Simulación

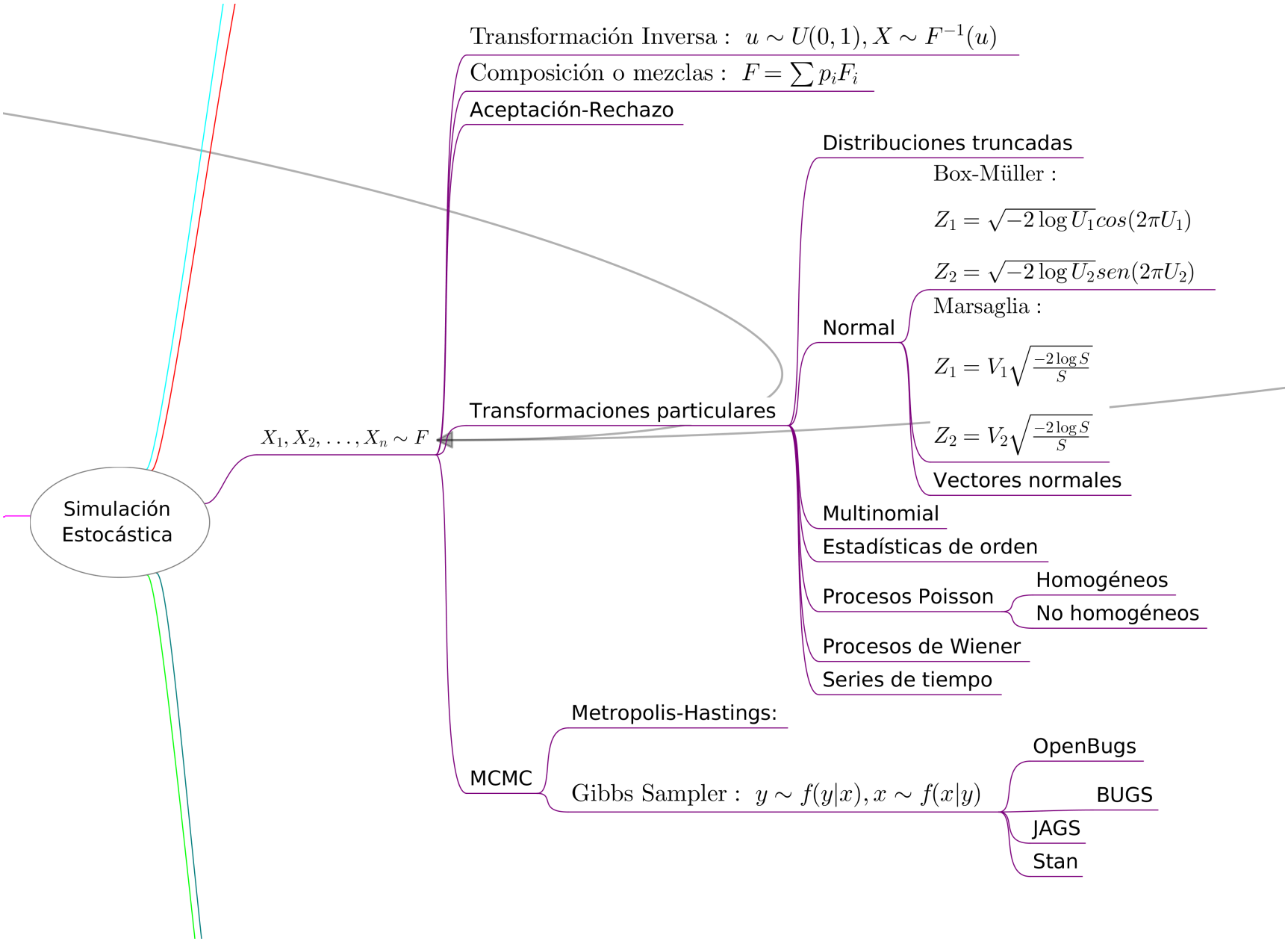
2. Generación de variables aleatorias.
2.1 Métodos generales
2.2 Métodos particulares

Jorge de la Vega Góngora

Departamento de Estadística,
Instituto Tecnológico Autónomo de México

Semana 4: 4-6 de septiembre de 2018

Generación de variables aleatorias



Métodos generales

Metodo de la transformación inversa

Teorema de la Transformación inversa

Si $X \sim F$ y F es continua y estrictamente creciente, entonces $Y = F(X)$ es una variable aleatoria con distribución $\mathcal{U}(0,1)$.

Demostración.

Sea G la distribución de Y. Entonces si $y \in (0,1)$,

$$\begin{aligned} G(y) &= P(Y \leq y) = P(F(X) \leq y) \\ &= P(X \leq F^{-1}(y)) \\ &= F(F^{-1}(y)) = y \end{aligned}$$

Entonces $G(y) = y$ en $(0,1)$. Esta es la función de distribución uniforme. □

Teorema de la función cuantíl (generalización del Teorema de la transformación inversa) I

¿Qué pasa si $X \sim F$ con F discontinua o no monótona? El teorema aplica de la misma manera, pero la prueba requiere más sustento teórico.

- Consideren X una variable aleatoria, $X : \Omega \to \mathbb{R}$ y sean $(\Omega, \mathcal{F}, P)$ y $(\mathbb{R}, \mathcal{B})$ espacios de probabilidad, donde $\mathcal{B}$ son el conjunto de los borelianos.
- Una distribución Q es una medida de probabilidad inducida por una variable aleatoria X si $Q(B) = P(X^{-1}(B))$ con $B \in \mathcal{B}$.
- Una función de distribución se puede asociar a una distribución a través de la relación definida por $F(x) = Q((-\infty, x])$.

Teorema de la función cuantíl

Sea F una función de distribución. Una variable aleatoria X se puede construir como sigue: Sea P la distribución uniforme en $(0, 1)$ y se define: $X : \Omega \to \mathbb{R}$ tal que

$$X(\omega) = \sup\{x | F(x) < \omega\} \qquad \omega \in (0, 1)$$

La variable aleatoria definida de esta manera, también se denota como $X = F_X^{-1}$, aunque la función F no sea uno a uno y estrictamente continua.

Teorema de la función cuantíl (generalización del Teorema de la transformación inversa) II

Demostración.

Basta con probar que $A = \{\omega | X(\omega) \leq x\} = \{\omega | F(x) \geq \omega\} = B$ ya que si P es la medida de Lebesgue, se sigue que $P(B) = P(A) = F(x)$.
Sea $\omega \in B$. Entonces $\omega \leq F(x)$. Así que $x \notin \{y | F(y) < \omega\}$ y por lo tanto $X(\omega) \leq x$, por lo que $\omega \in A$. Por lo tanto, $B \subset A$.
Ahora bien, si $\omega \notin B$, $F(x) < \omega$. Como F es continua por la derecha por ser función de distribución, $\exists \epsilon > 0$ tal que $F(x + \epsilon) < \omega$ y $X(\omega) \geq x + \epsilon > x$.
Entonces $\omega \notin A$. Por lo tanto $A = B$. □

Resultados de los teoremas

Los teoremas anteriores dan lugar a un algoritmo que se puede descomponer en dos casos para obtener una variable aleatoria $X \sim F$:

- Caso continuo
  1. Genera $u \sim \mathcal{U}(0,1)$
  2. Obtener $X \sim F^{-1}(u)$
- Caso discreto: Si $x \in \{x_1, \ldots, x_n\}$, el conjunto de puntos de discontinuidad de F (n puede ser ∞), entonces la transformación inversa es $F^{-1}(u) = x_i$ donde $F(x_{i-1}) < u \leq F(x_i)$.
  1. Genera $u \sim \mathcal{U}(0,1)$
  2. Tomar $x = x_i$, donde $F(x_{i-1}) < u \leq F(x_i)$.

  La solución del paso 2 puede ser difícil para algunas distribuciones. El capítulo III de Debroyé desarrolla algunos métodos para implementar el algoritmo.

Ejemplos

a. $X \sim exp(\lambda)$. Entonces X tiene función de densidad $f(x) = \frac{1}{\lambda} e^{-x/\lambda} I(x > 0)$ para $\lambda > 0$. La función de distribución es:

$$F(x') = \int_0^{x'} \frac{1}{\lambda} e^{-x/\lambda} dx = \int_0^{x/\lambda} e^{-z} dz = 1 - e^{-x/\lambda}$$

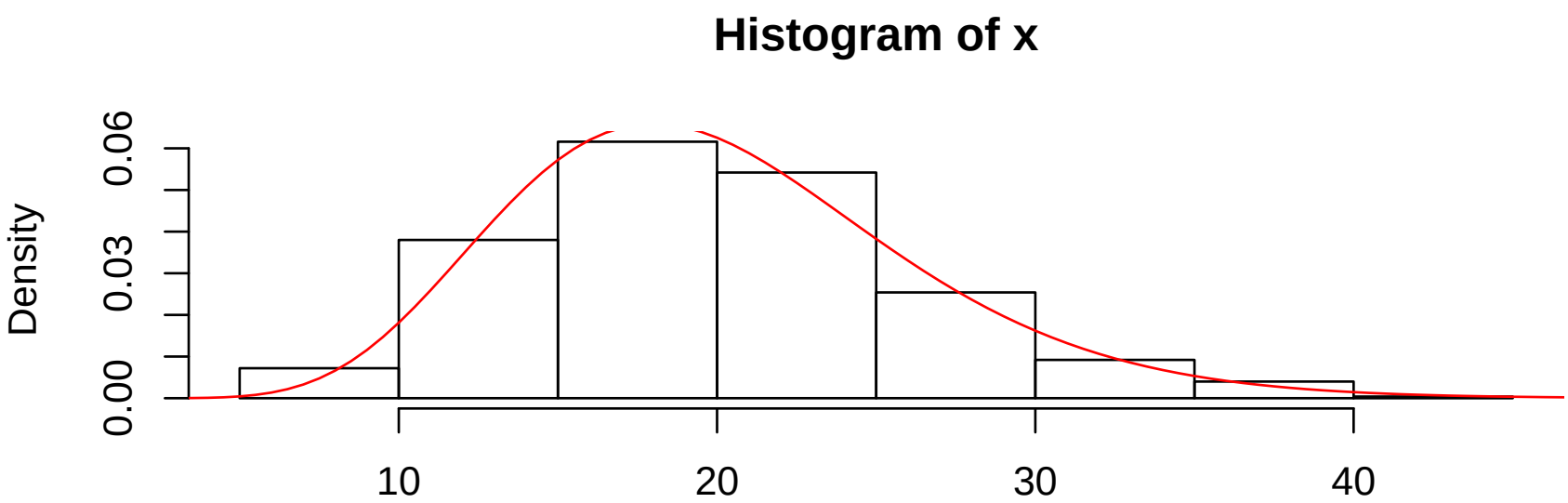
Entonces $X = F^{-1}(u) = -\lambda \log(1-u) = -\lambda \log(u)$. Ahora bien, podemos sustituir $1-u$ por u ya que $u \sim \mathcal{U}(0,1) \implies 1-u \sim \mathcal{U}(0,1)$.

Ejemplos

b. Si $X \sim \mathcal{G}(\alpha, \beta)$, entonces X tiene densidad $f(x) = \frac{1}{\Gamma(\alpha)\beta\alpha} x^{\alpha-1} e^{-x/\beta}$. Noten que si $Y_i \sim exp\{\lambda\}$ y $Y_1, \ldots, Y_n$ son independientes, entonces $X = \sum_{i=1}^{n} Y_i \sim \mathcal{G}(n, \lambda)$. Esta distribución se conoce como la *distribución de Erlang*.
Entonces generamos

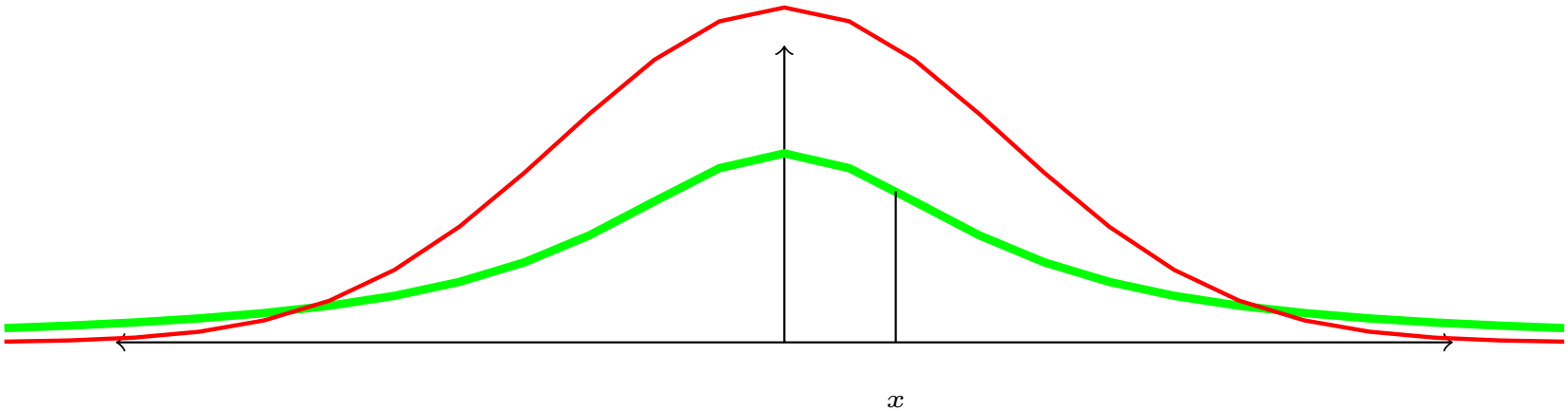
$$X = -\lambda \sum_{i=1}^{n} \log(1 - u_i) = -\lambda \log(\prod_{i=1}^{n} u_i)$$

```
x <- NULL
for(i in 1:1000) x <- append(x,-2*sum(log(runif(10))))
hist(x,prob=T)
curve(dgamma(x,10,1/2),from=0,to=50,add=T,col="red")
```



Ejemplos

d. Distribución Cauchy. X tiene densidad $f(x) = \frac{c}{\pi(1+x^2)}$.



Entonces la función de distribución es de la forma:

$$F(x) = \int_{-\infty}^{x} \frac{c}{\pi(1+v^2)} dv = \frac{2c}{\pi} \int_{0}^{x} \frac{1}{1+v^2} dv = \frac{c}{2} + \frac{c}{\pi} tan^{-1}(x)$$

Entonces podemos tomar $X = F^{-1}(u) = tan(\pi(u - \frac{c}{2}))$.

Ejemplos

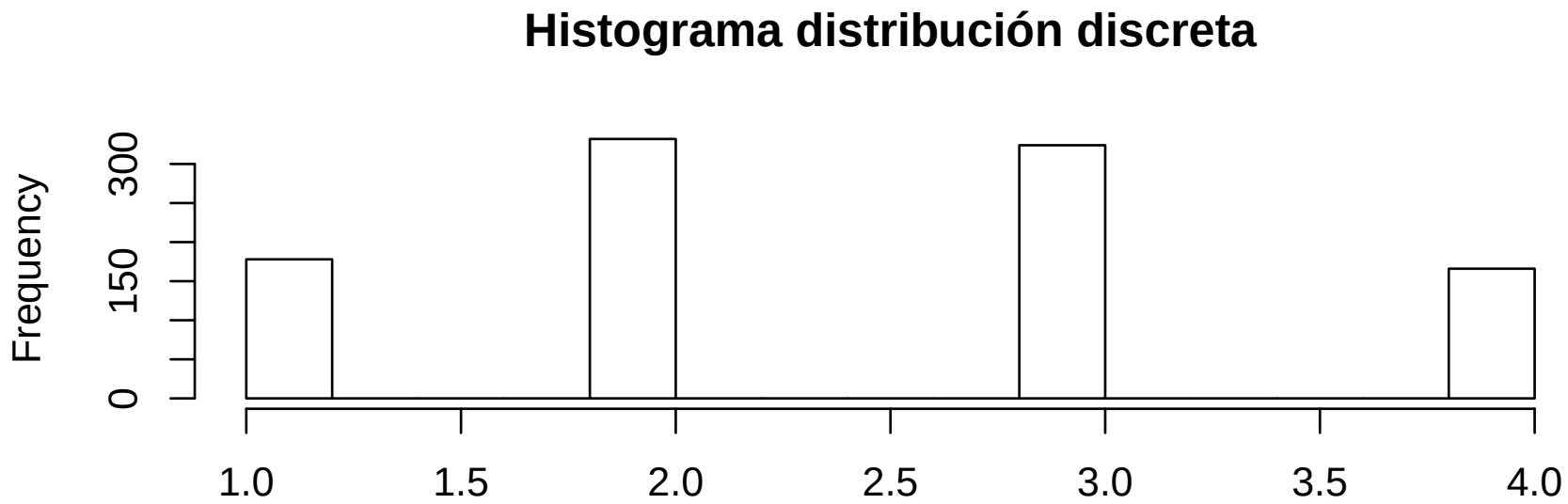
e. X tiene una función de masa de probabilidad dada por:

X	1	2	3	4
$P(X=x)$	$1/6$	$1/3$	$1/3$	$1/6$

Entonces generamos $u \sim \mathcal{U}(0,1)$ si

$$u \in \begin{cases} [0, 1/6] \implies x = 1 \\ (1/6, 1/2] \implies x = 2 \\ (1/2, 5/6] \implies x = 3 \\ (5/6, 1] \implies x = 4 \end{cases}$$

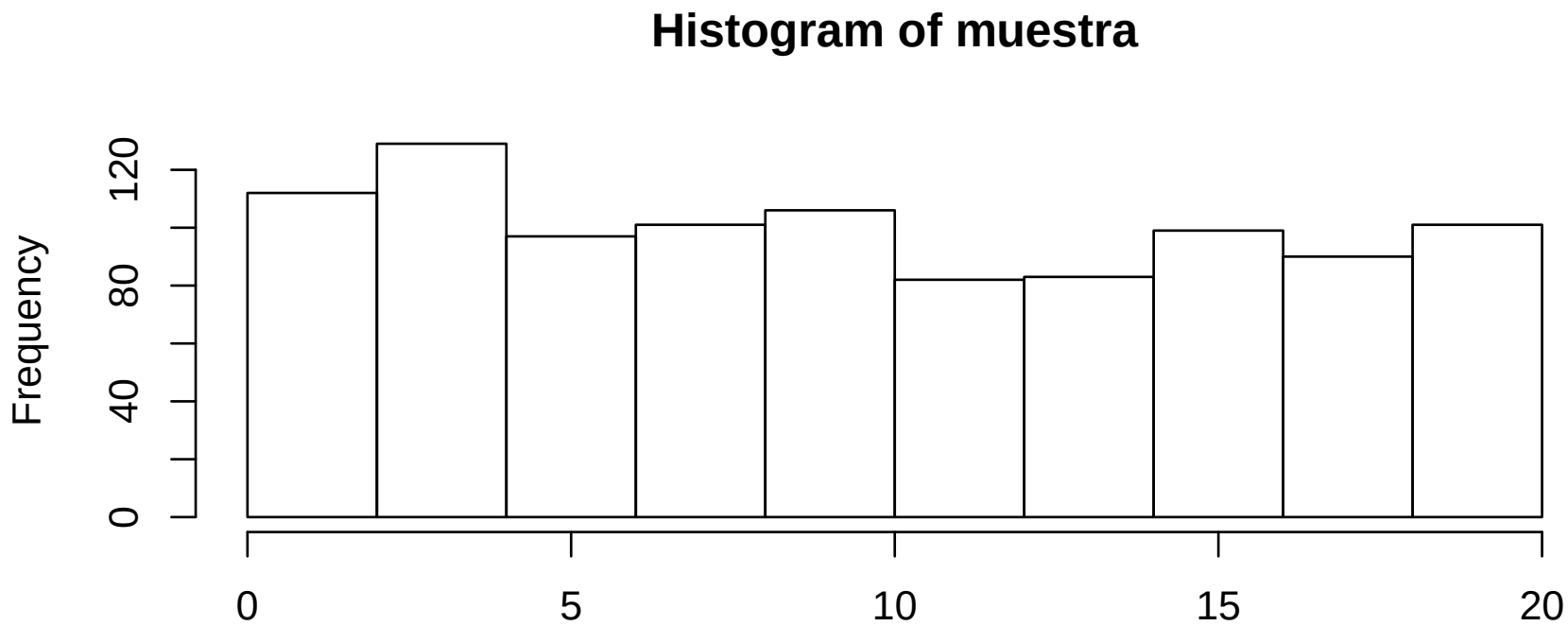
```
hist(sample(1:4,size=1000,replace=T,prob=c(1/6,1/3,1/3,1/6)),main="Histograma distribución discreta")
```



Ejemplos

f. Para una muestra de una uniforme discreta en $\{1, \ldots, n\}$ con $P(Y = k) = \frac{1}{n}$, generamos $U \sim \mathcal{U}(0, 1)$ y define $Y = [Un] + 1$, donde $f(x) = [x]$ es la función parte entera.
En este caso también se puede usar la función `sample` para generar la muestra deseada.

```
n <- 20
muestra <- sample(1:n,1000,replace=T)
head(muestra)

[1] 18 11  9  4 15  2

hist(muestra)
```



Ejemplos

g. Distribución geométrica.

- La función de masa de probabilidad es $f(x) = p(1-p)^x$ con $x = 0, 1, 2, \ldots$.
- La función de distribución es $F(x) = 1 - q^{x+1}$ donde $q = 1 - p$. Entonces, con el método de la transformación inversa necesitamos resolver para x:

$$1 - q^x < u \leq 1 - q^{x+1}$$

- Esta desigualdad se simplifica a $x < \log(1-u)/\log(q) \leq x + 1$. La solución es $x + 1 = \lceil \frac{\log(1-u)}{\log(q)} \rceil$[1]. Por ejemplo, con $p = 1/4$

```
n <- 1000
p <- 0.25
u <- runif(n)
k <- ceiling(log(1-u)/log(1-p))-1 #se puede simplificar a floor(log(u)/log(1-p))
k[1:20]
```

```
 [1] 8 1 2 3 0 3 3 5 1 0 0 9 6 2 2 6 3 0 2 7
```

[1] $\lceil t \rceil$ es la función techo (el menor entero es que mayor o igual a t)

Ejemplos I

g. Distribución Poisson.

- Para la distribución Poisson, la aplicación del método anterior es más complicado, porque no se tiene una fórmula explícita para el valor de x tal que $F(x-1) < u \leq F(x)$
- El método para generar Poisson usa la fórmula recursiva:

$$f(x+1) = \frac{\lambda f(x)}{x+1} \qquad F(x+1) = F(x) + f(x+1)$$

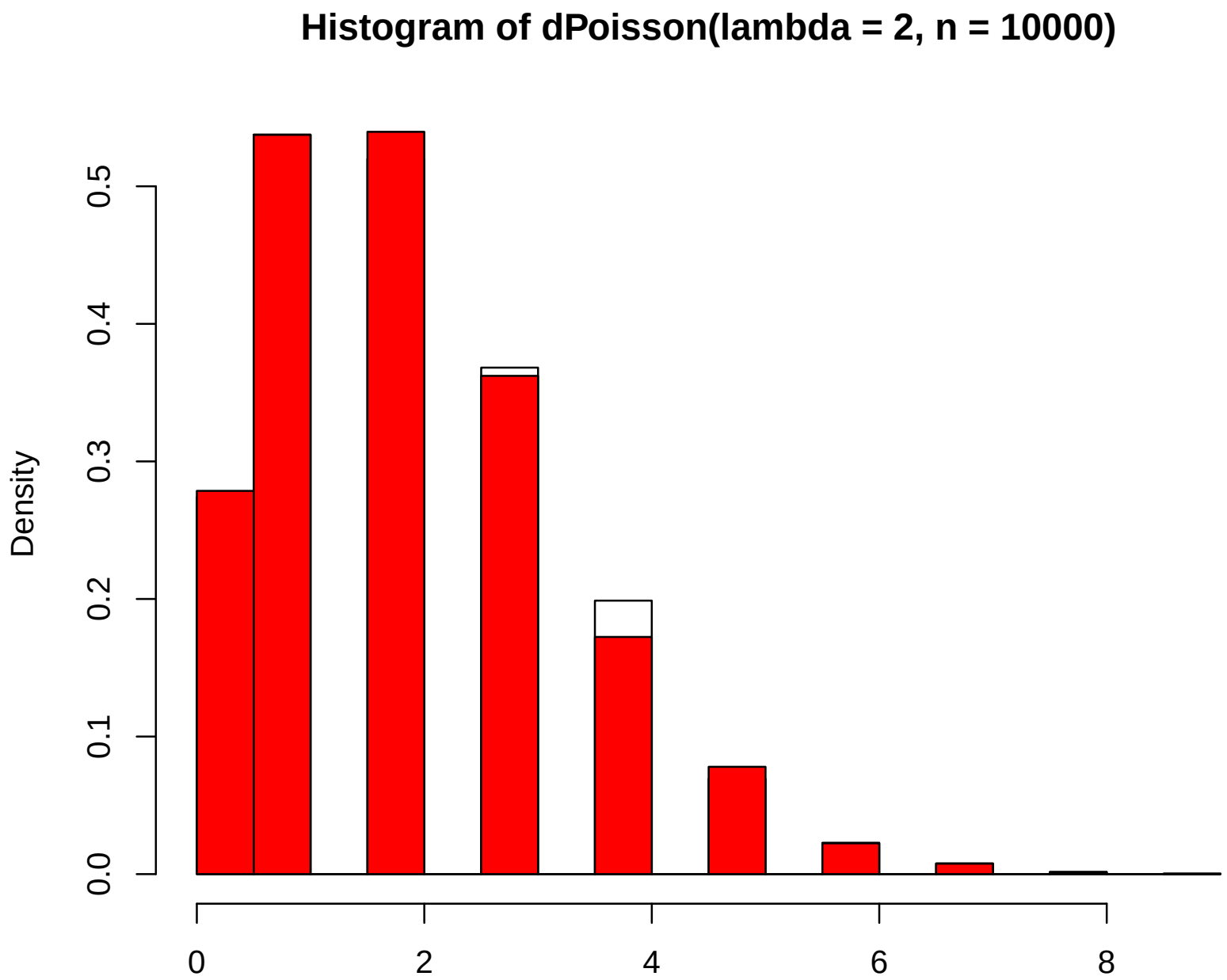
- Entonces, para generar una Poisson, se genera $u \sim \mathcal{U}(0,1)$ y se busca en la tabla (vector) con los valores de $F(x)$ para encontrar el valor de x que es solución de $F(x-1) < u \leq F(x)$.

```
dPoisson <- function(lambda,n){
  #función recursiva para generar muestras de la Poisson(lambda)
  #Primero se requiere generar un vector con los valores de la distribución Poisson
  f <- Fn <- NULL
  f[1] <- exp(-lambda) #densidad en 0
  Fn[1] <- f[1] #distribución en 0
  #General el vector de valores de la distribución
  for (i in 2:(n+1)){
    f[i] <- lambda*f[i-1]/(i-1)
    Fn[i] <- Fn[i-1] + f[i]
  }
  u <- runif(n) # muestra de uniformes
  x <- NULL
  for(i in 1:n){
    x <- append(x,sum(Fn < u[i]))
  }
  return(x)
}
```

Ejemplos II

```
hist(dPoisson(lambda=2,n=10000),prob=T)
hist(rpois(10000,lambda=2),col="red",add=T,prob=T)
```

Ejemplos III



Notas y comentarios

1. No siempre es fácil calcular F^{-1} en forma cerrada (e.g. distribución normal). Se pueden aplicar métodos numéricos.
2. Para las distribuciones discretas: en papel, los métodos son intuitivos, pero computacionalmente hay que usar algunos métodos de búsqueda, evaluación de funciones, etc. que pueden ser lentos. La eficiencia es importante para un buen método.
3. En relación a la eficiencia: cuando hay más de un método, hay que usar el que sea menos costoso (computacionalmente). Por ejemplo,

$$X_1 \sim F_1, X_2 \sim F_2$$

entonces

$$\text{máx}\{X_1, X_2\} \sim F_1(x)F_2(x), \text{mín}\{X_1, X_2\} \sim F_1(x) + F_2(x) - F_1(x)F_2(x)$$

(estadísticas de orden). Ahora consideren la distribución $F(x) = x^2 I_{[0,1]}(x) = F_1(x)F_2(x)$ donde $x \sim \mathcal{U}(0, 1)$.

- con el método de la transformación inversa, $X = \sqrt{u}$.
- con el método del máximo, $X = \text{máx}\{u_1, u_2\}$.
- ¿Cuál es el método más rápido? Depende de la velocidad del generador uniforme, y de las funciones $\sqrt{}$ y $\text{máx}\{\}$. Y en un caso se usa una u y en el otro, dos u's.

Método de composición o mezclas

A veces f o F se pueden escribir como una combinación lineal convexa de dos o más cdf's:

$$F = \sum_{i=1}^{n} p_i F_i, \quad p_i > 0, \sum_{i=1}^{n} p_i = 1$$

Muestreo por composición

Método de muestreo:

1. Muestrear un valor j tal que $P(J = j) = p_j$.
2. Obtener X con distribución F_j.

Demostración.

X tiene distribución:

$$\begin{aligned} F(x) = P(X \leq x) &= \sum_{i=1}^{n} P(X \leq x | J = j) P(J = j) \\ &= \sum_{i=1}^{n} F_j(x) p_j \end{aligned}$$

□

Ejemplos

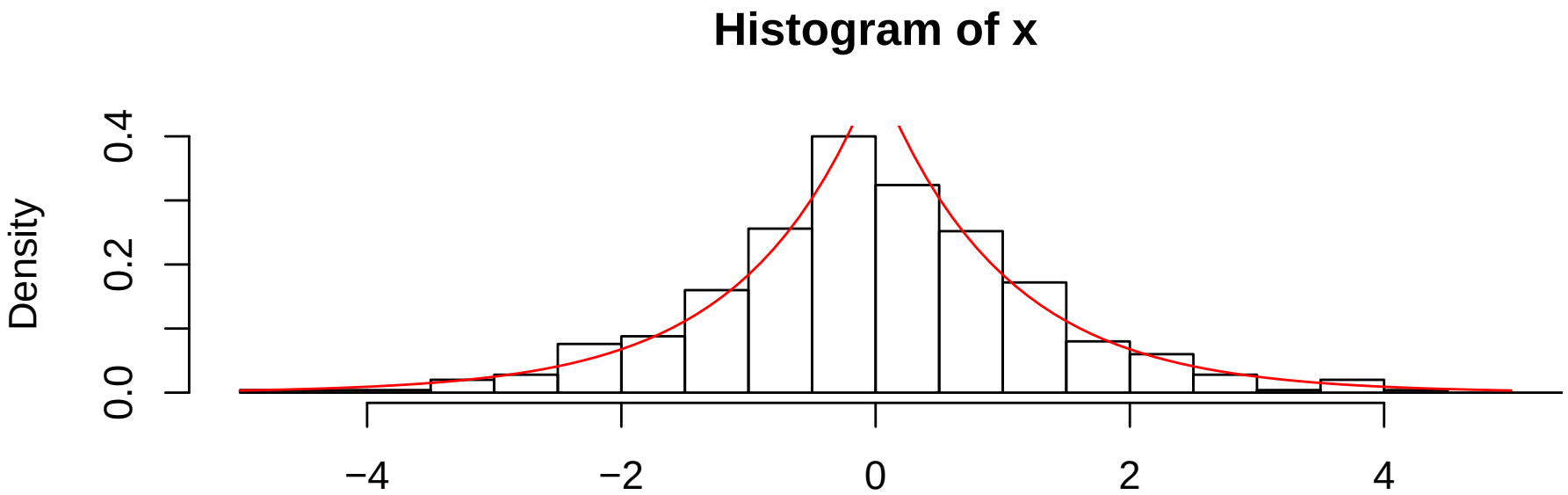
a. Doble exponencial. La densidad es $f(x) = \frac{1}{2}e^{-|x|}$. Noten que f se puede escribir como

$$f(x) = \frac{1}{2}e^{x}I(x \leq 0) + \frac{1}{2}e^{-x}I(x > 0)$$

Entonces el algoritmo es:

- Generar $u_1, u_2 \sim \mathcal{U}(0, 1)$
- Si $u_1 \leq \frac{1}{2}$, hacer $X = log(u_2)$, en otro caso $X = -log(u_2)$.

```
u <- matrix(runif(1000), ncol = 2)
x <- ifelse(u[,1] < 0.5, -1, 1)*log(u[,2])
hist(x, xlim = c(-5,5), prob = T,breaks = 20)
curve(0.5*exp(ifelse(x<0, 1, -1)*x),from = -5, to = 5, add = T, col = "red")
```

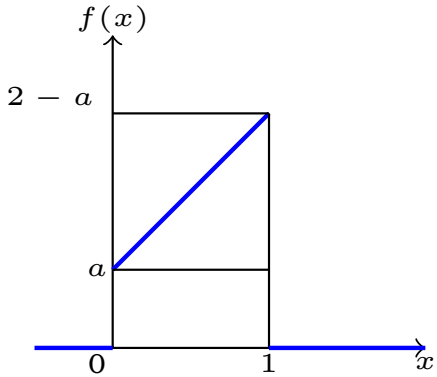


b. Triangular o trapezoidal.

$f(x) = [a + 2(1 - a)x]I_{(0,1)}(x)$ para $0 < a < 1$. f se puede esribir como

$$f(x) = aI_{(0,1)}(x) + (1-a)2xI_{(0,1)}(x) = af_1(x) + (1-a)f_2(x)$$

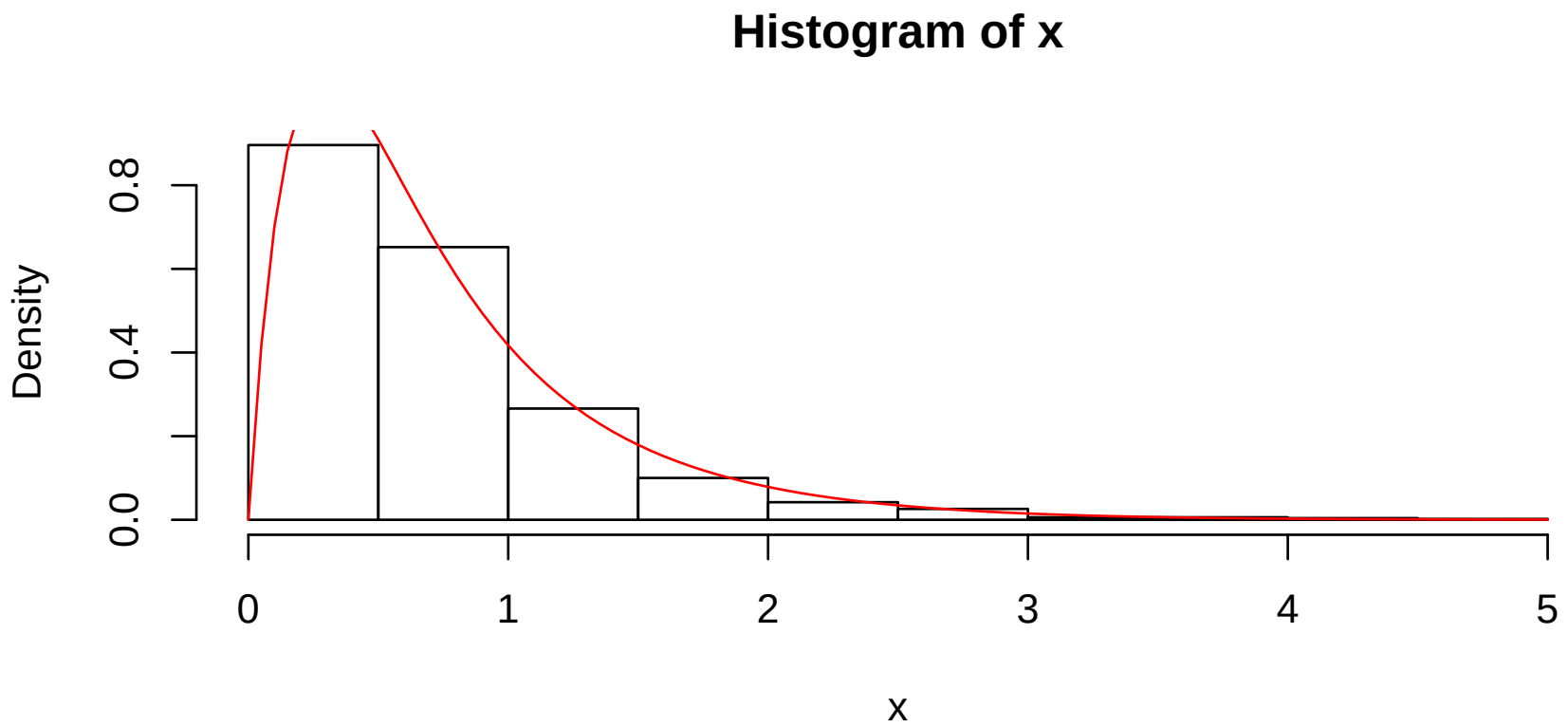
donde $f_1(x)$ es una uniforme, y $f_2(x)$ corresponde a $\sqrt{U}$ o $X = \text{máx}\{U_1, U_2\}$.



Ejemplos

c. Mezcla de distribuciones Gamma. Sean $X_1 \sim \mathcal{G}(2,2)$ y $X_2 \sim \mathcal{G}(2,4)$ independientes. Obtener una muestra de la mezcla $F = 0.5F_{X_1} + 0.5F_{X_2}$.

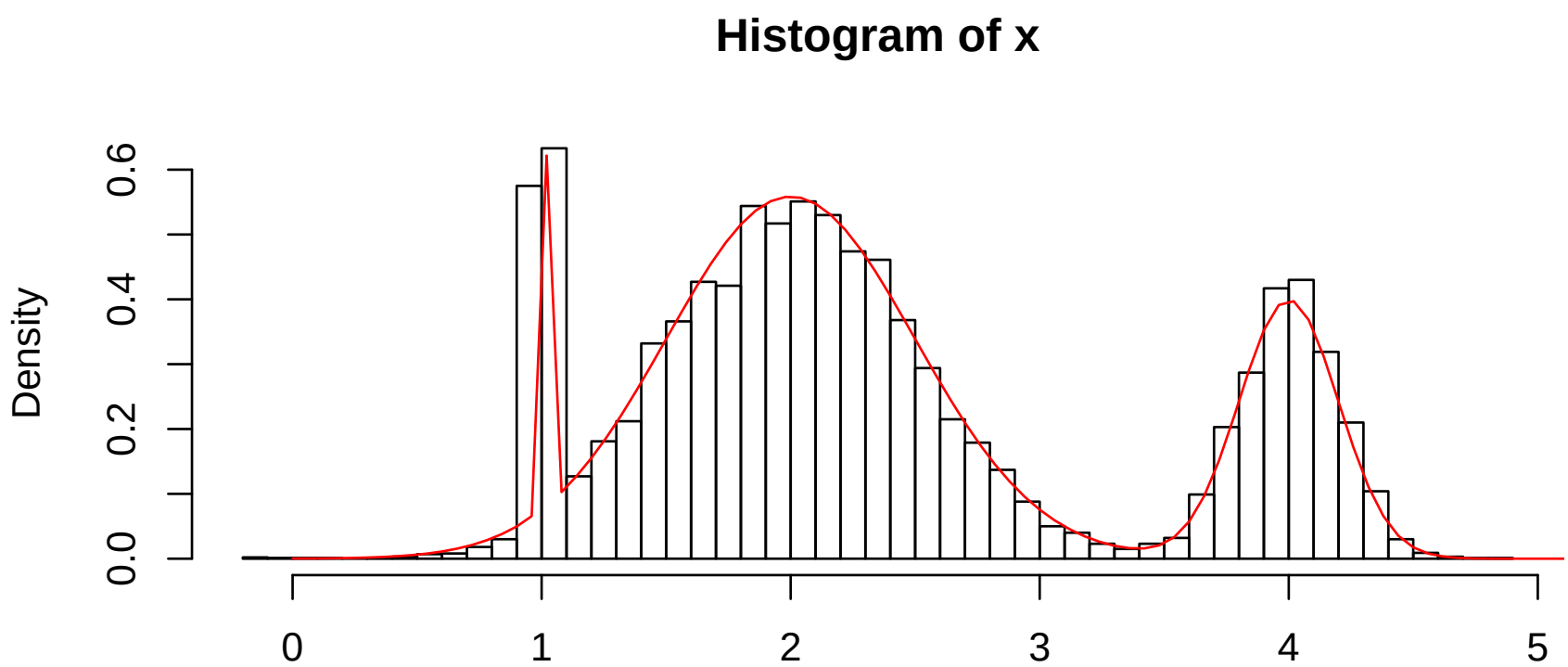
```
n <- 1000
x1 <- rgamma(n,2,2)
x2 <- rgamma(n,2,4)
u <- runif(n)
k <- as.integer(u > 0.5)
x <- k*x1 + (1-k)*x2
hist(x, prob = T)
curve(0.5*dgamma(x,2,2) + 0.5*dgamma(x,2,4), from = 0, to = 5, add = T, col = "red")
```



Ejemplos

d. Mezcla de distribuciones normales. Sean $X_1 \sim \mathcal{N}\left(1, 0.01^2\right)$ y $X_2 \sim \mathcal{N}\left(2, 0.5^2\right)$ y $X_3 \sim \mathcal{N}\left(4, 0.2^2\right)$ independientes. Obtener una muestra de la mezcla $F = 0.1F_{X_1} + 0.7F_{X_2} + 0.2F_{X_3}$.

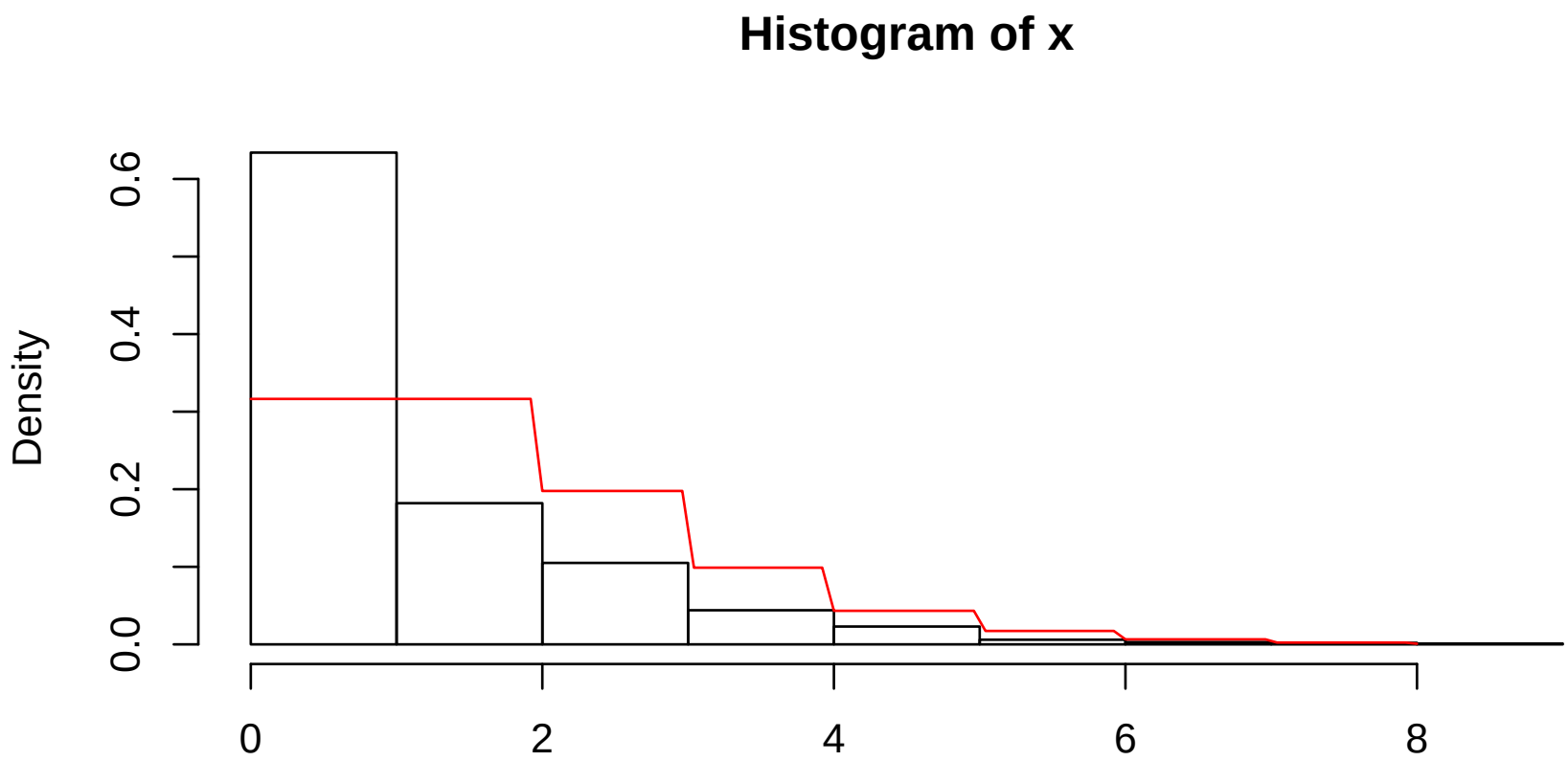
```
n <- 10000
y <- sample(1:3, size = 1000, prob = c(0.1, 0.7, 0.2), replace = T)
x <- rnorm(n, mean = ifelse(y == 1, 1, ifelse(y == 2, 2, 4)),
           sd = ifelse(y == 1, 0.01, ifelse(y == 2, 0.5, 0.2)))
hist(x, prob = T, breaks = 50)
curve(0.1*dnorm(x, 1, sd = 0.01) + 0.7*dnorm(x, 2, sd = 0.5) + 0.2*dnorm(x, 4, sd = 0.2),
           from = -0,to = 6,add = T, col = "red")
```



Ejemplos

e. Mezcla Poisson-Gamma. En este caso, la mezcla es continua. La distribución binomial negativa es una mezcla de distribuciones $x|\lambda \sim \mathcal{P}(\lambda)$, con $\lambda \sim \mathcal{G}(k, \beta)$. Entonces $X \sim Nbin(k, p)$ donde $p = \frac{\beta}{1+\beta}$

```
n <- 1000
k <- 4; beta<- 3
lambda <- rgamma(n,k,beta)
x <- rpois(n,lambda)
hist(x,prob=T)
curve(dnbinom(as.integer(x),k,beta/(1+beta)),from=0,to=8,add=T,col="red")
```



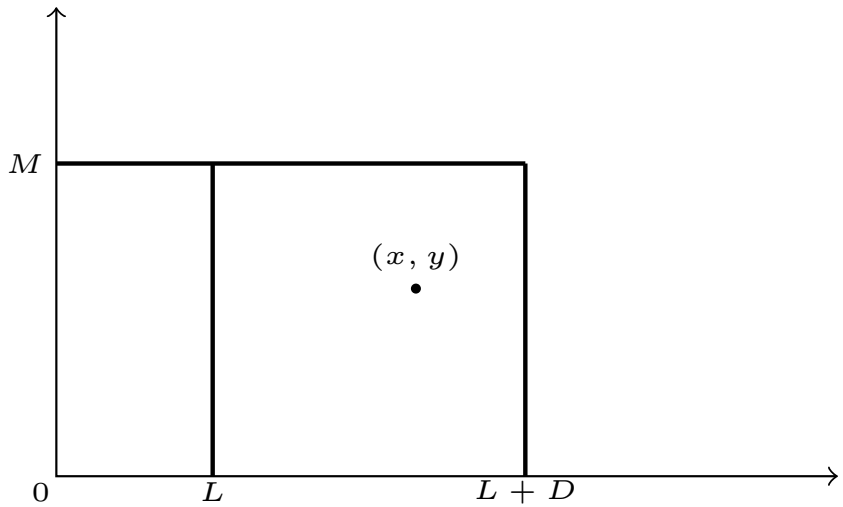
Notas

- Grandes p_j's deben asociarse con métodos rápidos para producir una va $X \sim F_j$. Y conversamente, si se usa un método lento para casos con p_j's pequeñas, no habrá mucha pérdida de eficiencia.
- Las particiones no tienen porqué ser simétricas.
- Los modelos de mezclas, junto con algunos modelos bayesianos jerárquicos y modelos de cadenas de Markov son ejemplos de modelos con estructura jerárquica: hay varios niveles de aleatoriedad y la distribución de probabilidad de variables aleatorias en modelos superiores depende de los valores de variables aleatorias en los niveles inferiores. En estos casos, el proceso de simulación se realiza en pasos, siguiendo la estructura del modelo.

Método de aceptación y rechazo (Von Neumman, 1951)

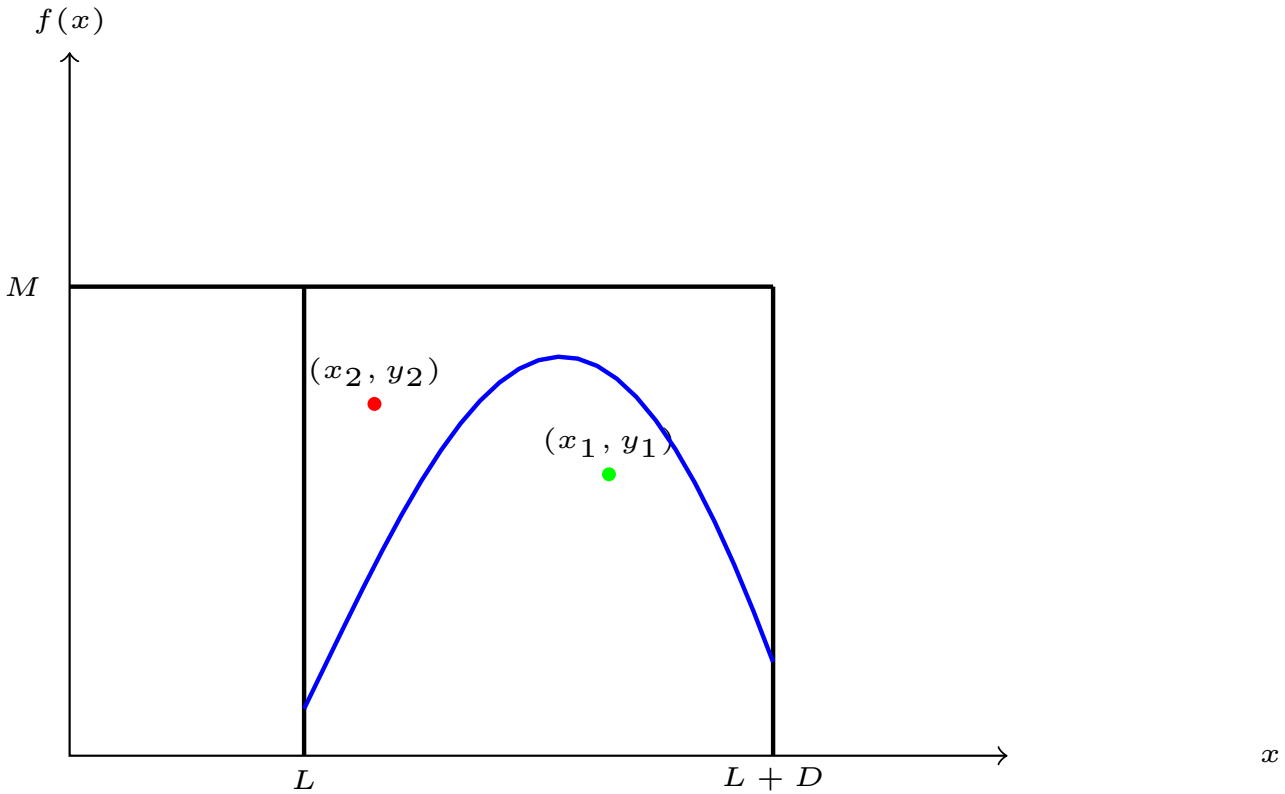
- Supongan que u_1, u_2 son va iid $\mathcal{U}(0,1)$.
- Definan: $X = L + Du_1$, $Y = Mu_2$.
- Entonces (X, Y) es uniforme sobre el rectángulo $[L, L+D] \times [0, M]$:

$$
\begin{aligned}
P(X \leq x, Y \leq y) &= P(X \leq x)P(Y \leq y) \\
&= P(L + DU_1 \leq x)P(MU_2 \leq y) \\
&= P(U_1 \leq \frac{x-L}{D})P(U_2 \leq \frac{y}{M}) \\
&= \frac{x-L}{D}\frac{y}{M}
\end{aligned}
$$



Método de aceptación y rechazo (Von Neumman, 1951)

- Para una v.a. con soporte $[L, L + D]$ y densidad acotada (mayorizada) por M, se puede utilizar una técnica de rechazo como sigue:
  1. Obtener (X, Y) como arriba
  2. Si $Y \leq f(X)$, usa X.
  3. Si $Y > f(X)$, rechaza (X, Y) y regresar a 1.



- La eficiencia del método depende de:
  (a) la facilidad para hacer la comparación
  (b) la proporción de rechazo del método, esto es:

$$P(\text{rechazo}) = 1 - \frac{\text{área de } f(x)}{\text{área del rectángulo}}$$

- Entonces, mientras más se ciña el rectángulo a la función, obtendremos mayor eficiencia.

Método de aceptación-rechazo mejorado

- Supongan que la densidad objetivo a muestrear es f.
- En lugar de usar un rectángulo uniforme $[L, L+D] \times [0, M]$, se puede usar como "envoltura" una distribución g de la cual sea fácil muestrear.
- Podemos suponer que podemos encontrar una constante c tal que $cg(x) \geq f(x) \quad \forall x$. En este caso se dice que g *mayoriza a* f.
- Entonces el algoritmo queda como:
  1. Muestrear $Y \sim g$,
  2. Muestrear $u \sim \mathcal{U}(0, 1)$.
  3. Aceptar $X = Y$ si $ucg(x) \leq f(x) \iff u \leq \frac{f(x)}{cg(x)}$, si no, rechaza Y y repite 1.

Prueba del método de aceptación-rechazo

Demostración.

Sea $A = \left\{\omega | U(\omega) \leq \frac{f(X(\omega))}{cg(X(\omega))}\right\}$ el evento que tiene los casos en donde se acepta la observación. Queremos probar que una variable aleatoria aceptada tiene la distribución objetivo cuando se da el evento A.

$$
\begin{aligned}
P(Y \leq x|A) &= \frac{P(Y \leq x, A)}{P(A)} \\
&= \frac{P(Y \leq x, U \leq \frac{f(Y)}{cg(Y)})}{P(U \leq \frac{f(Y)}{cg(Y)})} \\
&= \frac{\int_{-\infty}^{x} \int_{0}^{f(y)/cg(y)} du\, g(y)\, dy}{\int_{-\infty}^{\infty} \int_{0}^{f(y)/cg(y)} du\, g(y)\, dy} \\
&= \frac{\int_{-\infty}^{x} \frac{f(y)}{cg(y)} g(y)\, dy}{\int_{-\infty}^{\infty} \frac{f(y)}{cg(y)} g(y)\, dy} \\
&= \frac{\int_{-\infty}^{x} f(y)\, dy}{\int_{-\infty}^{\infty} f(y)\, dy} = \int_{-\infty}^{x} f(y)\, dy = P(X \leq x) = F_x(x)
\end{aligned}
$$

□

Método de aceptación-rechazo mejorado

- En este caso, la probabilidad de aceptación es $1/c$.

Demostración.

Considerar la probabilidad de que una iteración produce un valor aceptable:

$$\begin{aligned} P(\text{aceptación}|Y=y) &= P(Y=y, U \leq \frac{f(y)}{cg(y)}) \\ &= P(Y=y)P(U \leq \frac{f(y)}{cg(y)}|Y=y) \\ &= g(y)\frac{f(y)}{cg(y)} = \frac{f(y)}{c} \end{aligned}$$

Integrando sobre los posibles valores de y, obtenemos que $P(\text{aceptación}) = 1/c$. □

- El número de iteraciones necesarias para una aceptación en el algoritmo es una v.a. $W \sim geo(1/c)$. Por lo tanto el número promedio de iteraciones es $c = E(W)$.

Ejemplos

a. Generar una muestra de $x \sim f(x) = 20x(1-x)^3 I_{(0,1)}(x) \equiv \mathcal{B}e\,(2,4)$. Usemos como función mayorizante un rectángulo $g(x) \equiv \mathcal{U}\,(0,1)$. Para escoger la constante c mínima tal que $\frac{f(x)}{g(x)} \leq c$, usemos cálculo.

$$\frac{f(x)}{g(x)} = 20x(1-x)^3 = h(x)$$

$h'(x) = 20(1-x)^3 - 60x(1-x)^2 = 0 \implies x = 1/4$. Entonces
$c = \frac{20}{4}(1-1/4)^3 = 2.109375$ y $\frac{f(x)}{g(x)} \leq 2.109375$. El algoritmo es entonces:

- Genera $u_1, u_2 \sim \mathcal{U}\,(0,1)$.
- si $u_2 \leq 9.48u_1(1-u_1)^3 \implies x = u_1$, si no, regresa.

El número de veces que se realiza el paso 1 es $c = 135/64 \approx 2.11$.

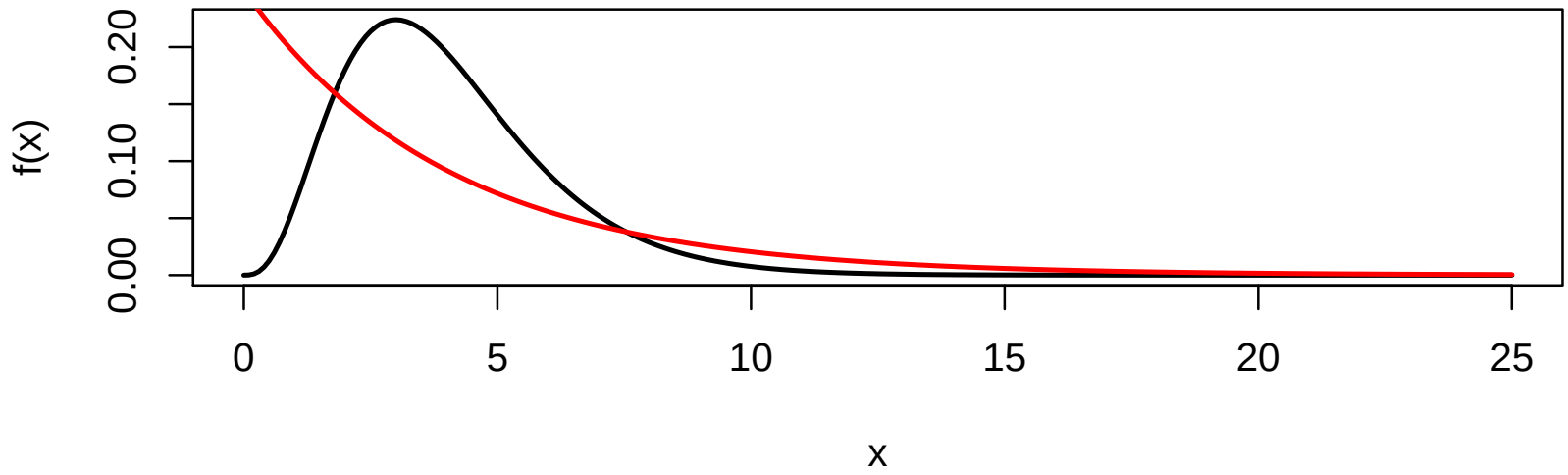
Ejemplos I

b. Envoltura exponencial para una Gamma

- Consideren muestrear de una distribución $f(x) \sim \mathcal{G}(\alpha, 1)$, usando una exponencial como función mayorizante.
- Usamos una $\exp(\alpha)$ para envolver, con densidad $g(y) = \frac{1}{\alpha} e^{-y/\alpha}$. Tenemos que encontrar el valor de c que optimiza la g para cubrir a la f.

```
alfa <- 4
f <- function(x){dgamma(x,alfa,1)}
g <- function(x){dexp(x,1/alfa)}
x <- seq(0,25,0.1) #sucesión de valores de x
plot(x,f(x),type="l",lwd=2)
lines(x,g(x),col="red",lwd=2)
```

Ejemplos II



- Noten que $h(x) = \frac{f(x)}{g(x)} = \alpha\Gamma(\alpha) \times \frac{x^{\alpha-1}e^{-x}}{e^{-x/\alpha}} = \alpha\Gamma(\alpha)x^{\alpha-1}e^{\frac{1-\alpha}{\alpha}x}$ Necesitamos optimizar $h(x)$. Se puede hacer a mano o con la función `optimize` en `R`.

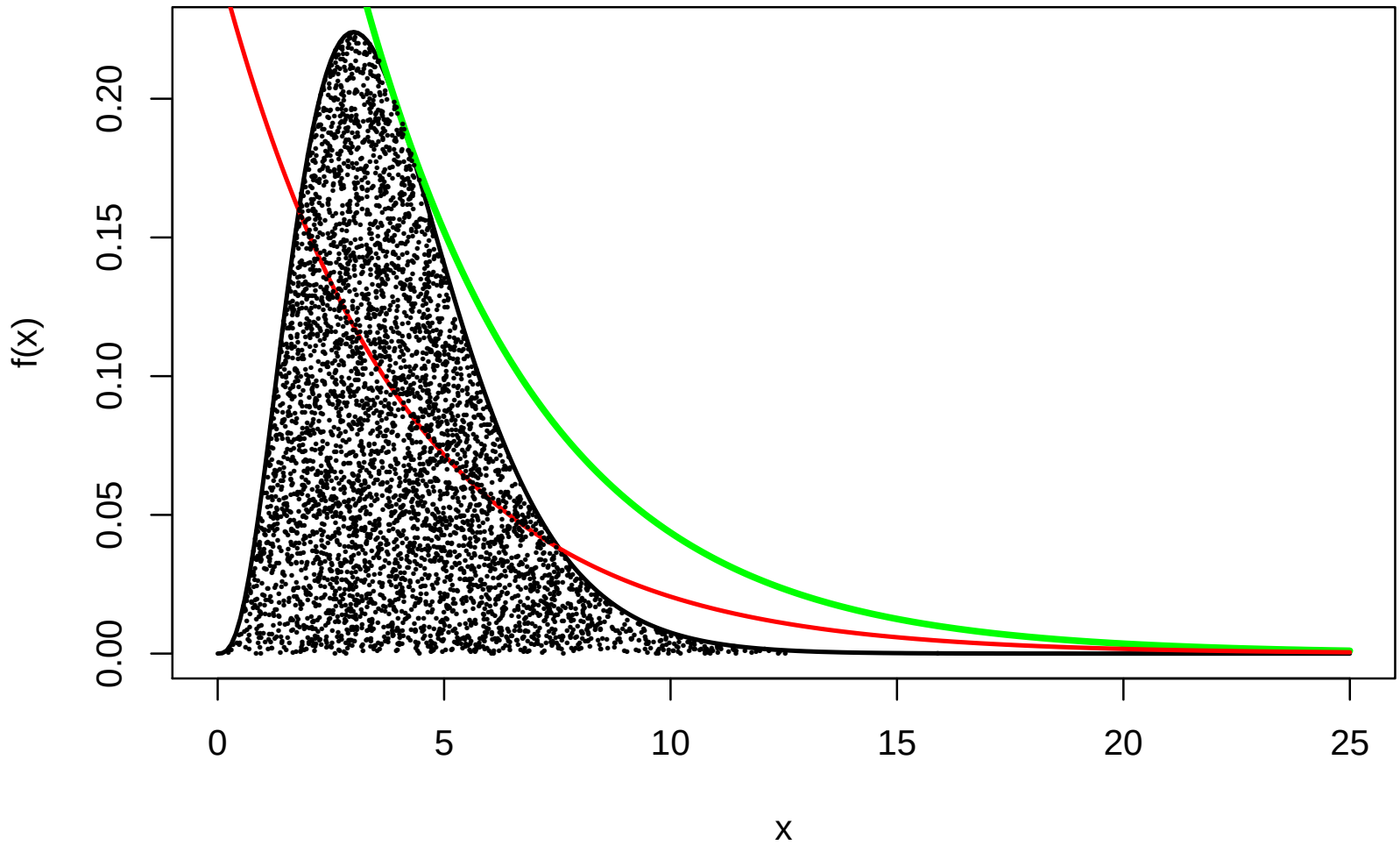
Ejemplos III

```
v <- optimize(f = function(x){f(x)/g(x)}, interval = c(0,25),
              maximum = T)
v

$maximum
[1] 3.999982

$objective
[1] 2.124248

c <- v$objective
```

- La siguiente gráfica muestra los puntos que caen debajo del área relevante, y la proporción del total de puntos generados.

```
n <- 10000 #puntos a generar
plot(x,f(x),type="l",lwd=2)
lines(x, c*g(x), col="green", lwd=3) #exponencial mayorizada
lines(x, g(x), col="red",lwd=2) #exponencial original
u1 <- runif(n) #en principio u1 y u2 son independientes.
u2 <- runif(n)
y <- -alfa*log(1-u2)
indicadora <- u1 <= f(y)/(c*g(y)) #selecciona los puntos que se aceptan
w <- y[indicadora]
points(w,u1[indicadora]*c*g(w),pch=16,ce=0.3)
```

Ejemplos IV



Ejemplos V

```
table(indicadora)/n
```

```
indicadora
 FALSE   TRUE
0.5327 0.4673
```

```
1/c #puntos aceptados
```

```
[1] 0.4707548
```

Ejemplo

c. Muestras de la distribución media-normal. Esta distribución tiene densidad:

$$f(x) = \frac{2}{\sqrt{2\pi}} \exp(-x^2/2) I(x \geq 0)$$

La función cobertura puede ser la distribución exponencial con parámetro $1/\lambda$. Para encontrar la constante c tal que $f(x) \leq cg(x)$, tenemos que

$$\frac{f(x)}{g(x)} = \frac{2}{\sqrt{2\pi}\lambda} e^{-x^2/2+\lambda x}$$

La función cuadrática dentro de la exponencial alcanza su máximo en $x = \lambda$. Entonces tenemos que $f(x)/g(x) \leq c^*$, donde

$$c^* = \frac{2}{\sqrt{2\pi}\lambda} e^{-\lambda^2/2+\lambda^2} = \sqrt{\frac{2}{2\pi\lambda^2}} e^{\lambda^2/2}$$

Método de convolución

- La distribución de probabilidad de una suma de dos o más variables aleatorias $S = X_1 + \cdots + X_n$ se llama una *convolución* de las distribuciones de las variables originales y se denota por:

$$f_1 * f_2 * \cdots * f_n$$

- El método de convolución se refiere a sumar variables aleatorias para obtener una nueva variable aleatoria con la distribución deseada.
- Lo que es importante entonces no es la función de distribución de la variable deseada, sino su relación con otras variables que se generan con mayor facilidad.

Ejemplos I

1. Distribución de Erlang. Esta distribución es un caso particular de la distribución $Gamma(\alpha, \beta)$, en donde el parámetro α es natural. Hemos visto que una suma de exponenciales es Erlang:

$$\sum_{i=1}^{n} Y_i \sim \mathcal{G}(n, \beta)$$

   donde $Y_i \sim exp(\beta)$. Entonces la distribución de Erlang es una convolución.
2. Distribución Binomial. Sabemos que la suma de n Bernoullis con parámetro p independientes, son una binomial con parámetros n y p.
3. Si $Y_i \sim \chi^2_{(1)}$ y $Y_1, \ldots, Y_n$ son iid, entonces $\sum_{i=1}^{n} Y_i \sim \chi^2_{(n)}$.
4. $X_1, \ldots, X_n \sim \mathcal{G}(\alpha_i, \beta)$ e independientes $\implies \sum_{i=1}^{n} X_i \sim \mathcal{G}\left(\sum_{i=1}^{n} \alpha_i, \beta\right)$
5. Suma de normales es normal.
6. La distribución Binomial Negativa NegBin(k,p) es la convolución de k iid geométricas con parámetro p.

Distribuciones truncadas

- Si F es una función de distribución con densidad f, la densidad truncada en (a, b) es:

$$f^*(x) = \frac{f(x)}{F(b) - F(a)} I_{(a,b)}(x)$$

con distribución truncada

$$F^*(x) = \frac{F(x) - F(a)}{F(b) - F(a)} I_{(a,b)}(x) + I_{\{x \geq b\}}(x)$$

Entonces para generar una muestra de F^*, tenemos que recorrer la muestra de números uniformes al intervalo de interés:

1. Genera $u \sim \mathcal{U}(0, 1)$.
2. Define $\nu = F(a) + (F(b) - F(a))u$. Entonces $\nu \sim \mathcal{U}(F(a), F(b))$.
3. Define $x = F^{-1}(\nu)$.

Generación de estadísticas de orden I

Para generar una estadística de orden $X_{(i)}$ de una distribución F , podemos usar propiedades de la distribución beta:

Algoritmo generación de estadísticas de orden

Sea $X_1, \ldots, X_n \sim F$ independientes.

- Si n es pequeña, hacer $Y_i = X_{(i)}$ (ordena directamente la muestra).
- Si n es grande,
  1. genera $\nu \sim \mathcal{B}e\,(i, n-i+1)$
  2. define $X = F^{-1}(\nu)$

Recordar que si $X \sim F$ entonces la $i-$ésima estadística de orden tiene la distribución dada por:

$$F^*(x_{(i)}) = \sum_{k=i}^{n} \binom{n}{k} F(x_{(i)})^k [1 - F(x_{(i)})]^{n-k}$$

Para la variable aleatoria X generada con el algoritmo indicado tenemos que:

$$\begin{aligned} F(x) &= P(X \leq x) = P(F^{-1}(\nu) \leq x) \\ &= P(\nu \leq F(x)) \\ &= \int_0^{F(x)} \frac{1}{B(i, n+i-1)} \nu^{i-1}(1-\nu)^{n-i+1-1} d\nu = * \end{aligned}$$

Noten que

$$B(i, n-i+1) = \frac{\Gamma(i)\Gamma(n-i+1)}{\Gamma(i+(n-i+1))} = \frac{\Gamma(i)\Gamma(n-i+1)}{\Gamma(n+1)} = \frac{(i-1)!(n-i)!}{n!}.$$

Entonces continuando con el desarrollo e integrando por partes:

Generación de estadísticas de orden III

$$
\begin{aligned}
* &= \int_0^{F(x)} \frac{n!}{(i-1)!(n-i)!}\nu^{i-1}(1-\nu)^{n-i}d\nu \\
&= \int_0^{F(x)} \binom{n}{i} i\nu^{i-1}(1-\nu)^{n-i}d\nu \\
&= \binom{n}{i}\int_0^{F(x)} i\nu^{i-1}(1-\nu)^{n-i}d\nu \\
&= \binom{n}{i}\left[F(x)^i(1-F(x))^{n-i} + \int_0^{F(x)} \nu^i(n-i)(1-\nu)^{n-i-1}d\nu\right] \\
&= \cdots \text{continuar usando la recurrencia.} \\
&= \sum_{k=i}^{n}\binom{n}{k}F(x)^k[1-F(x)]^{n-k}
\end{aligned}
$$

Métodos particulares

Distribución Normal

- Probablemente, una de las más usadas (y abusadas) distribuciones de probabilidad es la distribución Normal con media μ y varianza σ^2:

$$\Phi_{(\mu,\sigma^2)}(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(t-\mu)^2}{2\sigma^2}} dt$$

- Una variable normal estándar $Z \sim \mathcal{N}(0,1)$, se puede transformar fácilmente: $X = \mu + \sigma Z$. Por eso nos podemos concentrar en generar normales estándar.

Métodos basados en el Teorema del Límite Central

Un método histórico para generar distribuciones normales está basado en el Teorema del Límite Central (TLC). Sea $U_1, U_2, \ldots, U_k$ una muestra aleatoria uniforme estándar. En este caso, $E(U_i) = 1/2$ y $\text{Var}(U_i) = 1/12$. Entonces por el TLC:

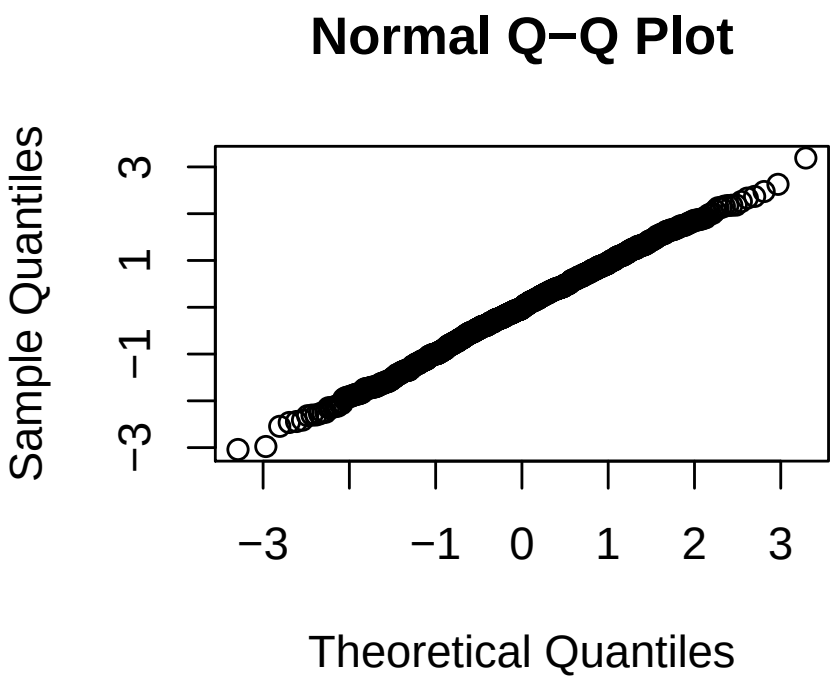
$$Z = \frac{\sum_{i=1}^{k} U_j - k/2}{\sqrt{k/12}} \sim \mathcal{N}(0, 1)$$

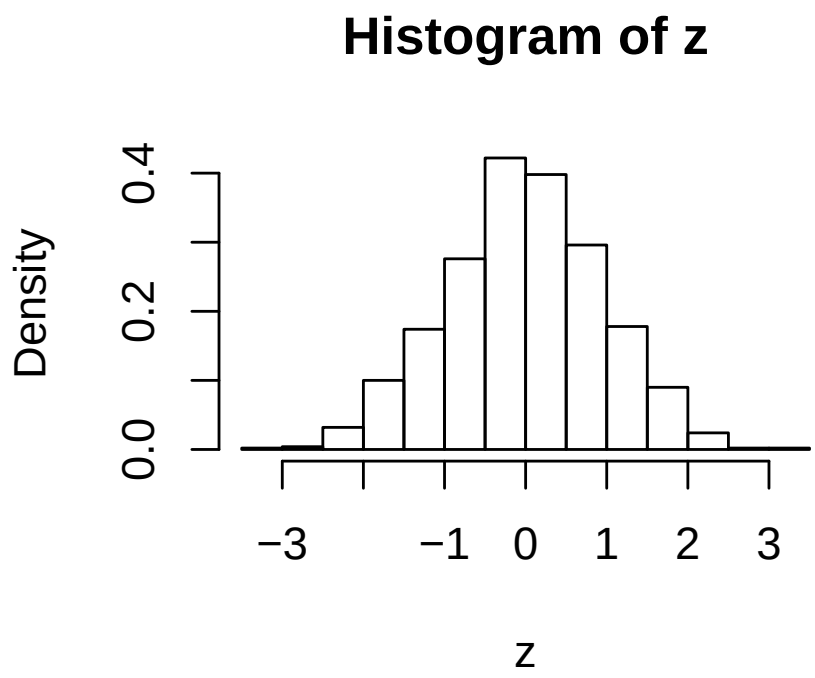
Si $k = 12$ la ecuación se simplifica. ¿Qué tan bueno es el método? Esto depende de k, sin embargo, no importa que tan grande sea k, el método sólo es aproximado.

Ejemplo

A continuación generamos una muestra de $n = 1000$ números aleatorios con el método propuesto. ¿Detectan algún problema?

```
n <- 1000; k <-12; a <- k*n
u <- runif(a)
z <- NULL
for(i in 1:n) z[i] <- (sum(u[k*(i-1)+1:k])-6)
par(mfrow=c(1,2))
qqnorm(z)
hist(z,breaks=20,prob=T)
```

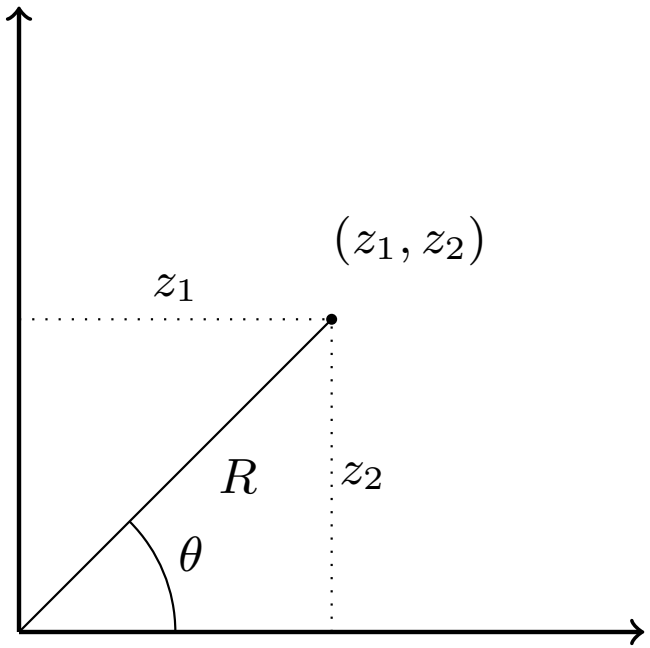




Método de Box-Müller (1958)

El método se basa en la transformación a coordenadas polares

$$(u_1, u_2) \to (z_1, z_2)$$



Supongan que $z_1 \perp\!\!\!\perp z_2$. Entonces

- $z_1^2 + z_2^2 \sim \chi^2_{(2)} \equiv \mathcal{G}\,(2/2, 2) \equiv \exp(2)$
- La distancia
  $d((z_1, z_2), (0, 0)) = ||(z_1, z_2)|| = \sqrt{z_1^2 + z_2^2} = R$
- $\theta \sim \mathcal{U}\,(0, 2\pi)$ por la simetría de los contornos de la densidad bivariada.

En términos de z_1 y z_2:

- $sen(\theta) = c.o./h = z_2/R$ y
- $cos(\theta) = c.a./h = z_1/R$ Entonces
- $tan\theta = z_2/z_1 \implies \theta = tan^{-1}(z_2/z_1)$.

Usando u_1 para generar R y u_2 para generar θ, $R = \sqrt{-2\log u_1}$ (ya que $R^2 \sim \exp(2)$) y $\theta = 2\pi u_2$. Por lo tanto

Transformación de Box-Müller

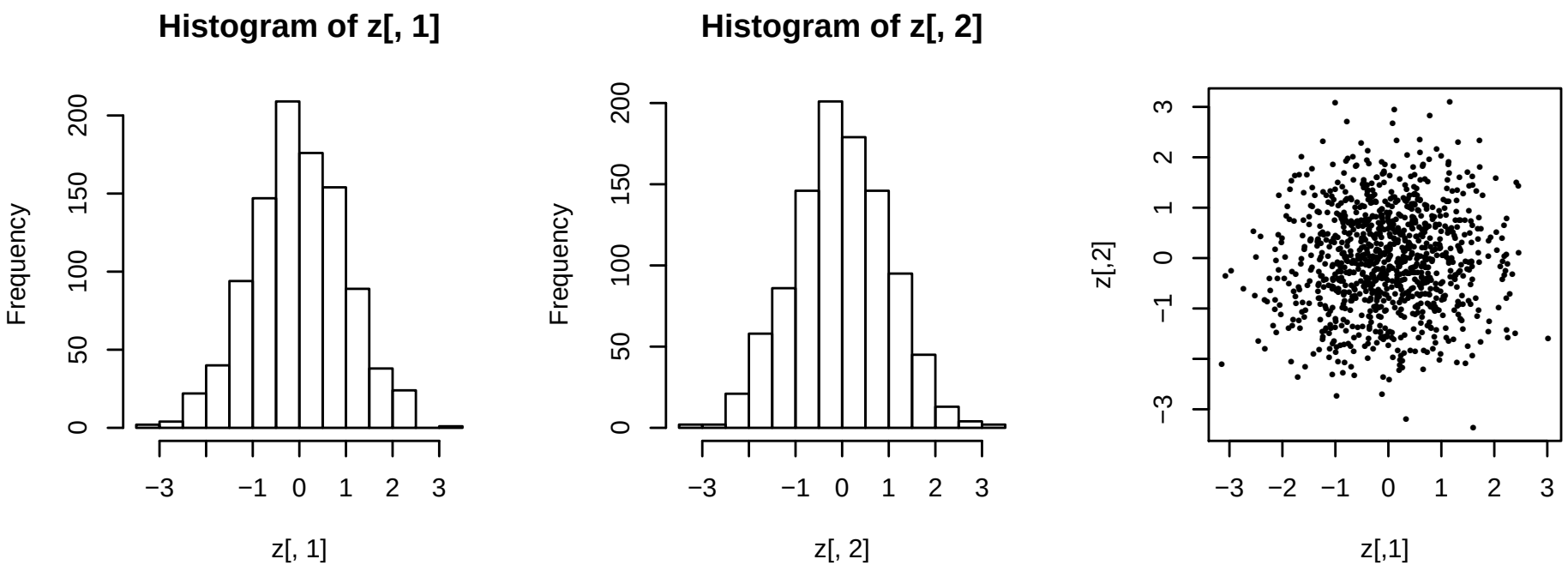
Si $U_1 \perp\!\!\!\perp U_2 \sim \mathcal{U}\,(0, 1)$

$$\begin{aligned} Z_1 &= \sqrt{-2\log u_1}\cos(2\pi u_2) \\ Z_2 &= \sqrt{-2\log u_1}\sin(2\pi u_2) \end{aligned}$$

Entonces $Z_1 \perp\!\!\!\perp Z_2 \sim \mathcal{N}\,(0, 1)$.

Ejemplo

```
seed <- 10; n <- 1000
BM <- function(u){ #u es un par de uniformes
        R <- sqrt(-2*log(u[1])); th <- u[2]
        z1 <- R*cos(2*pi*th); z2 <- R*sin(2*pi*th)
        return(c(z1,z2))
}
z <- matrix(0,nrow=n,ncol=2)
for(i in 1:n) z[i,] <- BM(c(runif(1),runif(1)))
par(mfrow=c(1,3)); par(pty="s")
hist(z[,1],breaks=20); hist(z[,2],breaks=20);plot(z,pch=16,cex=0.5)
```

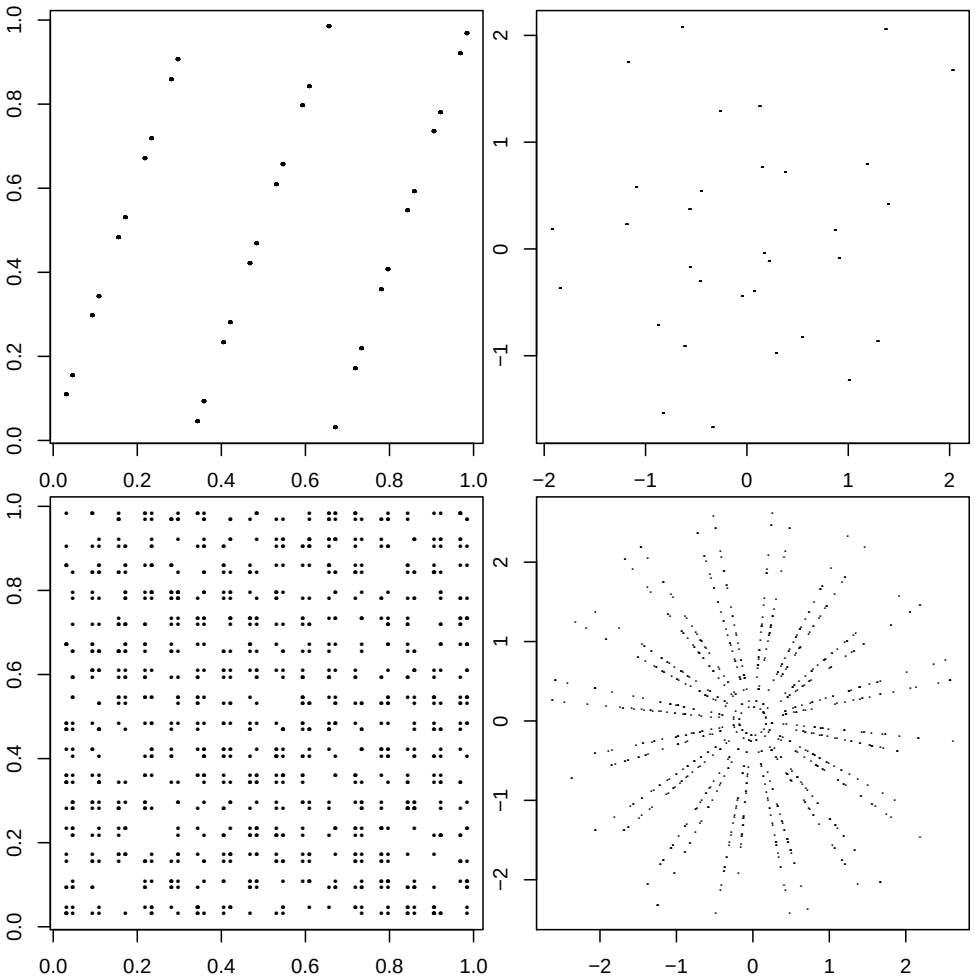


Comentarios al Método de Box-Müller

- El método de Box y Müller es muy costoso computacionalmente. Requiere el cálculo de varias funciones complejas: $\sqrt{}, \log, \sin, \cos$. Sin embargo, es muy fácil de implementar en cualquier computadora, usando C, Fortran, Java, Python, R, etc.
- La calidad de los normales dependerán de los números uniformes que se utilicen. En el siguiente ejemplo, usaremos un generador congruencial malo (RANDU).
- Algunos problemas se pueden resolver permutando los números uniformes antes de usarlos.

Ejemplo

```
seed <- 10
RANDU <- function() {seed <<-(3 * seed+1) %% (64); seed/(64)} #generador RANDU
u <- NULL; n <- 1000
for(i in 1:n) u <- c(u,RANDU()) #genera muestra de n uniformes RANDU
R <- sqrt(-2*log(u[1:(n-1)]))  #Ahora obten normales con Box-Muller
th <- u[2:n]
z1 <- R*cos(2*pi*th); z2 <- R*sin(2*pi*th)
par(mfrow=c(2,2), oma = c(1,1,1,1) + 0.1, mar=c(1,1,1,1) + 0.1)
plot(u[1:(n-1)],u[2:n],pch=16,cex=0.4)
plot(z1,z2,pch=16,cex=0.2)
#Ahora permutamos para mejorar la cobertura.
v <- sample(u); R <- sqrt(-2*log(v[1:(n-1)])); th <- v[2:n]
plot(v[1:(n-1)],v[2:n],pch=16,cex=0.4)
plot(R*cos(2*pi*th), R*sin(2*pi*th), pch=16, cex=0.2)
```



Método Polar (Marsaglia, 1964)

El método polar se basa en la idea original de Box-Müller, pero elimina el cálculo de las funciones trigonométricas, generando un método más eficiente.

Método Polar

1. Muestrea $u_1 \perp\!\!\!\perp u_2 \sim \mathcal{U}(0,1)$.
2. Definir: $V_1 = 2u_1 - 1$ y $V_2 = 2u_2 - 1$. Entonces $V_i \sim \mathcal{U}(-1,1)$.
3. Si $S = V_1^2 + V_2^2 > 1$ se rechaza el punto. Regresa a (1). Si $V_1^2 + V_2^2 \leq 1$ transforma:

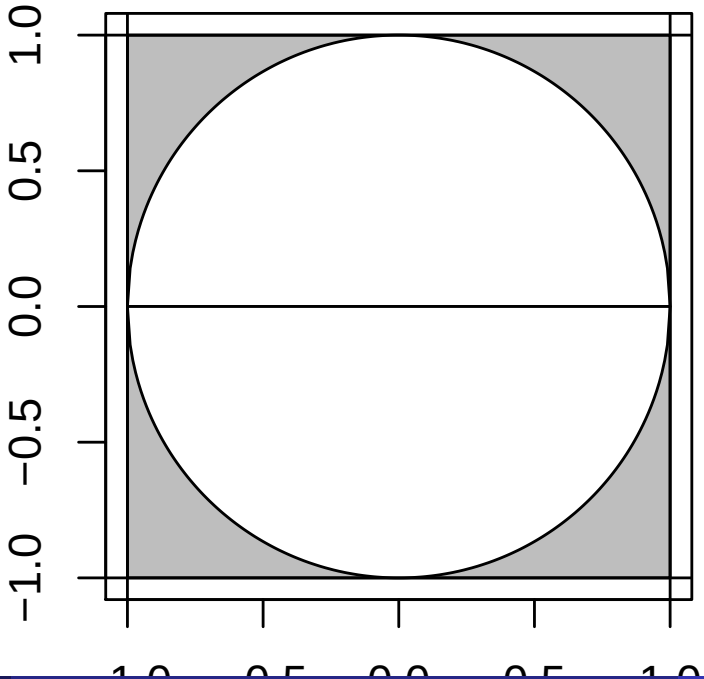
$$Z_1 = V1\sqrt{\frac{-2\log S}{S}}, Z_2 = V_2\sqrt{\frac{-2\log S}{S}}$$

Demostración.

Problema de tarea. Hint: $h(z_1, z_2) = f(v_1, v_2)|J|$. □

Comentarios al Método Polar

- El tiempo ahorrado reemplazando funciones trigonométricas es importante comparado con el uso de aceptación y rechazo.
- Obtenemos nuevas uniformes sólo si caemos en la región sombrada, que tiene área $\frac{4-\pi}{4} = 1 - \pi/4 = 0.215$. Entonces se rechaza 21.5 % del tiempo.
- Entonces se requieren en promedio (basados en la distribución geométrica) $4/\pi = 1.27$ uniformes por normal: $X =$ número de rechazos antes de aceptar. Entonces $X \sim geo(\pi/4)$ y por lo tanto $E(X) = 1/(\pi/4) = 4/\pi \approx 1.27$.



Distribución Beta

- La variable aleatoria $X \sim \mathcal{B}e\,(\alpha, \beta)$ tiene función de densidad de probabilidad:

$$f(x) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1}, \quad \alpha, \beta > 0, x \in [0,1]$$

- Si partimos de que $Y_1 \perp\!\!\!\perp Y_2$, $Y_1 \sim \mathcal{G}\,(\alpha, 1)$, $Y_2 \sim \mathcal{G}\,(\beta, 1)$, respectivamente, entonces

$$X \sim \frac{Y_1}{Y_1 + Y_2} \sim \mathcal{B}e\,(\alpha, \beta)$$

- Otro enfoque es cuando α y β son enteros, entonces podemos usar estadísticas de orden:

Generando $\mathcal{B}e\,(m, n)$

1. Genera $m + n - 1$ uniformes independientes $u_1, u_2, \ldots, u_{n+m-1}$.
2. Calcular $X = u_{(m)}$. Entonces $X \sim \mathcal{B}e\,(m, n)$

- El número total de comparaciones necesarias para encontrar $u_{(m)}$ es igual a $(m/2)(m + 2n - 1)$, por lo que el procedimiento no es eficiente para m y n grandes.

Distribución Poisson

- Una variable Poisson N con media $\lambda > 0$ tiene función de masa de probabilidad:

$$P(N = n) = \frac{e^{-\lambda}\lambda^n}{n!} \quad n = 0, 1, \ldots$$

- N se puede interpretar como el número de llegadas de un proceso Poisson en una unidad de tiempo.
- Los tiempos de interarribo en un proceso Poisson siguen una distribución exponencial con tasa λ. Entonces hay una relacion entre los tiempos de interarribo y la distribución Poisson:

$$N = n \iff \sum_{i=1}^{n} T_i \leq 1 < \sum_{i=1}^{n+1} T_i$$

Entonces podemos usar la exponencial para generar variables aleatorias Poisson.

Distribución Poisson I

Variable aleatoria Poisson

1. Definir $n = 0$, $P = 1$
2. Generar $u_{n+1} \sim \mathcal{U}(0, 1)$ y definir $P \leftarrow P u_{n+1}$
3. Si $P < e^{-\lambda}$, acepta $N = n$. Si no, rechaza n, incrementa a $n + 1$ y regresa a 2.

Demostración.

Si $T_i = -(1/\lambda)log(u_i)$, por la relación mencionada

$$\sum_{i=1}^{n} \frac{-\log u_i}{\lambda} \leq 1 < \sum_{i=1}^{n+1} \frac{-\log u_i}{\lambda}$$

Multiplicando por $-\lambda$ y agrupando suma de logaritmos:

$$\log \prod_{i=1}^{n} u_i \geq -\lambda > \log \prod_{i=1}^{n+1} u_i$$

Por último, exponenciando ambos lados, obtenemos el resultado, notando que P es el producto acumulado requerido.

$$\prod_{i=1}^{n} u_i \geq e^{-\lambda} > \prod_{i=1}^{n+1} u_i$$

□

Eficiencia del procedimiento Poisson

- ¿Cuántos números aleatorios, en promedio serán requeridos para generar una variable Poisson?
- Si $N = n$, entonces se requieren $n + 1$ números, así que el número promedio está dado por

$$E(N + 1) = \lambda + 1$$

que es muy grande si la media de la Poisson es muy grande.

- Cuando λ es grande (digamos mayor a 15), podemos usar una aproximación normal que trabaja bien.

$$Z = \frac{N - \lambda}{\sqrt{\lambda}} \overset{a}{\sim} \mathcal{N}(0, 1)$$

Entonces aproximadamente:

$$N \overset{a}{=} \ulcorner \lambda + \sqrt{\lambda} Z \urcorner$$

donde $\ulcorner y \urcorner$ es la función que redondea y al entero más cercano.

Ejemplo Poisson

- Genera una muestra de 100 variables Poisson con media $\lambda = 3$.

  Solución.

```
N <- NULL; k <- 100
for(i in 1:k){
        n <- 0; P <- runif(1)
        while(P > exp(-3)){u <- runif(1); P <- P*u;  n <- n+1;}
        N[i] <- n
}
N
```

```
  [1]  4  1  4  4  5  2  3  1  2  2  4  4  4  1  2  3  1  3  4  2  2  4  4
 [24]  1  4  3  3  2  3  2  3  3  5  1  4  4  4  2  1  3  3  3  3  3  5  3
 [47]  3  2  3 11  3  1  1  1  5  2  2  6  2  4  1  3  5  3  5  3  4  4  2
 [70]  3  5  2  2  4  3  1  6  2  4  2  3  5  0  4  3  3  3  4  1  4  3  3
 [93]  4  1  1  2  3  6  3  4
```

□

- Genera 100 Poisson, con media $\lambda = 20$.

  Solución.

```
z <- rnorm(100) #haciendo trampa, tomando normales
N <- ceiling(20 + sqrt(20)*z)
N
```

```
  [1] 20 21  7 22 26 20 10 19 24 21 16 28 15 16 17 13 22 20 17 24 17 11 16
 [24] 23 29 24 26 17 17 20 16 22 19 20 24 21 20 21 22 23 17 21 18 21 10 16
 [47] 26 21 17 18 23 22 26 10 17 21 19 14 30 16 26 24 24 21 23 25 24 25 20
 [70] 22 17 16 22 28 24 21 18 21 18 20 21 15 16 13 20 22 23 27 18 19 15 21
 [93] 23 26 23 12 16 15 15 16
```

□

Algoritmo para distribución multinomial

La distrbución multinomial es la del vector de dimensión k, $X = (X_1, \ldots, X_k)$ donde cada componente corresponde a una de k categorías, y se realizan n ensayos. La distribución multinomial tiene la siguiente función de masa de probabilidad:

$$P(X_1 = x_1, X_2 = x_2, \ldots, X_k = x_k) = \binom{n}{x_1, \ldots, x_k} \prod_{i=1}^{k} p_i^{x_i}$$

Como se puede ver, es una generalización de una variable aleatoria binomial.

Algoritmo para multinomial

1. Define $r_j = \sum_{l=1}^{j} p_l$, para obtener $(r_1 = p_1, r_2, \ldots, r_k = 1)$
2. Genera $u \sim \mathcal{U}(0, 1)$
3. Compara u con $r_1, r_2, \ldots$ hasta que $u > r_j$ y $u \leq r_{j+1}$
4. Entonces seleccionamos la $j+1$-ésima celda de la multinomial. Define $X = \epsilon_k$, donde ϵ_k es un vector un 1 en la entrada k y 0 en el resto.
5. La suma de los vectores $X = \sum_{i=1}^{n} \epsilon_i$ tiene la distribución deseada

Ejemplo multinomial, $n = 10, k = 4$

Consideremos una distribución multinomial con $p = (0.2, 0.1, 0.3, 0.4), n = 10$, así que $k = 4$. Este ejercicio genera una muestra de tamaño 20 de un vector multinomial

```
p <- c(0.2,0.1,0.3,0.4)
k <- length(p)
n <- 10
r <- cumsum(p)
X <- matrix(0,nrow=20,ncol=4)
for(i in 1:20){
u <- runif(10)
eps <- findInterval(u,r) #esta función compara cada u con el vector r
X[i,] <- as.vector(table(eps,exclude=NULL))[1:4] #conteo de categorías.
}
X[1:5,]
```

```
     [,1] [,2] [,3] [,4]
[1,]    3    2    5   NA
[2,]    2    3    5   NA
[3,]    2    1    7   NA
[4,]    2    2    6   NA
[5,]    3    2    5   NA
```

Normal Multivariada

- Un vector aleatorio $\mathbf{X} = (X_1, \ldots, X_d)$ tiene una distribución normal multivariada que se denota como $\mathcal{N}_d(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ si la densidad de $\mathbf{X}$ es:

$$f(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right), \qquad \mathbf{x} \in \mathbb{R}^d$$

  donde $\boldsymbol{\mu}$ es el vector de medias, y $\boldsymbol{\Sigma}$ es una matriz simétrica definida positiva de $d \times d$.
- Para generar una normal multivariada, se siguen dos pasos:
  - Se genera un vector $\mathbf{Z} = (Z_1, \ldots Z_d)$, donde $Z_1, \ldots, Z_d$ son normales estándares independientes.
  - Se transforma $\mathbf{Z}$ para que tenga el vector de medias y matriz de covarianzas dados. Esta transformación requiere factorizar la matriz de covarianzas $\boldsymbol{\Sigma}$.
- Para estandarizar, necesitamos descomponer la matriz de covarianzas en su "raíz cuadrada": tenemos que encontrar $\mathbf{B} \ni \mathbf{B}\mathbf{B}' = \boldsymbol{\Sigma}$. Entonces

$$\mathbf{Z} \sim \mathcal{N}_p(\mathbf{0}, \mathbf{I}_p) \implies \mathbf{X} = \boldsymbol{\mu} + \mathbf{B}\mathbf{Z} \sim \mathcal{N}_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

Métodos para descomponer Σ

- La descomposición de Σ se puede conseguir por tres métodos (con su función en `R`:
  - Método de descomposición espectral (eigenvalores): `eigen`.
  - Factorización de Cholesky `chol`.
  - descomposición en valor singular `svd`.
- Usualmente, no se aplica una transformación lineal a cada vector aleatorio en la muestra de manera individual; más bien se hace de golpe sobre todos los datos:
  - Supongan que $\mathbf{Z} = (Z_{ij})$ es una matriz de $n \times d$ donde Z_{ij} son iid $\mathcal{N}(0, 1)$. Los reglones de esta matriz son las observaciones de la distribución normal multivariada. La transformación que se aplica a la matriz de datos es

    $$\mathbf{X} = \mathbf{ZQ} + \mathbf{e}\boldsymbol{\mu}'$$

    donde $\mathbf{Q}'\mathbf{Q} = \boldsymbol{\Sigma}$, $\mathbf{e}$ es un vector de unos.
  - Los renglones de $\mathbf{X}$ son las observaciones aleatorias que tienen la media y varianza multivariada deseada.

Algoritmo para generar muestras de normal multivariada

Para generar una muestra de tamaño n de $\mathcal{N}_d(\boldsymbol{\mu}, \boldsymbol{\Sigma})$:

1. Generar una matriz $\mathbf{Z}$ de $n \times d$ con nd normales estándar independientes (n vectores aleatorios en $\mathbb{R}^d$).
2. Calcular una factorización $\boldsymbol{\Sigma} = \mathbf{Q}'\mathbf{Q}$.
3. Aplicar la transformación $\mathbf{X} = \mathbf{Z}\mathbf{Q} + \mathbf{e}\boldsymbol{\mu}'$
4. $\mathbf{X}$ es una matriz con la muestra deseada en cada renglón.

Ejemplo I

a. Obtener una muestra de un vector normal multivariado de orden 3: $\mathbf{X} = (X_1, X_2, X_3) \sim \mathcal{N}_3(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, con $\boldsymbol{\mu} = (1, 2, 3)'$ y

$$\boldsymbol{\Sigma} = \begin{pmatrix} 2.5 & 0.75 & 0.175 \\ 075 & 0.7 & 0.135 \\ 0.175 & 0.135 & 0.43 \end{pmatrix}.$$

- Con el método de descomposición espectral

```
rmvn.eigen <- function(n,mu,Sigma){
        d <- length(mu)
        ev <- eigen(Sigma, symmetric=T)
        lambda <- ev$values; V <- ev$vectors
        Q <- V %*% diag(sqrt(lambda)) %*% t(V)
        Z <- matrix(rnorm(n*d),nrow=n, ncol=d)
        X <- Z %*% Q + matrix(mu,n,d,byrow=T)
        X
}
```

- Con el método de descomposición de Cholesky

```
rmvn.chol <- function(n,mu,Sigma){
        d <- length(mu)
        Q <- chol(Sigma)
        Z <- matrix(rnorm(n*d),nrow=n)
        X <- Z %*% Q + matrix(mu,n,d,byrow=T)
        X
}
```

Ejemplo II

- Con el método de descomposición en valor singular.

```
rmvn.svd <- function(n,mu,Sigma){
        d <- length(mu)
        S <- svd(Sigma)
        Q <- S$u %*% diag(sqrt(S$d)) %*% t(S$v)
        Z <- matrix(rnorm(n*d),nrow=n, ncol=d)
        X <- Z %*% Q + matrix(mu,n,d,byrow=T)
        X
}
```

Ejemplo

Generamos la muestra con las funciones creadas

```
Sigma <- matrix(c(2.5, 0.75, 0.175,     0.75, 0.7, 0.135, 0.175, 0.135, 0.43),byrow=T,nrow=3)
mu <- c(1:3)
set.seed(1000)
 system.time(for(i in 1:100) X1 <- rmvn.eigen(1000,mu,Sigma))

   user  system elapsed
  0.087   0.053   0.043

set.seed(1000)
 system.time(for(i in 1:100) X2 <- rmvn.svd(1000,mu,Sigma))

   user  system elapsed
  0.076   0.101   0.045

set.seed(1000)
 system.time(for(i in 1:100) X3 <- rmvn.chol(1000,mu,Sigma))

   user  system elapsed
  0.026   0.000   0.026

head(X1,3);head(X2,3);head(X3,3)

         [,1]     [,2]     [,3]
[1,] 1.261355 2.962566 3.467948
[2,] 2.427646 2.486649 3.792581
[3,] 1.320942 2.998867 3.347262
         [,1]     [,2]     [,3]
[1,] 1.261355 2.962566 3.467948
[2,] 2.427646 2.486649 3.792581
[3,] 1.320942 2.998867 3.347262
          [,1]     [,2]     [,3]
[1,] 0.8188025 2.800563 3.505366
[2,] 2.3273613 2.512565 3.821431
[3,] 0.8761331 2.853971 3.390704
```

Distribución uniforme en $\mathbb{S}^d$

- La d-Esfera es el conjunto

$$\mathbb{S}^d = \{\mathbf{x} \in \mathbb{R}^d | \quad \|\mathbf{x}\| = \sqrt{\mathbf{x}'\mathbf{x}} = 1\}$$

- Vectores aleatorios uniformes distribuídos en $\mathbb{S}^d$ tienen direcciones distribuídas uniformes.
- Utilizamos la siguiente propiedad de normalidad:

Distribución en $\mathbb{S}^d$

Si $X_1, \ldots, X_d$ son iid $\mathcal{N}(0,1)$, entonces $\mathbf{U}$ se distribuye uniforme en $\mathbb{S}^d$, donde cada componente es de la forma:

$$U_i = \frac{X_j}{\|X\|}, \quad i = 1, \ldots d$$

Ejemplo

a. Función para generar puntos en la esfera unitaria $\mathbb{S}^d$:

```
runif.esfera <- function(n,d){
        M <- matrix(rnorm(n*d),nrow=n)
        L <- apply(M,1,function(x){sqrt(sum(x*x))})
        D <- diag(1/L)
        return(D %*% M)
}
X <- runif.esfera(100,2)
par(pty="s")
plot(X)
```