

Generación de números aleatorios

Estadística Computacional I

Departamento de Estadística e Investigación Operativa (Univ. Sevilla)

Introducción

Números Completamente Aleatorios

Números Pseudo-Aleatorios

Contrastes para los generadores de números aleatorios

Generación de variables aleatorias

Aplicaciones

Introducción

3

INTRODUCCIÓN

- Las técnicas de simulación son fundamentales en la Estadística y otras Ciencias:
 - ✓ Extracción de muestras
 - ✓ Simulación de distribuciones
 - ✓ Estimación de la capacidad de generalización
 - ✓ Inferencia mediante métodos de remuestreo
 - ✓ Agregación de modelos predictivos: *Bagging, Random Forests*
 - ✓ Métodos de Monte-Carlo
 - ✓ Generación de claves criptográficas
 - ✓ Integración numérica
- En la generación de secuencias de números aleatorios se distinguen tres grandes categorías:
 - **Números completamente aleatorios:** la secuencia es totalmente impredecible, se basan en mediciones de un proceso físico (por ejemplo el ruido atmosférico o la emisión de fotones).
 - **Números pseudo-aleatorios:** pretenden generar secuencias aleatorias en el intervalo (0,1), son predecibles si se conoce la “semilla”.
 - **Números cuasi-aleatorios:** se trata de generar secuencias bien distribuidas en el intervalo (0,1), también son predecibles.
- Los números pseudo y cuasi-aleatorios se generan mediante programas informáticos. La naturaleza determinística del ordenador implica que las secuencias generadas no son realmente aleatorias, pero un buen generador produce secuencias que son estadísticamente indistinguibles de realizaciones de secuencias de variables uniformes independientes.

2

4

Números Completamente Aleatorios

5

NÚMEROS COMPLETAMENTE ALEATORIOS

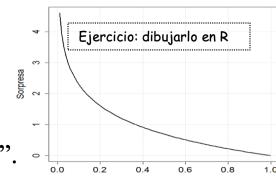
1. Números Completamente Aleatorios

- La librería **random** de R proporciona varias funciones para acceder mediante Internet a un servicio de generación de números aleatorios disponible en <http://random.org>.
- Este servicio muestrea ruido atmosférico a través de una radio que sintoniza una frecuencia no utilizada, y aplica una corrección de asimetría originalmente propuesta por John von Neumann.
- Un programa informático convierte cada muestra de ruido en un bloque de 8 bits, de los cuales se toma solo el primero, que se añade a la secuencia resultante de bits.
- La secuencia obtenida se caracteriza por un alto valor de la entropía, por tanto cabe esperar una gran diversidad o incertidumbre en la fuente de bits.
- En un experimento aleatorio con n posibles resultados, con probabilidades, p_1, \dots, p_n , la “sorpresa” de observar el resultado i -ésimo, como le llamó Tribus en 1961, es $-\log(p_i)$. Shannon definió la **entropía** como la media de las n sorpresas:

$$H(p_1, \dots, p_n) = -\sum_{i=1}^n p_i \log(p_i)$$

Si se utilizan logaritmos neperianos, la unidad de medida es el “nart”.
En base 2, se mide en bits. Expresiones alternativas:

$$H(p_1, \dots, p_n) = \sum_{i=1}^n p_i \log\left(\frac{1}{p_i}\right) = E[\log(1/p)] = E[-\log p]$$

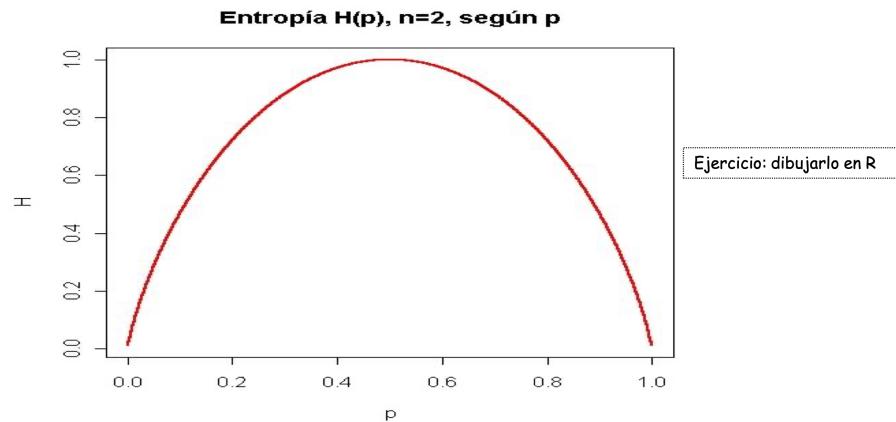


3

6

En el caso $n=2$, se puede trabajar con la llamada **entropía binaria**. Definiendo $p=p_1$,

$$H(p_1, p_2) = H_2(p) = \begin{cases} 0 & \text{si } p \in \{0,1\} \\ p \log \frac{1}{p} + (1-p) \log \frac{1}{1-p} & \text{si } p \in (0,1) \end{cases}$$



La máxima incertidumbre corresponde a la distribución uniforme discreta, la mínima se da en distribuciones degeneradas.

4

7

- La fuente de bits resultante del muestreo del ruido es sometido a una corrección de asimetría propuesta por Von Neumann:
 - ✓ Se leen los bits en bloques de 2
 - ✓ Si ambos son iguales se eliminan y se leen los dos siguientes
 - ✓ Si son distintos (01 o bien 10), se pasa el primero.
- Existen otros métodos más eficientes para asegurar que el 0 y el 1 son equiprobables.
- La librería **random** dispone de diversas funciones para convertir la secuencia resultante de bits en números enteros o en bytes.

□ Ejemplo:

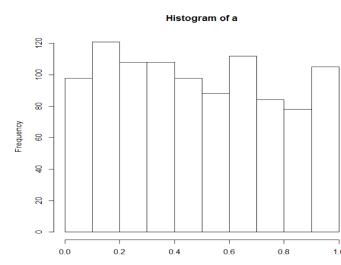
Generar 10 números aleatorios entre 0 y 1000, guardarlos en una matriz de 5 columnas, y dividir el resultado entre 1000 para obtener una muestra de tamaño 10 de una ley U(0,1).

```
library(random)
randomNumbers(n=10, min=0, max=1000, col=5)/1000

V1      V2      V3      V4      V5
[1,] 0.850  0.548  0.814  0.692  0.623
[2,] 0.623  0.387  0.660  0.064  0.224
```

La imagen adjunta es el histograma de 1000 valores generados de forma similar.

Ejercicio: hacerlo en R



5

8

Números Pseudo-Aleatorios

9

NÚMEROS PSEUDO-ALEATORIOS

2. Números Pseudo-Aleatorios

- ❑ La generación de números completamente aleatorios depende de recursos cuyo acceso puede ser difícil y lento: fenómenos físicos, análisis de alguna magnitud, conexión remota...
- ❑ En los ordenadores se suele trabajar con secuencias de números pseudo-aleatorios, que son secuencias determinísticas a partir de una configuración inicial definida por un parámetro llamado *semilla*. Si no se da este valor, se toma a partir de la hora del ordenador.
- ❑ Un buen generador producirá secuencias que son estadísticamente indistinguibles de muestras aleatorias simples de una ley uniforme.
- ❑ Ripley (*Stochastic Simulation*, 1990) señala las siguientes propiedades deseables:
 - ✓ Los números deben seguir una ley Uniforme
 - ✓ Deben ser independientes
 - ✓ La longitud de la secuencia entre dos números idénticos debe ser muy larga
 - ✓ La secuencia debe ser impredecible si se ignora la semilla.
- ❑ Basta con disponer de secuencias de realizaciones de una ley $U(0,1)$, ya que cualquier distribución puede ser simulada a partir de dicha ley.
- ❑ Un generador de números pseudo-aleatorios viene definido por los siguientes elementos:
 - S : conjunto finito de estados
 - P : función de probabilidad sobre S , llamada distribución de probabilidad inicial
 - Una función de transición $f: S \rightarrow S$
 - U : Conjunto finito de los símbolos de salida
 - Una función de salida o transferencia $g: S \rightarrow U$

6

10

La generación de la secuencia se basa en el siguiente algoritmo:

1. Generar un estado inicial (semilla) s_0 según P y calcular $u_0=g(s_0)$.
2. Repetir este proceso obteniendo $s_i=f(s_{i-1})$, $u_i=g(s_i)$, $i=1,2,\dots$

En general la semilla se determina a partir de la hora del sistema, y la secuencia de las u_i cabe esperar que sea indistinguible de una secuencia de realizaciones de la $U(0,1)$.

El periodo, característica fundamental, es el menor entero n tal que $s_{k+n}=s_k$ para cualquier k .

2.1. Generador de congruencia lineal

La función de transición es de la forma $f(x)=(ax+c) \bmod m$, donde a es el multiplicador, c es el incremento, y m es el módulo. Todos los elementos son números naturales, por lo que $S=\mathbb{N}$.

La secuencia de estados será $x_i=(ax_{i-1}+c) \bmod m$.

c y m se toman primos entre sí, y a se elige de modo que $ax \bmod m \neq 0$ para todo x natural.

Cuando $c=0$ se tiene el algoritmo de Park-Miller o Lehmer.

Es evidente que el periodo no puede ser superior a m , pero se puede maximizar con las siguientes condiciones:

- El incremento c y el módulo m son primos relativos
- $a-1$ es múltiplo de todo primo que divide a m
- $a-1$ es un múltiplo de 4 cuando m es un múltiplo de 4

7

11

Las funciones de transferencia habituales son:

$$g(x)=x/m, \quad g(x)=x/(m-1), \quad g(x)=(x+1/2)/m$$

Este generador está disponible en la función `congruRand` de la librería `randtoolbox`. Por defecto construye una secuencia de Park-Miller con $a=16807$, $m=2^{31}-1$ y $c=0$.

Ejercicio: hacerlo en R

2.2. Generadores recursivos múltiples

Son una generalización del anterior, se basa en la siguiente recurrencia, donde k es un entero positivo fijo, y se requieren k semillas iniciales:

$$x_n = (a_1 x_{n-1} + \dots + a_k x_{n-k} + c) \bmod m$$

Un caso particular es el algoritmo Knuth-TAOCP-2002, que construye secuencias retardadas de Fibonacci:

$$x_n = (x_{n-37} + x_{n-100}) \bmod 2^{30}$$

Su periodo es aproximadamente 2^{129} .

La función `knuthTAOCP` de `randtoolbox` la implementa.

8

12

2.3. Algoritmo de Mersenne-Twister

Los anteriores métodos no explotan la estructura binaria de los ordenadores. Matsumoto y Nishimura propusieron este generador que sí lo hace, ya que utiliza operaciones binarias que se pueden implementar de manera muy eficiente.

Un número natural x se representa mediante un vector de w bits (por ejemplo 32).

Se considera la siguiente ecuación recurrente para el término $n+i$

$$x_{i+n} = x_{i+m} \oplus \{(x_i^{upp} | x_{i+1}^{low})A\}$$

$n > m$ son constantes enteras, A es una matriz $w \times w$ de bits (0,1), $|$ es el operador concatenación, x_i^{upp} es la secuencia de los $w-r$ bits superiores de x_i , x_{i+1}^{low} es la secuencia de los r bits inferiores de x_{i+1} , y \oplus es el operador suma binaria.

A partir de n valores iniciales x_0, \dots, x_{n-1} (que dependen de la semilla inicial x_0) se genera una secuencia de enteros en $0, 1, \dots, 2^w - 1$.

Si se divide cada entero por 2^w se obtienen valores en $[0,1]$ (función de transformación).

9

13

Para mejorar la equidistribución, se le añadieron algunas operaciones.

Los principales parámetros son $w=32$, $n=624$, $m=397$, $r=31$.

El periodo es $2^{nw-r}=2^{19937}-1$.

Ventajas:

- Periodos muy superiores a los métodos anteriores,
- Una equidistribución mejor,
- Tiempo de cálculo es más reducido al basarse en operaciones que se implementan de manera eficiente en ordenadores.

La función `runif` de R, y el paquete `randtoolbox` disponen de implementaciones.

Ejercicio: ilustrar estas operaciones en R

10

14

Contrastes para los generadores de números aleatorios

15

CONTRASTES PARA LOS GENERADORES DE NÚMEROS ALEATORIOS

3. Contrastes para los generadores de números aleatorios

Se trata de comprobar si la secuencia resultante se corresponde con secuencias de realizaciones independientes de una $U(0,1)$. También hay procedimientos para secuencias en $\{0,1\}$.

3.1. Contrastes sobre una secuencia de n números

Bondad de ajuste

Los contrastes de bondad de ajuste comparan la función de distribución empírica de la secuencia con la función de distribución de la $U(0,1)$. Se pueden destacar los tests de Kolmogorov-Smirnov, Cramer-von Mises y Anderson-Darling. Según L'Ecuyer y Simard (2007) no son recomendables para esta tarea.

Test de huecos (gap test) (hueco: secuencia de valores 0 consecutivos)

Investiga patrones especiales en la secuencia de los U_i , $i=1,\dots,n$. Se toma un subintervalo $[l,u]$ del intervalo $[0,1]$ y se definen las variables $G_i=1$ si U_i cae en $[l,u]$, $G_i=0$ en otro caso.

Si los $U_i \sim U(0,1)$, $P[G_i=1]=u-l=p$.

Sea n_j el número de huecos de longitud j (10001 es un hueco de longitud 3).

Tomando $j=1,2,\dots,m$, para un m suficientemente grande, se construye un test chi-cuadrado comparando los n_j y los valores esperados np_j , donde la probabilidad de observar un hueco de longitud j es $p_j=(1-p)p^2$.

Ejercicio: probar con gap.test de randtoolbox

L'Ecuyer, P. and Simard, R. (2007). Testu01: A c library for empirical testing of random number generators. *ACM Trans. on Mathematical Software* 33(4) Art. 22.

11

16

Test de orden

Trata con otro tipo de patrones. Fijado d , divisor del tamaño n ($n=md$) de la secuencia generada, se consideran todas las subsecuencias de tamaño d :

$$\underline{x_1, \dots, x_d}, \underline{x_{d+1}, \dots, x_{2d}}, \underline{x_{2d+1}, \dots, x_{3d}}, \dots, \underline{x_{(m-1)d+1}, \dots, x_{md}}$$

Para un vector de d valores consecutivos de la secuencia, hay $d!$ posibles ordenaciones de esos valores. Por ejemplo para $d=3$: $\{123, 132, 213, 231, 312, 321\}$. Bajo el supuesto de aleatoriedad, las $d!$ ordenaciones son equiprobables.

Sea n_j el número de veces que se da la ordenación j . Se construye un test chi-cuadrado de bondad de ajuste basado en el estadístico:

$$S = \sum_{j=1}^{d!} \frac{\left(n_j - m \frac{1}{d!}\right)^2}{m \frac{1}{d!}}$$

Ejercicio: probar con order.test de randtoolbox

Test de frecuencias

Se considera un conjunto de d enteros consecutivos $J=\{j_1, \dots, j_d\}$. A partir de la secuencia de los U_i , se calculan los enteros asociados $m_i = [U_i * (j_d + 1) + j_1]$, siendo $[]$ el operador parte entera. Por tanto los m_i toman valores en J , y se distribuyen según una ley uniforme discreta (bajo H_0). El número esperado de enteros m_i igual a j_k es $e=n/(j_d - j_1 + 1)$, cualquiera que sea j_k . De nuevo se aplica un test chi-cuadrado.

Ejercicio: implementarlo y probar con freq.test de randtoolbox

$$S = \sum_{k=1}^d \frac{(\#\{m_i = j_k\} - e)^2}{e}$$

12

17

3.2. Contrastes basados en secuencias múltiples

Bajo la hipótesis de que las $U_i \sim U(0,1)$, un vector (U_i, \dots, U_{i+d-1}) se distribuye uniformemente en el hipercubo unitario $[0,1]^d$. La idea es particionar el hipercubo en celdas, y comparar el número de vectores que caen en cada celda con los valores esperados.



U_{i+1}

Test serial

En la función serial.test de **randtoolbox** se calcula una serie de parejas de enteros (j_i, j_{i+1}) , $j_i = [U_i * d]$, d es el parámetro dimensión ($d=8$ por defecto), por lo tanto los posibles valores de j_i son $\{0, 1, \dots, d-1\}$. Además $P[j_i=r]=1/d$, $r=0, 1, \dots, d-1$.

Nótese que los j_i indican a qué intervalo pertenece U_i cuando el intervalo $[0,1]$ se divide en d intervalos de la misma amplitud.

Realizando la misma división para U_{i+1} se tiene los correspondientes j_{i+1} , dividiendo así el cuadrado unitario en d^2 celdas, todas equiprobables bajo el supuesto de uniformidad.

Sea n_k el número de pares (U_i, U_{i+1}) en la celda k , es decir, $j_i d + j_{i+1} = k$, numerando las celdas según los valores $0, 1, \dots, d^2-1$. Teniendo en cuenta que todos los pares de dos elementos en $\{0, \dots, d-1\}$ son equiprobables, con probabilidad $p=1/d^2$, se construye un test chi-cuadrado de bondad de ajuste comparando las frecuencias observadas y las esperadas.

En la función serial.test se construyen los pares mediante
pares= matrix(floor(U * d), length(U)/2, 2)

13

18

Test de colisiones

Se trabaja con m secuencias de tamaño n resultantes del generador, y se consideran k celdas de $[0,1]^t$, donde t ha sido previamente fijado.

Dada una secuencia de tamaño n , (U_1, \dots, U_n) , se generan los posibles subvectores de tamaño t , $v_i = (U_i, \dots, U_{i+t-1})$. Se dice que v_i presenta una colisión cuando cae en una celda previamente ocupada por otro vector anterior.

Para cada secuencia (U_1, \dots, U_n) se calcula el número de colisiones c .

Denotando por C la variable aleatoria número de colisiones,

$$P[C = c] = \prod_{i=0}^{n-c-1} \frac{k-i}{k} \frac{1}{k^c} \binom{n-c}{c}, \quad c = 0, 1, \dots, n-1$$

${}_2S_n^{n-c}$ es el número de Stirling de segunda clase:

$${}_2S_n^r = r \cdot {}_2S_{n-1}^r + r \cdot {}_2S_{n-1}^{r-1}, \quad {}_2S_n^1 = {}_2S_1^r = 1$$

Dado que su cálculo directo es inviable para n grande, al requerir una cantidad de tiempo de CPU $O(n \log(n))$ en esos casos se utiliza una aproximación por la ley Normal (cuya varianza tiene una expresión complicada) o bien Poisson $P(n^2/(2k))$.

A partir de las m secuencias, se realiza un test chi-cuadrado comparando los valores observados y esperados para cada valor posible de C .

14

19

Test póker

Sea k el parámetro número de cartas. Dados n números resultantes del generador, n múltiplo de k , se consideran las n/k secuencias de k valores consecutivos.

Para cada una de estas secuencias, se define una mano de cartas $\{I_1, \dots, I_k\}$, siendo $I_i = [U_i * k]$. Cada carta es un entero en $\{0, \dots, k-1\}$. Sea n_j el número de manos con exactamente j cartas diferentes, lo que ocurre con probabilidad

$$p_j = \frac{1}{k^k} \frac{k!}{(k-j)!} {}_2S_k^j$$

El test chi-cuadrado se basa en

$$S = \sum_{j=1}^k \frac{\left(n_j - \frac{np_j}{k} \right)^2}{\frac{np_j}{k}}$$

Test de rachas

Sirve para contrastar la independencia de las observaciones. A partir de la secuencia, se trabaja con el número de rachas, entendiendo por racha una secuencia de valores consecutivos con el mismo signo (comparando con 0.5). (Implementación en `runs.test` de la librería `tseries`).

Ejemplo: 0.7 0.6 0.4 0.8 0.3 → ++ - + - → 4 rachas

Test de autocorrelaciones

Herramienta habitual en la modelización ARIMA de series temporales, se estiman las autocorrelaciones hasta un determinado retardo y se trabaja con la banda de confianza para determinar si existen autocorrelaciones significativas, lo que iría en contra de la independencia de las observaciones que componen la secuencia generada.

15

20

Generación de variables aleatorias

21

GENERACIÓN DE VARIABLES ALEATORIAS

4. Generación de variables aleatorias

A partir de una m.a.s. $U_1, \dots, U_n \sim U(0,1)$ se pueden generar muestras de cualquier distribución paramétrica mediante alguna de las diversas técnicas existentes. Como ilustración, se presenta a continuación la generación de algunas distribuciones.

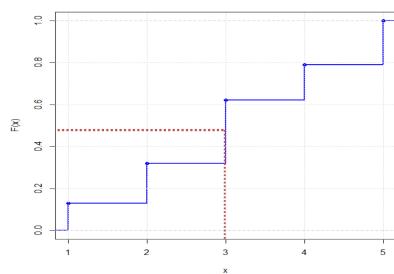
Uniforme discreta

Suponiendo d valores posibles, la variable $X = [U*d] + 1$ puede tomar cualquiera de los valores enteros $\{1, 2, \dots, d\}$. $P[X=j] = P\{(j-1) \leq U*d < j\} = j/d - (j-1)/d = 1/d$.

En este ejemplo se ha usado el método de *transformación*.

Variable discreta o categórica con probabilidades conocidas

Si una variable X toma k valores $\{1, 2, \dots, k\}$ con probabilidades $\{p_1, \dots, p_k\}$ y función de distribución $F(x)$, la variable $Y = F^{-1}(U) = \inf\{x | F(x) \geq U\}$ sigue dicha distribución.



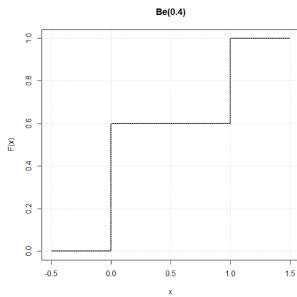
Esta propiedad siempre se verifica (**Inversión**):

Teorema. Si X es una v.a. con f.D. $F(x)$, y $U \sim U(0,1)$, la variable aleatoria $Y = F^{-1}(U)$ se distribuye según la función de distribución $F(x)$.

16

Bernouilli

Para generar $X \sim Be(p)$, se puede aplicar el método de inversión :



$$F^{-1}(u) = \begin{cases} 0 & u \leq 1-p \\ 1 & 1-p < u \end{cases}$$

También se puede definir, de forma equivalente,

$$X = \begin{cases} 0 & p \leq u \\ 1 & u < p \end{cases}$$

Binomial

Se generan n realizaciones independientes de una $Be(p)$, la suma sigue una ley $Bi(n,p)$.

Normal

El algoritmo de Box-Muller se basa en el procedimiento de transformación utilizando dos realizaciones de la $U(0,1)$.

Se generan $U_1, U_2 \sim U(0,1)$. Las variables X e Y que se definen a continuación son independientes y siguen una ley $N(0,1)$:

$$X = \sqrt{-2 \log U_1} \cos(2\pi U_2), Y = \sqrt{-2 \log U_1} \sin(2\pi U_2)$$

17

23

Demostración del método de Box-Muller:

$$Z = (Z_1, Z_2) = T(U_1, U_2) = (\sqrt{-2 \log U_1} \cos(2\pi U_2), \sqrt{-2 \log U_1} \sin(2\pi U_2))$$

$$Z_1^2 + Z_2^2 = -2 \log U_1; \quad Z_2 / Z_1 = \tan(2\pi U_2)$$

$$(U_1, U_2) = T^{-1}(Z_1, Z_2) = (e^{-(Z_1^2 + Z_2^2)/2}, (2\pi)^{-1} \arctan(Z_2 / Z_1))$$

$$J(z) = \begin{bmatrix} -z_1 e^{-(z_1^2 + z_2^2)/2} & -z_2 e^{-(z_1^2 + z_2^2)/2} \\ -\frac{1}{2\pi} \frac{z_2}{z_1^2} \frac{1}{1 + (z_2^2/z_1^2)} & \frac{1}{2\pi} \frac{1}{z_1} \frac{1}{1 + (z_2^2/z_1^2)} \end{bmatrix} \Rightarrow |J(z)| = \frac{1}{2\pi} e^{-(z_1^2 + z_2^2)/2}$$

$$f_Z(z_1, z_2) = f_{U_1, U_2}(T^{-1}(z_1, z_2)) |J(z)| = \frac{1}{2\pi} e^{-(z_1^2 + z_2^2)/2} = \frac{1}{\sqrt{2\pi}} e^{-z_1^2/2} \frac{1}{\sqrt{2\pi}} e^{-z_2^2/2}$$

■

Para generar realizaciones de una normal cualquiera:

$$X = \mu + \sigma \sqrt{-2 \log U_1} \cos(2\pi U_2), Y = \mu + \sigma \sqrt{-2 \log U_1} \sin(2\pi U_2)$$

18

24

Log-Normal

Si X sigue una ley Normal, $\exp(X)$ se distribuye según una Log-Normal.

Chi-cuadrado

Para k grados de libertad, se suman los cuadrados de k variables $N(0,1)$.

Exponencial

Si X sigue una ley exponencial $E(\lambda)$ y por tanto con función de distribución $F(x)=1-e^{-\lambda x}$, la inversa de F proporciona fácilmente una realización de la exponencial:

$$-\log(1-U)/\lambda \text{ o bien } -\log(U)/\lambda.$$

Gamma

Para una ley $X \sim Ga(n,\lambda)$ se aprovecha que es la suma de n exponenciales i.i.d. $\sim E(\lambda)$:

$$-\frac{1}{\lambda} \sum_{j=1}^n \log(U_j) = -\frac{1}{\lambda} \log \prod_{j=1}^n U_j$$

Si el primer parámetro no es entero se usan otros procedimientos.

19

25

Método del muestreo con rechazo

Se puede utilizar en el siguiente contexto:

- se conoce la expresión de dos funciones de densidad f, h
- se dispone ya de algún método para simular valores de h pero no de f
- hay una constante $M > 1$ tal que $f \leq Mh$.

El algoritmo es:

1. Generar X con densidad h , y generar $U \sim U(0,1)$.
2. Si $M \cdot U \cdot h(X) \leq f(X)$, ir a 3, en otro caso volver a 1.
3. Devolver X .

Propiedad:

En el algoritmo anterior, el número de intentos (pasos 1 y 2) sigue una ley $Ge(1/M)$. Por tanto, su media es M y el número de intentos es independiente de la variable aleatoria simulada.

Ejemplo:

Para simular de una Beta(a,b), si $a \geq 1$ y $b \geq 1$ $f(x)$ está acotada y se puede elegir la densidad de la $U(0,1)$ como $h(x)$. Según la anterior propiedad interesa un valor de M lo menor posible, se puede calcular entonces

$$M = \sup_{x \in (0,1)} \frac{f(x)}{h(x)} = \sup_{x \in (0,1)} f(x) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \left(\frac{a-1}{a+b-2} \right)^{a-1} \left(\frac{b-1}{a+b-2} \right)^{b-1}$$

En tal caso

$$MUh(X) \leq f(X) \Leftrightarrow U \leq \frac{X^{a-1}(1-X)^{b-1}}{M'}, \quad M' = \left(\frac{a-1}{a+b-2} \right)^{a-1} \left(\frac{b-1}{a+b-2} \right)^{b-1}$$

Ejercicio: hacerlo en R

20

26

Aplicaciones

27

APLICACIONES

5. Aplicaciones

La generación de números aleatorios permite construir simulaciones de diversos procesos estadísticos, por ejemplo:

- ✓ Realizar simulaciones de diversas propiedades (Teorema Central del Límite, Glivenko-Cantelli,...)
- ✓ Se pueden comparar métodos para construir contrastes de hipótesis o intervalos de confianza en diversas configuraciones (tamaño muestral, población, parámetros,...)
- ✓ Permite obtener estimaciones de características poblacionales para las que no se conoce expresión analítica
- ✓ Se pueden simular mecanismos de creación de valores perdidos y comparar técnicas de imputación

Estos procedimientos de simulación suelen ser conocidos con el nombre de métodos de Monte-Carlo.

Ejemplo:

Estimar $E\{h(X)\}$, $X \sim N(0,1)$, $h(x) = x/(2^x - 1)$.

Tras generar n realizaciones independientes X_1, \dots, X_n de la $N(0,1)$, la media de las n transformaciones proporciona una estimación razonable (si bien hay métodos para reducir la varianza del estimador)

$$\frac{1}{n} \sum_{i=1}^n h(X_i)$$

Ejercicio: hacerlo en R

21

28