶ Si $U < p_0 + p_1$ hacer $X = x_1$ y terminar.
- ▶ Si $U < p_0 + p_1 + p_2$ hacer $X = x_2$ y terminar.
- ▶ ...

Notemos que si $x_0 < x_1 < x_2 < \dots$, entonces $F(x_k) = \sum_{i=0}^k p_i$, y por lo tanto

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

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

Ejemplo

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

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

- ▶ Generar $U \sim \mathcal{U}(0, 1)$.
- ▶ Si $U < 0.20$ hacer $X = 1$ y terminar.
- ▶ Si $U < 0.35$ hacer $X = 2$ y terminar.
- ▶ Si $U < 0.60$ hacer $X = 3$ y terminar.
- ▶ Si no, hacer $X = 4$.

Si se ordenan las probabilidades p_0, p_1, \dots , se puede obtener un algoritmo más eficiente:

Ejemplo

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

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

- ▶ Generar $U \sim \mathcal{U}(0, 1)$.
- ▶ Si $U < 0.40$ hacer $X = 4$ y terminar.
- ▶ Si $U < 0.65$ hacer $X = 3$ y terminar.
- ▶ Si $U < 0.85$ hacer $X = 1$ y terminar.
- ▶ Si no, hacer $X = 2$.

Para generar una v.a. uniforme discreta en $[1, n]$:

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

$$X = j \quad \text{si} \quad j-1 \leq nU < j$$

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

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

Generar U

$$X = \lfloor 11U \rfloor + 5$$

Generación de una permutación aleatoria

Aplicación: Generación de permutaciones aleatorias de un conjunto de n elementos, todas con la misma probabilidad

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

Paso 2 : $k = n$

Paso 3 : Generar U y hacer $l = [kU] + 1$.

Paso 4 : Intercambiar los valores de P_l y P_k .

Paso 5 : Hacer $k = k - 1$ y si $k > 1$ ir a Paso 3.

Paso 6 : P_1, \dots, P_n es la permutación buscada.

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

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 $1 + 2 + \dots + (n - 1) = O(n \log(n))$ comparaciones.

Un "mal" método

Si en el primer algoritmo permutamos el k -ésimo lugar de la tabla con uno de **toda** la tabla, también obtenemos permutaciones aleatorias. ¿Tienen una distribución uniforme?

No. ¿Por qué?

Sencillo:

$\frac{n^n}{n!}$ no es un número entero.

Cálculo de promedios

Ejemplo

Aproximar el valor de

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

siendo que $n \gg 1$ y $a(i)$ tiene una expresión complicada.

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

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

- ▶ 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:

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

Ejemplos

Ejemplo (Variable aleatoria geométrica)

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

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

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

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

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

Variable aleatoria geométrica

- ▶ El logaritmo es una función creciente,
- ▶ $\log(q)$ es negativo,
- ▶ usando la notación "parte entera de",
- ▶ $1 - U$ también es uniforme en $(0, 1)$:

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

Ejemplos

Ejemplo (Variable Aleatoria Poisson)

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

Usamos la identidad

$$p_{i+1} = \frac{\lambda}{i+1} p_i, \quad i \geq 0.$$

$$E[X] = \lambda \quad \text{Var}[X] = \lambda.$$

Variable aleatoria Poisson

Algoritmo

Paso 1 Generar $U \sim \mathcal{U}(0, 1)$

Paso 2 $i = 0, p = e^{-\lambda}, F = p.$

Paso 3 Si $U < F$, hacer $X = i$ y terminar.

Paso 4 $p = \frac{\lambda p}{(i + 1)}, F = F + p, i = i + 1.$

Paso 5 Volver al Paso 3.

- ▶ 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 $\lambda + 1$.

Una mejora para generar Poisson

- ▶ Si $\lambda \gg 1$, el algoritmo anterior realiza muchas comparaciones.
- ▶ Los valores más probables son los enteros más cercanos a λ .

Mejora del algoritmo

- ▶ Tomar $I = \lfloor \lambda \rfloor$.
- ▶ Calcular $F(I)$ usando la definición recursiva de p_i .
- ▶ Generar $U \sim \mathcal{U}(0, 1)$.
- ▶ Si $U \leq F(I)$, generar X haciendo búsqueda descendente.
- ▶ Si $U > F(I)$, generar X haciendo búsqueda ascendente.

$$\text{Promedio de búsquedas} \simeq 1 + 0.798\sqrt{\lambda}$$

Ejemplos

Ejemplo (Variable Aleatoria Binomial $B(n, p)$)

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

Usamos la identidad

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

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

Variable aleatoria Binomial

Algoritmo

Paso 1 Generar $U \sim \mathcal{U}(0, 1)$

Paso 2 $i = 0$, $c = p/(1 - p)$, $pr = (1 - p)^n$, $F = pr$.

Paso 3 Si $U < F$, hacer $X = i$ y terminar.

Paso 4 $pr = [c(n - i)/(i + 1)]pr$, $F = F + pr$, $i = i + 1$.

Paso 5 Volver al Paso 3.

- ▶ 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 $np + 1$.

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 . (n comparaciones)
- ▶ Utilizar la sugerencia análoga para Poisson. (menor número de comparaciones).