Capítulo 3.1: Generación de Variables Aleatorias

Métodos generales

Simulación

2020-10-28

Contents

1 Introducción

En el esquema de cuatro pasos el segundo consiste en transformar los números aleatorios en entradas al modelo. Las entradas suelen ser valores de las variables aleatorias adecuadas. Existen paquetes comerciales que son capaces de generar casi cualquier variable aleatoria, pero ni un paquete puede contener todas las distribuciones teóricas ni por supuesto las empíricas. Por otra parte, la aparición de nuevos algoritmos para distribuciones ya conocidas es constante.

Vamos a presentar tres métodos generales y alguno particular para generar valores de variables aleatorias.

2 Métodos generales

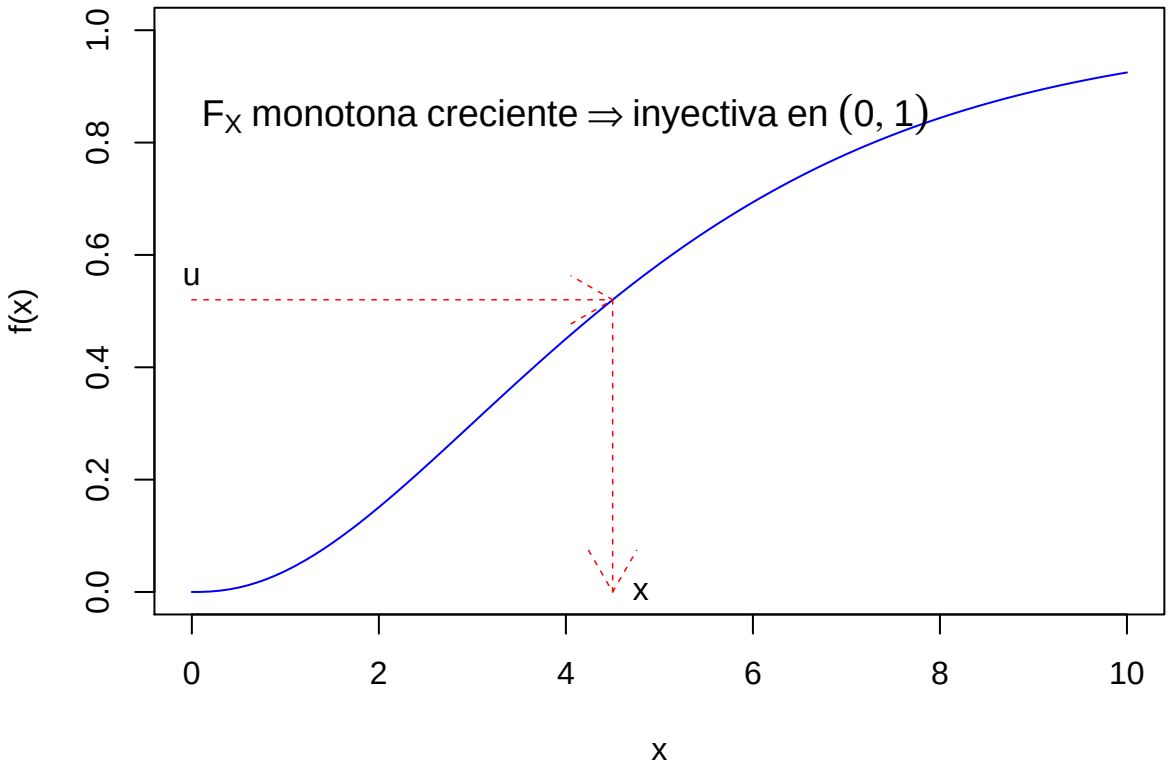
Presentaremos los siguientes métodos:

1. Método de la función inversa
2. Método de Aceptación y rechazo
3. Método de composición

2.1 Método de la función inversa

2.1.1 Caso Continuo

Sea X una VA con función de densidad f tal que $F(x) = \int_{-\infty}^{x} f(t)dt$



Proposición

Sea X una variable aleatoria continua con función de distribución F_X, la función F_X^{-1} está definida en $(0,1)$. Entonces, la variable aleatoria $U = F_X^{-1}(X)$ se distribuye según una $\mathcal{U}(0,1)$, luego si $U \sim \mathcal{U}(0,1)$ la variable $F_X^{-1}(U)$ tiene función de distribución F_X, la función de distribución de la variable aleatoria X.

Demostración

Sea F_U la función de distribución de U, dado un valor $u \in (0,1)$, se tiene

$$F_U(u) = P(U \leq u) = P(F_X(X) \leq u) = P\left(X \leq F_X^{-1}(u)\right) = F_X\left(F_X^{-1}(u)\right) = u$$

Lema Sea $U \sim U(0,1)$, y F función de distribución inyectiva (mónotona creciente en $F^{-1}(0,1)$). La VA $X = F^{-1}(U)$ tiene como función de de distribución a F.

Demostración

$F_X(x) = \Pr(X \leq x) = \Pr[F^{-1}(U) \leq x] =$ F monotona creciente $= \Pr[F(F^{-1}(U)) \leq F(x)] = \Pr(U \leq F(x) = F(x)$

El algoritmo general del método de inversión toma la forma,

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

2.- Hacer $X = F^{-1}(U)$

3.- Devolver X

Alguna de las distribuciones de las que se conoce F^{-1} de forma explícita son: Uniforme, Exponencial, Weibull, Cauchy, Beta, etc.

Notas:

1. En el caso de poder aplicarse el método de inversión proporciona una realización de la variable aleatoria por cada generación de un número aleatorio.
2. La obtención de F^{-1} de forma explícita puede ser dependiendo de la distribución en cuestión.
3. Hay situaciones en las que se conoce F^{-1} pero aplicar el método de la inversa es complicado como por ejemplo en el caso de la distribución beta.

Ejemplo 1: $U(a,\ b)$. Su función de distribución es,

$$F(x) = \begin{cases} 0 & x < a \\ \frac{x-a}{b-a} & a \leq x \leq b \\ 1 & x > b \end{cases}$$

$$\text{Despejando } u = \frac{x-a}{b-a} \Rightarrow x = a + (b-a) \cdot u \Rightarrow F^{-1}(U) = a + (b-a) \cdot U$$

El algoritmo queda,

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

2.- $X = a + (b-a) \cdot U$

3.- Devolver X

Ejemplo 2: Sea $X \sim Exp(\lambda)$. Su función de distribución es,

$$F(x) = 1 - e^{-\lambda x}, \quad x \geq 0$$

Despejando,

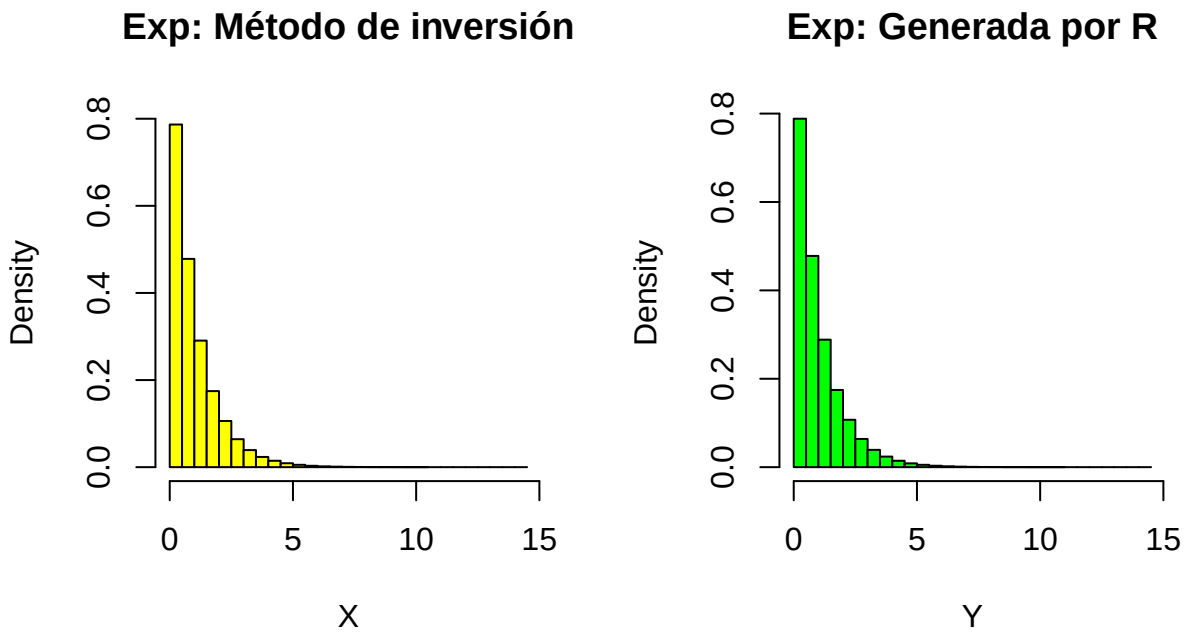
$$U = 1 - e^{-\lambda X} \Rightarrow e^{-\lambda X} = 1 - U \Rightarrow X = -\frac{1}{\lambda} \log(1 - U)$$

El algoritmo queda,

1.- Generar $U \sim \mathcal{U}(0, 1)$
2.- $X = -\frac{1}{\lambda} \log(U)$
3.- Devolver X

En R,

```
# Generar exp(1)

set.seed(1700)     # Semilla de simulación. Probar con 152104

Nsim <- 10^6       # Número de simulaciones
U <- runif(Nsim)
X <- - log(U)      # Genera valores de la Exp(lambda)
Y <- rexp(Nsim)    # Exponencial de R

# Gráficos*
par(mfrow = c(1,2))

hist(X, freq = FALSE, ylim = c(0,0.8), main = "Exp: Método de inversión", col = "yellow")
hist(Y, freq = FALSE, main = "Exp: Generada por R", col = "green")
```



```r
par(mfrow = c(1,1))
```

Ejemplo 3: Weibull, $W(\alpha,\ 1)$. Su función de distribución es,

$$F(x) = \begin{cases} 0 & x < 0 \\ 1 - e^{-x^{\alpha}} & x \geq 0 \end{cases}$$

Despejando,

$$U = 1 - \ \mathrm{e}^{-X^{\alpha}} \Rightarrow \log e^{-X^{\alpha}} = \log(1-U)$$

$$X = [-\mathrm{Ln}(1-U))]^{\frac{1}{\alpha}}$$

se obtiene,

$$F^{-1}(U) = [-\mathrm{Ln}(1-U))]^{\frac{1}{\alpha}}$$

Como U se distribuye según una $\mathcal{U}(0,1)$, entonces $1-\mathcal{U}$ se distribuye también según una $\mathcal{U}(0,1)$. Luego,

$$F^{-1}(U) = [-\mathrm{Ln}(U))]^{\frac{1}{\alpha}}$$

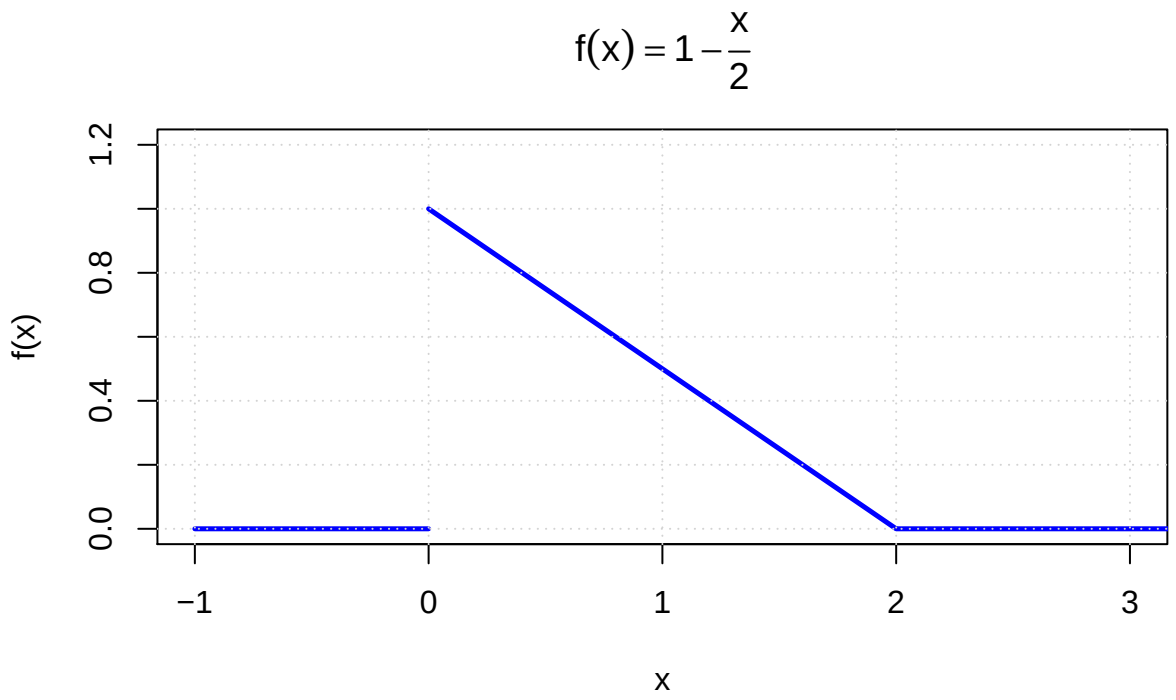
El algoritmo queda,

1.- Generar $U \sim \mathcal{U}(0,1)$ 2.- $X = [-\log(U)]^{\frac{1}{\alpha}}$ 3.- Devolver X

Ejemplo 4: Sea la VA X con función de densidad,

$$f(x) = \begin{cases} 1 - \frac{x}{2} & 0 \leq x \leq 2 \\ 0 & \text{otro caso} \end{cases}$$

su gráfica es,



su función de distribución es,

$$F(x) = \begin{cases} 0 & 0 < x \\ x - \frac{x^2}{4} & 0 \leq x < 2 \\ 1 & x \geq 2 \end{cases}$$

Hay que calcular $F^{-1}(U)$.

Sea $U \sim U(0,1)$,

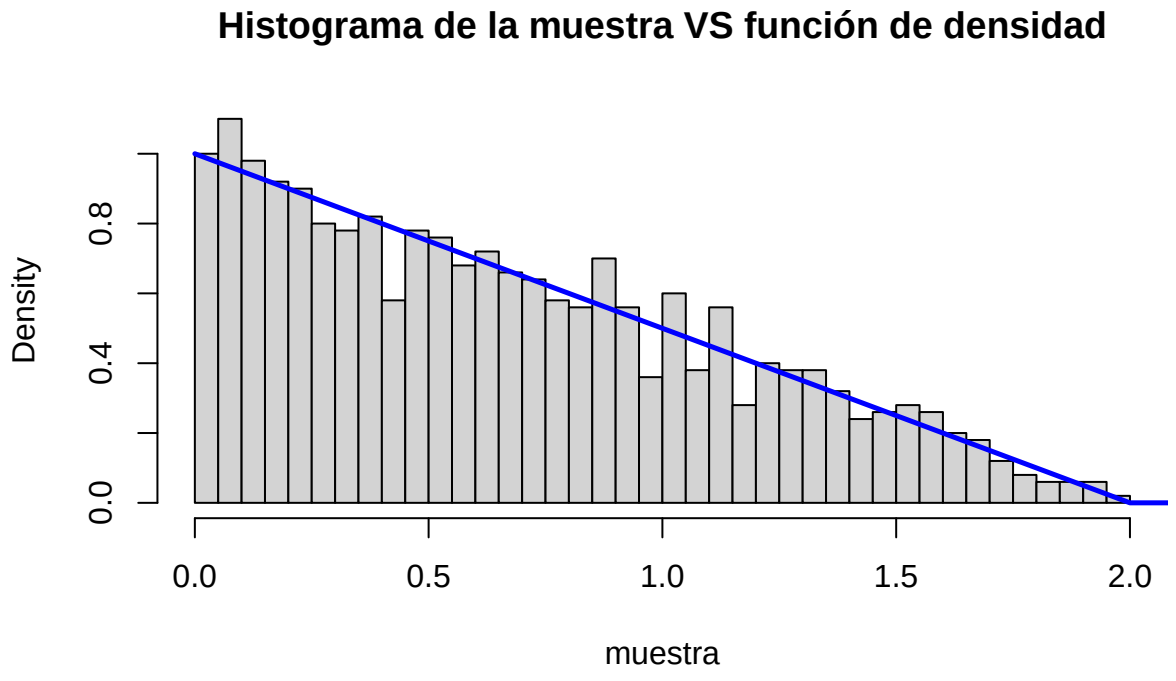
$$U = X - \frac{X^2}{4} \Rightarrow X^2 - 4X + 4U = 0 \Rightarrow X = 2(1 \pm \sqrt{1-U})$$

Ahora,

- $0 < 2(1 + \sqrt{1-U}) > 2$. Esta fuera del soporte.
- $0 \leq 2(1 - \sqrt{1-U}) \leq 2$.

El algoritmo es,

```
generar_X <- function(){
  U <- runif(1, min = 0, max = 1)
  return(2*(1 - sqrt(1 - U)))
}

# Generar muestra
Nsim <- 1000      # número de valores

muestra <- replicate(Nsim, generar_X())

hist(muestra, breaks = 30, freq = FALSE, main = "Histograma de la muestra VS función de densidad")
curve(f, from =  0, to = 4, col = "blue", lwd = 2.5, add = TRUE)
```



Contraste Kolmogorov-Smirnov

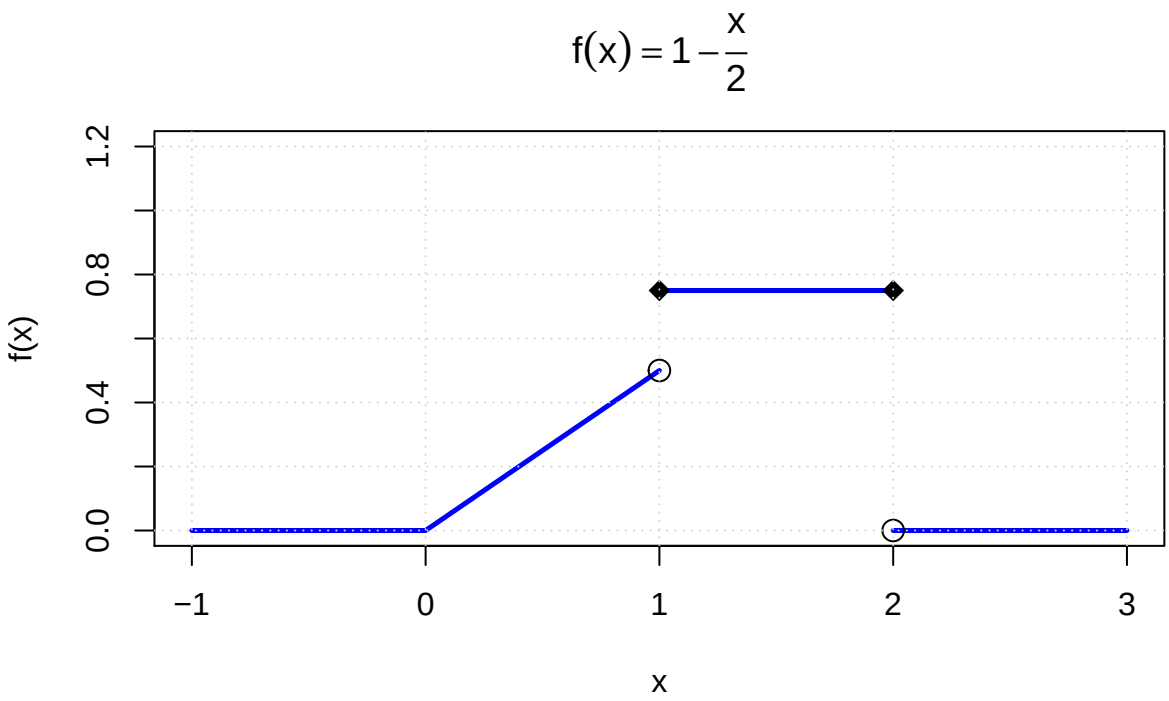
```r
# Función de distribución
FD <- function(x){
  ifelse(x < 0, 0,
         ifelse(x < 2, x - x^2/4, 1))
}

# Test de kolmogorov-Smirnov
ks.test(muestra, FD)
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  muestra
## D = 0.015608, p-value = 0.9679
## alternative hypothesis: two-sided
```

Ejemplo 5: Sea la va$X$ con función de densidad,

$$f(x) = \begin{cases} \frac{x}{2} & 0 \leq x < 1 \\ \frac{3}{4} & 1 \leq x \leq 2 \\ 0 & \text{otro caso} \end{cases}$$

su gráfica es,



Al ser una función de densidad a trozos, la función de distribución también lo será.

Caso 1: $x < 0 \Rightarrow F(x) = 0$

Caso 2: $0 \leq x < 1 \Rightarrow F(x) = \int_0^x \frac{t}{2} dt = \frac{x^2}{4}$

Caso 3: $1 \leq x \leq 2 \Rightarrow F(x) = F(1^-) + \int_1^x \frac{3}{4} dt = \frac{3}{4}x - \frac{1}{2}$

Caso 4: $x > 2 \Rightarrow F(x) = 1$

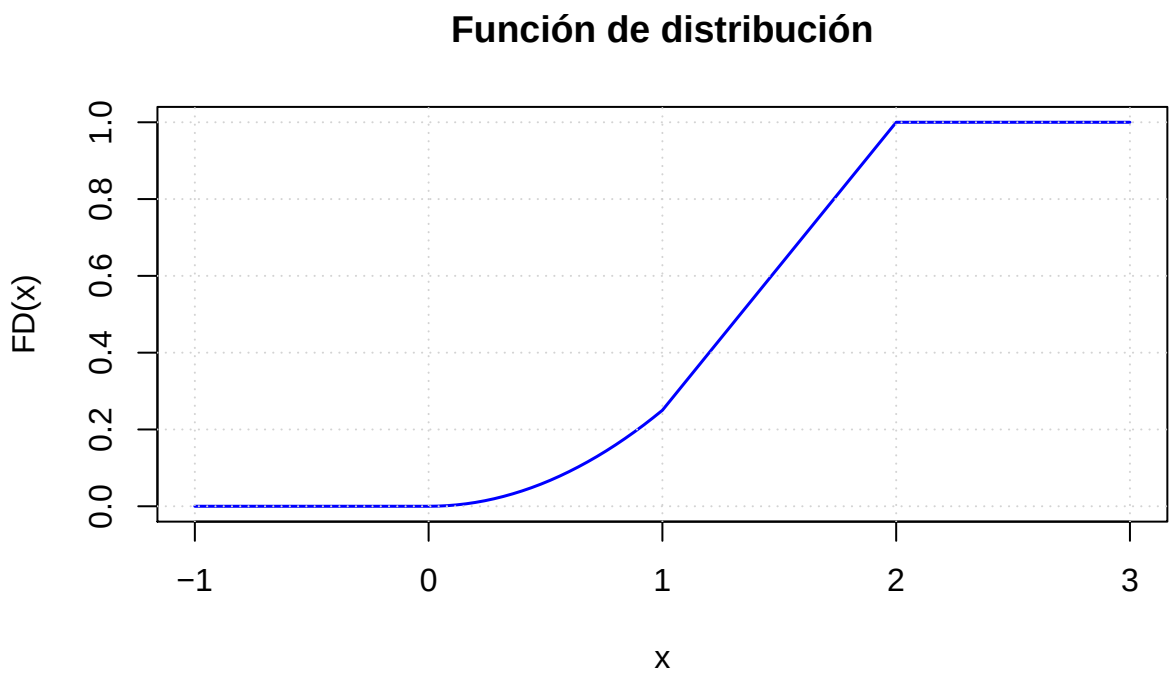
La función de distribución es,

$$F(x) = \begin{cases} 0 & 0 < x \\ \frac{x^2}{4} & 0 \leq x < 1 \\ \frac{3}{4}x - \frac{1}{2} & 1 \leq x \leq 2 \\ 1 & x \geq 2 \end{cases}$$

La función de distribución de X también es monotona creciente,

```
FD <- function(x){
  ifelse(x < 0, 0, ifelse( x < 1, x^2/4, ifelse(x < 2, 0.75*x - 0.5, 1)))
}
```

```r
# gráfica función de distribución

curve(FD, -1, 3, col = "blue", lwd = 1.5, main = "Función de distribución")
grid()
```



La función inversa de F también será a trozos.

Dado $U \sim U(0,1)$:

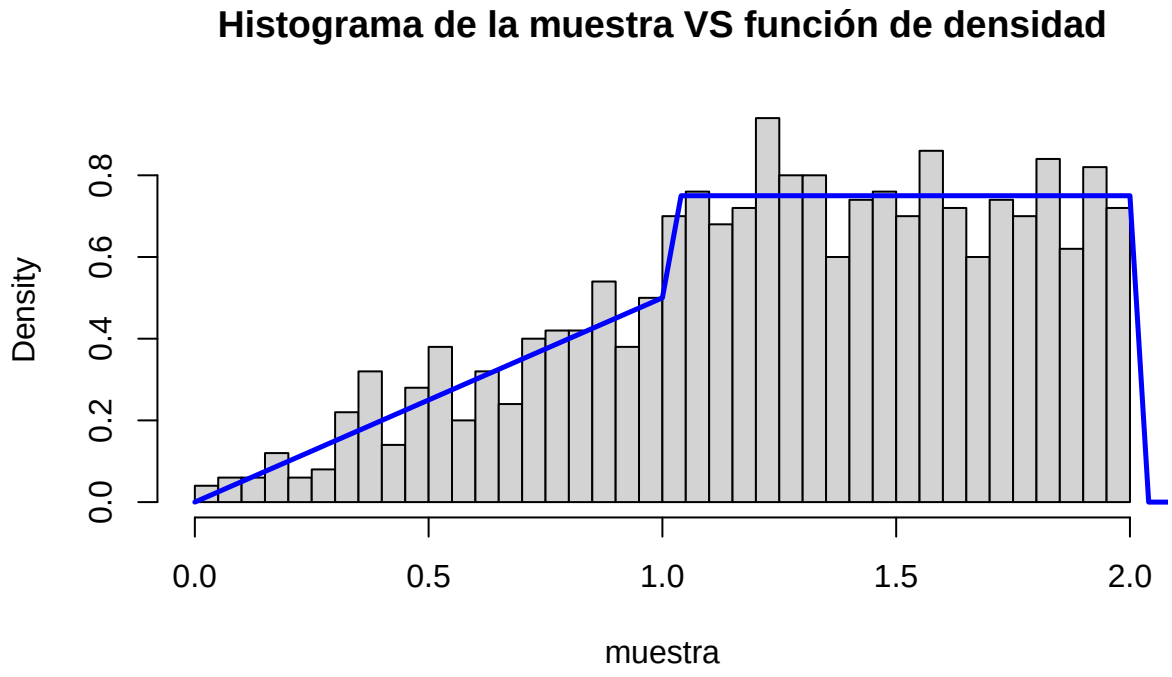
Si $U \in [0, 0.25) \Rightarrow U = \frac{X^2}{4} \Rightarrow X^2 = 4U \Rightarrow X = +2\sqrt{U}$

Si $U \in [0.25, 1) \Rightarrow U = \frac{3}{4}X - \frac{1}{2} \Rightarrow X = \frac{4U+2}{3}$

El algoritmo queda,

```r
generar_inv_trozos <- function(){
  U <- runif(1, min = 0, max = 1)
  if(U < 0.25){
    X = 2 * sqrt(U)
  } else{
    X <- (4*U + 2)/3
  }
  return(X)
}

# Generar muestra
Nsim <- 1000      # número de valores

muestra <- replicate(Nsim, generar_inv_trozos())

hist(muestra, breaks = 30, freq = FALSE, main = "Histograma de la muestra VS función de densidad")
```

```r
curve(f, from =  0, to = 4, col = "blue", lwd = 2.5, add = TRUE)
```



Contraste Kolmogorov-Smirnov

```r
# Test de kolmogorov-Smirnov
ks.test(muestra, FD)
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  muestra
## D = 0.019801, p-value = 0.8278
## alternative hypothesis: two-sided
```

2.1.2 Caso Discreto

Básicamente es el mismo método que en el caso continuo.

Proposición Sea X una variable aleatoria discreta con función de distribución $F_X(x)$. Definimos la función $\overline{F}(u) = \min\{x : F_X(x) \geq u\}$. Si U es una variable aleatoria con función de distribución $\mathcal{U}(0,1)$, entonces $\overline{F}(U)$ se distribuye según F_X.

Algoritmo para generar valores de una distribución de probabilidad discreta

Sea una variable aleatoria discreta X, con función de masa de probabilidad

$$P(X_j = x_j) = p_j, \quad \forall j = 1, 2, \ldots \quad \sum_j p_j = 1$$

tal que $x_1 < x_2 < x_3 < \cdots$.

Generamos $U \sim \mathcal{U}(0,1)$. Entonces,

$$X = \begin{cases} x_1 & U < p_1 \\ x_2 & p_1 \leq U < p_1 + p_2 \\ x_3 & p_1 + p_2 \leq U < p_1 + p_2 + p_3 \\ \vdots & \\ x_j & p_1 + \cdots + p_{j-1} \leq U < p_1 + \cdots + p_{j-1} + p_j \\ \vdots & \end{cases}$$

es decir,

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

donde F es la función de distribución de la VA X.

Para ver que X así construida tiene la distribución pedida hay que tener en cuenta que dado $0 < a < b < 1$ y $U \sim \mathcal{U}(0,1)$, entonces $P(a < U < b) = b - a =$ longitud del intervalo (a, b). Por lo tanto,

$$P(X = x_j) = P(\sum_{j=1}^{j-1} \leq U < \sum_{j=1}^{j}) = (p_1 + \cdots + p_{j-1} + p_j) - (p_1 + \cdots + p_{j-1}) = p_j$$

es decir X sigue la distribución pedida.

El algoritmo queda

Generar $U \sim \mathcal{U}(0,1)$
Si $U < p_1$, hacer $X = x_1$
OtroSi $U < p_1 + p_2$, hacer $X = x_2$
OtroSi $U < p_1 + p_2 + p_3$, hacer $X = x_3$
$\vdots$

Ejemplo 6: Sea la VA discreta X con función de masa de probabilidad

$$P(X = 1) = 0.35, \quad P(X = 2) = 0.15 \quad P(X = 3) = 0.40 \quad P(X = 4) = 0.10$$

Utilizando el método de la inversa, escribe un algoritmo para generar valores de X.

Solución

Paso 1: Dividimos el intervalo $[0, 1]$ en 4 sub-intervalos cada uno con longitud p_i

- $I_1 = [0, 35) \rightarrow$ longitud $= 0.35 = p_1$
- $I_2 = [35, 0.50) \rightarrow$ longitud $= 0.15 = p_2$
- $I_3 = [50, 90) \rightarrow$ longitud $= 0.40 = p_3$

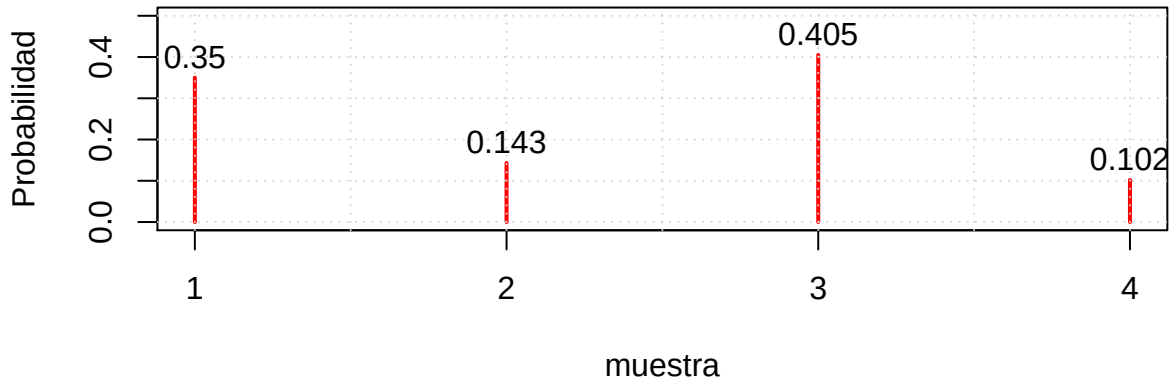
- $I_4 = [0.90, 1] \rightarrow$ longitud $= 0.10 = p_4$

Paso 2: Generamos $U \sim \mathcal{U}(0, 1)$,

- Si $U \in I_i$, entonces $X = x_i$

En R,

```r
generarVA1 <- function(){
  U <- runif(1)
  if(U < 0.35){
    return(1)
  } else if(U < 0.5){
    return(2)
  } else if(U < 0.9){
    return(3)
  } else{
    return(4)
  }
}

# Simulación
set.seed(1936)

Nsim <- 1000   # Tamaño de la muestra que se genera

muestra <- replicate(Nsim, generarVA1())
valores <- table(muestra)/Nsim

plot(valores, ylim = c(0, 0.5), ylab = "Probabilidad", col = "red")
grid()
text(x = c(1, 2, 3, 4), y = valores+0.05, labels = valores)
```



Se observa que el número de comparaciones que hace el algoritmo depende de como estén ordenados los x_i, si se ordena los x_i de forma que los de mayor probabilidad estén al principio se consigue un algoritmo más eficiente.

Ejemplo 7: Sea $X \sim Ge(p)$

Dada la VA $X \sim Ge(p)$, entonces,

- $X \equiv$ Número de fracasos independientes hasta obtener un éxito
- $X \in \{0, 1, 2, 3, \ldots\}$
- La probabilidad de éxito es p, luego la probabilidad de fracaso es $q = 1 - p$
- La función de masa de probabilidad es

$$P(X = j) = (1 - p)^j p = q^j p$$

y su función de distribución es,

$$F(j) = P(X \leq j) = 1 - P(X > j)$$

aplicando el método de la función inversa

$$X = j \qquad \text{Si } F(j-1) \leq U < F(j)$$

ahora,

$P(X > j) \equiv$ Los primeros j intentos han sido fracasos (independientes) y la probabilidad de fracaso es q.

luego,

$$P(X > j) = q^{j+1} \Rightarrow F(j) = 1 - P(X > j) = 1 - q^{j+1}$$

Entonces, dado $U \sim \mathcal{U}(0, 1)$ y sustituyendo el valor de $F(\cdot)$,

$$F(j-1) \leq U < F(j) \Rightarrow 1 - q^j \leq U < 1 - q^{j+1}$$

multiplicando por (-1) la expresión anterior

$$-1 + q^j \geq -U > -1 + q^{j+1}$$

sumando 1,

$$q^j \geq 1 - U > q^{j+1}$$

tomando logaritmos,

$$j \log(q) \geq \log(1 - U) > (j + 1) \log(q)$$

ahora, $q \in (0, 1)$, luego $\log(q) < 0$, dividiendo por $\log(q)$ queda

$$j \leq \frac{\log(1 - U)}{\log(q)} < j + 1$$

Por tanto, $\frac{\log(1-U)}{log(q)} \in [j, j + 1)$, es decir

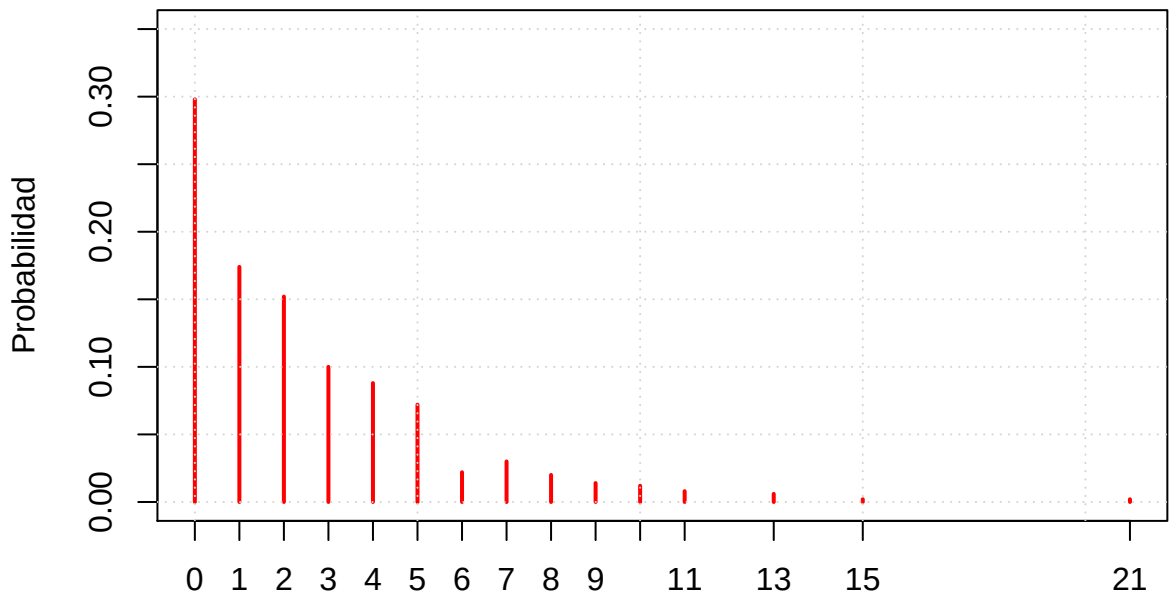
$$j = \left\lfloor \frac{\log(1 - U)}{log(q)} \right\rfloor$$

El algoritmo es,

Generar $U \sim \mathcal{U}(0, 1)$
Hacer $X = \left\lfloor \frac{\log(1-U)}{log(q)} \right\rfloor$

En R,

```r
generarGeo <- function(q){
  # Entrada
  # q       - probabilidad de fracaso
  U <- runif(1)
  X <- floor(log(1-U)/log(q))
  return(X)
}

# Simulación
set.seed(1936)

Nsim <- 500   # Tamaño de la muestra que se genera

p <- 0.3
q <- 1 - p

muestra <- replicate(Nsim, generarGeo(q))

valores  <- table(muestra)/Nsim
valoresR <- table(rgeom(Nsim, prob = p))/Nsim

plot(valoresR, ylim = c(0, 0.35), ylab = "Probabilidad", col = "red")
grid()
```



```
chisq.test(valores, valoresR)
```

```
## Warning in chisq.test(valores, valoresR): Chi-squared approximation may be
## incorrect
```

```
##
##  Pearson's Chi-squared test
##
## data:  valores and valoresR
## X-squared = 165, df = 143, p-value = 0.1005
```

Ejemplo 8: Sea $X \sim B(1,p)$ (Bernoulli)

$Valores$	$P(X=x_i)$
$x=1=$ Éxito	p
$x=0=$ Fracaso	$1-p$

El algoritmo queda,

1.- Generar $U \sim \mathcal{U}(0,1)$
2.- Si $U < 1-p$, hacer $X=0$
3.- Otro, hacer $X=1$
4.- Devolver X

El algoritmo queda,

La probabilidad de aceptar el punto x_1, es decir de generar un valor de X es,

$$\frac{\text{Area bajo la curva } \ f(x)}{\text{Area del rectangulo}} = \frac{1}{(b-a)\cdot c}$$

este valor es la eficiencia del método

Ejemplo 9:

1. • Sea $X \sim Be(2,\ 2)$. Con función de densidad

$$f(x) = 6x(1-x) \quad 0 \leq x \leq 1$$

y vamos a considerar como función $g(x)$ la función de densidad de una $\mathcal{U}(0,1)$.

$$g(x) = 1 \quad 0 \leq x \leq 1$$

Buscamos un valor $c > 0$ tal que

$$f(x) \leq c \cdot g(x) \quad 0 \leq x \leq 1$$

definimos

$$h(x) = \frac{f(x)}{g(x)} = \frac{f(x)}{1} = f(x) \quad 0 \leq x \leq 1$$

y definimos

$$c = \ h(x) = \ f(x)$$

El máximo de $f(x)$ se alcanza en $x = 0.5$, $f(0.5) = \frac{3}{2} = 1.5 = c$. Luego,

$$f(x) \leq \frac{3}{2} \cdot g(x) \Rightarrow f(x) \leq \frac{13}{2} \cdot 1$$

La eficiencia del método está determinada por la relación entre las áreas del rectángulo y del área bajo la curva $f(x)$,

$$\frac{\text{Area bajo la curva} f(x)}{\text{Area del rectangulo}} = \frac{1}{(1-0) \cdot 1.5} = \frac{2}{3} = 0.6$$

El algoritmo queda,

Vamos a ejecutar el algoritmo manualmente para demostrar como se opera, de 5 intentos se aceptan 2 y se rechazan 3. Los valores que se obtiene de la beta son: 0.32746728 y 0.26846862.

2.1.3 Ejercicios

1.- En un taller con una sola máquina se reciben trabajos de forma aleatoria. El tiempo entre llegadas es exponencial con media 2 horas. El tiempo necesario para procesar un trabajo es uniforme entre 1,1 y 2 horas. Suponga que el primer trabajo llega en el tiempo 0. Determine el tiempo de salida para los 10 primeros trabajos.

2.- Desarrollar dos métodos para generar una variable aleatoria X cuya distribución de probabilidad está dada por:

$$\Pr(X = i) = \frac{\frac{\lambda^i}{i!} e^{-\lambda}}{\sum_{j=0}^{k} \frac{\lambda^j}{j!} e^{-\lambda}} \qquad i = 0, \ldots, k$$

2.2 El método de aceptación-rechazo

Cuando de una variable aleatoria se conoce su función de densidad, pero no es posible calcular su función de distribución, o siendo posible no es eficiente (el caso de la beta) un método para generar valores de la variable aleatoria es el método del rechazo.

2.2.1 Caso continuo

Sea X una variable aleatoria con función de densidad conocida f y supongamos que existe otra función de densidad g (Y variable aleatoria con función de densidad g) que domina a f y de la que se conoce un método eficiente de simulación. Entonces el método de aceptación-rechazo es:

La función g domina f, si verifica que

$$f(x) \leq c \cdot g(x) \quad \forall\ x,\ con\ c > 1$$

entonces el algoritmo de aceptación y rechazo es el siguiente

Hasta que $U \leq \frac{f(Y)}{c \cdot g(Y)}$
  Generar $Y \sim g$
  Generar $U \sim U(0,1)$
Devolver $X = Y$

Teorema *El método de aceptación y rechazo genera valores de la variable aleatoria X.*

Demostración

Hay que demostrar que

$$\left(Y | U \leq \frac{f(Y)}{c \cdot g(Y)}\right) \sim X$$

Sea $h(y,u)$ la función de densidad de la variable bidimensional (Y,U). Por ser Y y U independientes se verifica que

$$h(y,x) = g(Y) \cdot f_U(u) = g(Y) \cdot 1 = g(Y)$$

Calculamos

$$\Pr(X \leq t) = \Pr\left(Y \leq t | U \leq \frac{f(Y)}{c \cdot g(Y)}\right) = \frac{\Pr\left(Y \leq t, U \leq \frac{f(Y)}{c \cdot g(Y)}\right)}{\Pr\left(U \leq \frac{f(Y)}{c \cdot g(Y)}\right)}$$

donde el numerador es

$$\begin{aligned}
\Pr\left(Y \leq t, U \leq \frac{f(Y)}{c \cdot g(Y)}\right) &= \int_{-\infty}^{t} \int_{0}^{\frac{f(y)}{c \cdot g(y)}} h(y,u)\ du\ dy \\
&= \int_{-\infty}^{t} \int_{0}^{\frac{f(y)}{c \cdot g(y)}} g(y)\ du\ dy \\
&= \int_{-\infty}^{t} g(y) \left[\int_{0}^{\frac{f(y)}{c \cdot g(y)}} du\right] dy \\
&= \int_{-\infty}^{t} g(y) \cdot \frac{f(y)}{c \cdot g(y)}\ dy \\
&= \int_{-\infty}^{t} \frac{f(y)}{c}\ dy \\
&= \frac{1}{c} \int_{-\infty}^{t} f(y)\ dy \\
&= \frac{1}{c} F(t)
\end{aligned}$$

y el denominador

$$\begin{aligned}\Pr\left(U \leq \tfrac{f(Y)}{c\cdot g(Y)}\right) &= \int_{-\infty}^{\infty}\int_{0}^{\frac{f(y)}{c\cdot g(y)}} h(y,u)\ du\ dy = \int_{-\infty}^{\infty}\int_{0}^{\frac{f(Y)}{c\cdot g(Y)}} g(y)\ du\ dy \\ &= \int_{-\infty}^{\infty} g(y)\left[\int_{0}^{\frac{f(Y)}{c\cdot g(Y)}}\ du\right]\ dy \\ &= \int_{-\infty}^{\infty} g(y)\cdot\left(\frac{f(y)}{c\cdot g(y)}\right)\ dy \\ &= \frac{1}{c}\int_{-\infty}^{\infty} f(y)dy \\ &= \frac{1}{c}\end{aligned}$$

sustituyendo,

$$\Pr(X \leq t) = \Pr\left(Y \leq t | U \leq \frac{f(Y)}{c\cdot g(Y)}\right) = \frac{\Pr\left(Y \leq t, U \leq \frac{f(Y)}{c\cdot g(Y)}\right)}{\Pr\left(U \leq \frac{f(Y)}{c\cdot g(Y)}\right)} = \frac{\frac{1}{c}F(t)}{\frac{1}{c}} = F(t).$$

Fijándonos en el denominador,

$$\Pr\left(U \leq \frac{f(Y)}{c\cdot g(Y)}\right) = \frac{1}{c}$$

encontramos que la probabilidad de aceptar un valor de X es $1/c$. Este valor da la eficiencia del método de aceptación y rechazo en cada caso y obviamente lo que interesa es que c sea lo más pequeño posible. Por lo tanto, ante distintas funciones g posibles conviene elegir aquella que está asociada al menor valor de c.

$$c = \max_{x} \frac{f(x)}{g(x)}$$

Ejemplo 10: Sea $X \sim Be(2,\ 2)$. Con función de densidad

$$f(x) = 6x(1-x) \quad 0 \leq x \leq 1$$

y vamos a considerar como función $g(x)$ la función de densidad de una $\mathcal{U}(0,1)$.

$$g(x) = 1 \quad 0 \leq x \leq 1$$

Buscamos un valor $c > 0$ tal que

$$f(x) \leq c\cdot g(x) \quad 0 \leq x \leq 1$$

definimos

$$h(x) = \frac{f(x)}{g(x)} = \frac{f(x)}{1} = f(x) \quad 0 \leq x \leq 1$$

y definimos $c = h(x) = f(x)$

El máximo de $f(x)$ se alcanza en $x = 0.5$, $f(0.5) = \frac{3}{2} = 1.5 = c$. Luego,

$$f(x) \leq \frac{3}{2} \cdot g(x) \Rightarrow f(x) \leq \frac{13}{2} \cdot 1$$

La eficiencia del método está determinada por la relación entre las áreas del rectángulo y del área bajo la curva $f(x)$,

$$\frac{\text{Area bajo la curva} f(x)}{\text{Area del rectangulo}} = \frac{1}{(1-0) \cdot 1.5} = \frac{2}{3} = 0.6$$

El algoritmo queda,

Mientras $Y_1 \geq f(X_1)$
  Generar $U_1, U_2 \sim g$
  Hacer $X_1 = U_1$
  Hacer $Y_1 = 1.5 \cdot U_2$
  Calcular $f(X_1) = 6 \cdot X_1(1 - X_1)$
Devolver X_1

Vamos a ejecutar el algoritmo manualmente para demostrar como se opera,

```
set.seed(1714)

U1 <- runif(5, min = 0, max = 1)
U2 <- runif(5, min = 0, max = 1)

X1 <- U1
Y1 <- 1.5*U2

f_X1 <- 6*X1*(1-X1)

SALIDA <- ifelse(Y1 > f_X1, 0, 1)

solucion <- data.frame(U1, U2, X1, Y1, f_X1, SALIDA)

# Imprimir tabla
knitr::kable(solucion)
```

U1	U2	X1	Y1	f_X1	SALIDA
0.9946949	0.6547601	0.9946949	0.9821401	0.0316617	0
0.8363427	0.7274897	0.8363427	1.0912345	0.8212415	0
0.6732251	0.9965817	0.6732251	1.4948726	1.3199584	0
0.7201614	0.1229329	0.7201614	0.1843994	1.2091738	1
0.6211803	0.7855215	0.6211803	1.1782822	1.4118920	1

de 5 intentos se aceptan 2 y se rechazan 3. Los valores que se obtiene de la beta son: 0.7201614 y 0.6211803.

2.2.2 Caso Discreto

Sea X una variable aleatoria discreta de la que se quiere generar valores y supongamos otra variable aleatoria discreta Y de la que se dispone un algoritmo cumpliendo las siguientes condiciones:

i. Para todo x_j del rango de X, tal que $\Pr(X = x_j) > 0$, se verifica que $\Pr(Y = x_j) > 0$

ii. Existe $c > 0$ tal que $\Pr(X = x_j) \leq c \cdot P(Y = y_j)$, para todo x_j tal que $\Pr(X = x_j) > 0$

El algoritmo es,

Paso 1: Generar $Y \sim F_Y$
Paso 2: Generar $U \sim U(0,1)$
Paso 3: Mientras $U \geq \frac{\Pr(X=Y)}{c \cdot \Pr(X=Y)}$
Paso 4: $Y \sim F_Y$
Paso 5: $U \sim U(0,1)$
Paso 6: FinMientras
Paso 7: Devolver Y

Proposición La probabilidad de aceptar Y es de $\frac{1}{c}$.

Demostración

Sea X variable aleatoria discreta con soporte, $Sop(X) = \{x_1, x_2, \ldots\}$ y llamamos $p_i = \Pr(X = x_i)$ y sea la variable aleatoria discreta Y tal que $\Pr(Y = x_i) = q_i > 0$ para todo $x_i \in Sop(X)$.

Entonces,

$$\begin{aligned}
\Pr(\text{aceptar Y}) &= \Pr(\{Y = x_1, U < \tfrac{p_1}{c \cdot q_1}\} \cup \{Y = x_2, U < \tfrac{p_2}{c \cdot q_2}\} \cup \ldots) \\
&= \Pr\left[\bigcup_{j \geq 1} \left\{Y = x_j, U < \frac{p_j}{c \cdot q_j}\right\}\right] \\
&= \text{ es la unión de conjuntos disjuntos y las va } Y \text{ y } U \text{ son independientes} \\
&= \Pr\left[\bigcup_{j \geq 1} Y = x_j\right] \cdot \Pr\left[U < \frac{p_j}{c \cdot q_j}\right] \\
&= \sum_{j \geq 1} \Pr(Y = x_j) \Pr(U < \frac{p_j}{c \cdot q_j}) \\
&= \sum_{j \geq 1} q_j \cdot \frac{p_j}{c \cdot q_j} \\
&= \sum_{j \geq 1} \frac{p_j}{c} \\
&= \frac{1}{c}
\end{aligned}$$

El número esperado de iteraciones que realiza el algoritmo hasta aceptar un valor de Y es una variable aleatoria geométrica con probabilidad de éxito $\frac{1}{c}$ y media c. Interesa que $\frac{1}{c}$ sea el mayor valor posible, o lo que es equivalente que c tome le menor valor posible que verifique

$$p_i \leq c \cdot p_i \qquad i \in Sop(X)$$

Veamos que el algoritmo genera valores de la VA X.

$$
\begin{aligned}
\Pr(\text{generar } x_j) &= \sum_{k\geq 1} \Pr(\text{generar } x_j \text{ en la iteración } k) \\
&= \sum_{k\geq 1} \Pr(\text{rechazar } Y \ k-1 \text{ veces y aceptar } Y = x_j \text{ en la iteración } k) \\
&= \sum_{k\geq 1} (1-\frac{1}{c})^{k-1} \Pr\left(Y = x_j, U \leq \frac{p_j}{c\cdot q_j}\right) \\
&= \sum_{k\geq 1} (1-\frac{1}{c})^{k-1} q_j \frac{p_j}{c\cdot q_j} \\
&= \sum_{k\geq 1} (1-\frac{1}{c})^{k-1} \frac{p_j}{c} \\
&= \frac{p_j}{c} \sum_{k\geq 1} (1-\frac{1}{c})^{k-1} \\
&= p_j
\end{aligned}
$$

Ejemplo Sea X una variable aleatoria cuyo soporte es, $Sop(X) = \{1,2,3,4,5,6\}$ y función de masa de probabilidad 0.12, 0.20, 0.05, 0.15, 0.26, 0.22. Se pide generar valores por el método de aceptación y rechazo.

Solución

Tomamos como VA Y la unifore discreta en $U_D[1,6]$ y buscamos

$$\max_{x_i \in Sop(X)} p_i q_i = \max\left(\frac{0.12}{1/6}, \frac{0.20}{1/6}, \frac{0.05}{1/6}, \frac{0.15}{1/6}, \frac{0.26}{1/6}, \frac{0.22}{1/6}\right) = 1.56$$

El valor esperado del número de iteraciones es la esperanza de una $Geom(p = \frac{1}{1.56}) = 0.64$

2.2.3 Ejercicios

1.- Determine las funciones $g(x)$ y $h(x)$ para aplicar el método del rechazo a la siguiente función:

$$f(x) = \frac{\operatorname{sen}(x) + \cos(x)}{2}; \qquad 0 \leq x \leq \frac{\pi}{2}$$

2.- El tiempo entre llegadas de clientes a la peluquería Tijeras viene dado por la función de densidad,

$$f_1(t) = \frac{k_1}{t}; \qquad 12 \leq t \leq 20$$

El tiempo para realizar un servicio está dado por la función,

$$f_2(t) = \frac{k_2}{t^2}; \qquad 18 \leq t \leq 22$$

Determine k_1 y k_2 de forma que f_1 y f_2 sean funciones de densidad. Utilice el método de rechazo para determinar cuándo saldrá el primer cliente y cuándo llegará el siguiente. Suponga que el primer cliente llega en $T = 0$.

3.- La demanda de una costosa pieza de repuesto de un jet de pasajeros es 0, 1, 2 o 3 unidades por día con probabilidades 0.2, 0.3, 0.4 y 0.1 respectivamente. El taller de mantenimiento de la línea aérea comienza sus operaciones con una existencia de 5 unidades y restablece el nivel de existencias a 5 unidades inmediatamente después de que éste cae por debajo de 2 unidades.

a. Realice un modelo de simulación.

b. ¿Cuántos días transcurrirán hasta que ocurra el primer reabastecimiento?

2.3 El método de composición

2.3.1 Caso continuo

Hay veces que la variable a simular se puede descomponer como una mixtura de variables aleatorias. El caso general es cuando dada X variable aleatoria con función de densidad $f(x)$ se puede escribir,

$$f\left(x\right) = \int g\left(x|y\right)\ \mathrm{dH}(y)$$

donde $g\left(x|y\right)$ es una familia de densidades que dependen del parámetro Y, variable aleatoria con función de distribución $H(y)$.

Algoritmo

1.- Generar $Y \sim H$
2.- Generar $X \sim g(\cdot|Y)$
3.- Devolver X

Las mixturas pueden ser finitas o infinitas el caso más interesante desde un punto de vista práctico es el de las mixturas finitas.

Sea una variable aleatoria X con función de densidad $f(x)$ y sean dos variables aleatorias X_1 y X_2 con sus respectivas funciones de densidad $f_1(x)$ y $f_2(x)$,

$$f\left(x\right) = t\ f_1\left(x\right) + \left(1-t\right) f_2\left(x\right); \quad 0 < t < 1$$

esto es equivalente a jugar a una lotería,

$$X \sim \begin{pmatrix} t & 1-t \\ X_1 & X_2 \end{pmatrix}$$

entonces,

$$\begin{array}{rcl} P(X \leq x) & = & \mathrm{tP}(X_1 \leq x) + (1-t)P(X_2 \leq x) \\ & = & t\int_{-\infty}^{x} f_1(y)\mathrm{dy} + (1-t)\int_{-\infty}^{x} f_2(y)\mathrm{dy} \\ & = & \int_{-\infty}^{x} [\mathrm{tf_1}(y) + (1-t)f_2(y)]\mathrm{dx} \\ & = & \int_{-\infty}^{x} f(y)\mathrm{dx} \end{array}$$

La base del método es que si se puede descomponer la variable aleatoria X de esta forma, en vez de simular la variable aleatoria X, con probabilidad t se simulará la variable aleatoria X_1 y con probabilidad $(1-t)$ se simulará X_2. Por supuesto, si simular X_1 y X_2 es más "**complicado**" que simular X no se aplicaría.

El algoritmo general es,

Generar $U \sim U(0,1)$
  Si $U \leq t$, entonces Generar $X \sim f_1(X)$
  Si $U > t$, entonces Generar $X \sim f_2(X)$ Devolver X

La situación que más se presenta es aquella en que hay que simular valores de una variable aleatoria $\mathbf{X}$ y se conoce una variable $\mathbf{X_2}$ de la que se dispone un método menos costoso (en tiempo, en complejidad, etc.). Entonces se realiza la descomposición siguiente,

$$f\left(x\right) = tf_1\left(x\right) + \left(1-t\right)\frac{f\left(x\right) - tf_{1(x)}}{\left(1-t\right)}$$

luego,

con probabilidad t hay que simular	$f_1(x)$
con probabilidad $(1-t)$ hay que simular	$f_2(x) = \frac{f(x)-tf_1(x)}{(1-t)}$

si t está cerca de 1, será interesante la descomposición, en caso contrario se debe elegir otro método.

Además, t no puede tomar un valor arbitrario ya que $f_2(x)$ tiene que ser una función de densidad. En resume, las condiciones son,

1. $f(x) - t\ f_1(x) \geq\ 0$
2. $\int_R f_2(x)\mathrm{dx} = 1$

la segunda condición es inmediata y de la primera obtenemos que el valor t debe cumplir, $t = \min_x \frac{f(x)}{f_1(x)}$

Ejemplo 11: Sea X una variable aleatoria con función de densidad,

$$f(x) = \begin{cases} 0.4x + 0.9 & x \in \left[0, \frac{1}{2}\right] \\ \\ -0.4x + 1.3 & x \in \left[\frac{1}{2}, 1\right] \end{cases}$$

y se pide escribir un algoritmo para simular esta variable.

Se puede aplicar el método de la función inversa o el de rechazo. Aquí desarrollamos el segundo.

La función de densidad tiene un aspecto como,

es del tipo triangular. Como variable X_1 se toma la $\mathcal{U}(0,1)$, $f_1(x) = 1$, $x \in [0,\ 1]$. La descomposición es,

$$f(x) = t \cdot 1 + (1-t) \cdot \frac{f(x) - t}{(1-t)}$$

aquí $t = 0.9$ (¿por qué?) por lo que

$$f_2(x) = \begin{cases} 0.4x & x \in \left[0, \frac{1}{2}\right] \\ 4(1-x) & x \in \left[\frac{1}{2}, 1\right] \end{cases}$$

Entonces con probabilidad 0.9 simulamos una $\mathcal{U}(0,1)$ y con probabilidad 0.1 la variable aleatoria X_2. En este caso es beneficioso el método.

Todavía no está terminado el ejercicio. Hay que simular X_2 y lo haremos aplicando el método de la inversa.

Calculamos la función de distribución de X_2.

$$F_2(y) = \begin{cases} 0 & y < 0 \\ 2 \cdot y^2 & 0 \leq y < \frac{1}{2} \\ -2 \cdot (1-y)^2 + 1 & \frac{1}{2} \leq y < 1 \\ 1 & y \geq 1 \end{cases}$$

entonces resolviendo la ecuación $U = F_2(Y)$, se obtiene:

$$y = \begin{cases} \sqrt{\frac{u}{2}} & u \in \left[0, \frac{1}{2}\right] \\ 1 - \sqrt{\frac{1-u}{2}} & u \in \left[\frac{1}{2}, 1\right] \end{cases}$$

El algoritmo queda,

Generar $U1, U2 \sim U(0,1)$
Si $u1 \leq t$, entonces hacer $X = U2$
en otro caso, Si $U2 < 1/2$, hacer $X = \sqrt(u/2)$
en otro caso hacer $x = 1\breve{}sqrt((1-u)/2)$
Devolver X

Nota:

En este ejemplo hemos trabajado con un tipo particular de función de densidad, las de tipo triangular. En general, si tenemos una variable aleatoria X, con función de densidad de tipo triangular su función de densidad $f(x)$ se puede descomponer como,

$$f(x) = \theta f_1(x) + (1-\theta)\frac{f(x) - \theta f_1(x)}{(1-\theta)}$$

luego,

- con probabilidad θ hay que simular $\mathcal{U}(0,1)$
- con probabilidad $(1-\theta)$ hay que simular $f_2(x) = \frac{f(x)-\theta f_1(x)}{(1-\theta)}$

según el valor de θ se elegirá este método u otro.

Ejemplo 12: Sea $X \sim Be(2,2)$, con función de densidad

$$f(X) = 6x(1-x) \quad x \in [0,1]$$

descomponemos f en

$$f(x) = t \cdot f_1(x) + (1-t)\frac{f(x) - t f_1(x)}{(1-t)}$$

donde

$$f_1(x) = \begin{cases} 4x & x \in [0, \frac{1}{2}] \\ 4(1-x) & x \in [\frac{1}{2}, 1] \end{cases}$$

entonces

$$f_2(x) = \frac{f(x) - t f_1(x)}{1-t} = \begin{cases} 12x(1-2x) & x \in [0, \frac{1}{2}] \\ 12(1-2x)(x-1) & x \in [\frac{1}{2}, 1] \end{cases}$$

Para calcular t sabemos que

$$t = min_x \frac{f(x)}{f_1(x)} = \begin{cases} \frac{6x(1-x)}{4x} & x \in [0, \frac{1}{2}] \\ \frac{6x(1-x)}{4(1-x)} & x \in [\frac{1}{2}, 1] \end{cases}$$

luego, $t = \frac{3}{4}$.

el algoritmo queda,

Si $U < 3/4$ generar un valor de f_1
otro si generar un valor de f_2

Tenemos que simular f_1 y f_2

Generar valores de f_1

Simulamos valores de f_1 que es una distribución triangular

vamos a utilizar el método de la inversa.

Obtenemos la función de distribución de f_1

$$F_1(x) = \begin{cases} 2x^2 & 0 \leq x \leq \frac{1}{2} \\ -1 + 4x - 2x^2 & 0 \leq x \leq \frac{1}{2} \end{cases}$$

Método de la inversa

Si $0 \leq x \leq 1/2 \Rightarrow F_1(0) = 0$ y $F_1(1/2) = 1/2$

entonces

$$0 \leq u \leq \frac{1}{2}\, u = x^2 \Rightarrow x = \sqrt{u}$$

Si $1/2 \leq x \leq 1 \Rightarrow F_1(1/2) = 1/2\, y\, F_1(1) = 1$

$$\frac{1}{2} \leq u \leq 1\, u = -14x - 2x^2 \Rightarrow x = 1 \pm \frac{1}{2}\sqrt{4 - 2(u+1)}$$

de las dos expresiones posibles de x hay que comprobar cuál verifica las condiciones.

Para $u = \frac{1}{2}$,

- tenemos que $x = 1 + \frac{1}{2}\sqrt{4 - 2(\frac{1}{2} + 1)} > 1$. No cumple las codiciones.
- tenemos que $x = 1 - \frac{1}{2}\sqrt{4 - 2(\frac{1}{2} + 1)} = 1/21$. Cumple las codiciones.

por lo tanto

$$x = \begin{cases} \sqrt{u} & 0 \leq u \leq \frac{1}{2} \\ 1 - \frac{1}{2}\sqrt{4 - 2(u+1)} & \frac{1}{2} \leq x \leq 1 \end{cases}$$

El algoritmo de generación de valores según

$$f_1$$

1.- Generar U ~U(0,1)
2.- Si U <= 1/2, hacer X = sqrt(U)
3.- Otro hacer X = 1 - (1/2)*sqrt{4 - 2(U+1)}
4.- Devolver X

En R,

```
# Genera valores de la f1
gen.f1 <- function(){
  U <- runif(1)
  if(U <= 0.5) X <- sqrt(U)
  else X <- 1 - 0.5*sqrt(4 - 2*(U+1))
  return(X)
}
```

Generar valores de f_2

Ahora hay que simular $f_2(x)$

vista su forma, elegimos el método del rechazo.

Buscamos una función de densidad $g(x)$ tal que

$$f_2(x) \leq c \cdot g(x) \quad \forall x \in [0,1]$$

La función que tomamos es $g(x) = 1 \quad \forall x \in [0,1]$ (la uniforme) y para determinar el escalar c, definimos la función $h(x) = f_2(x)/g(x)$

En este caso

$$c = \max_x h(x) = \max_x \frac{f_2(x)}{1} = \max_x f_2(x) \quad x \in [0,1]$$

el max $f_2(x) = 3/2 \Rightarrow c = 3/2$.

El algoritmo queda,

Mientras $U > f(X)/c * g(X)$
Mientras $U2 > 2 * f(X)/3$
Generar $X \sim g(x)$ Generar $U1 \sim U(0,1)$
Generar $U \sim U(0,1)$
Generar $U2 \sim U(0,1)$
Devolver X Devolver $U1$

con una eficiencia de 1.5.

En R,

```
# Generación variable X según f2
gen.f2 <- function(){
  U <- runif(2)
  while(U[2] > 2*f2*(U[1])/3){
    U <- runif(2)
    }
  return(U[1])
}
```

Generar valores de

X

Recordemos que el algoritmo era,

Si U $< 3/4$ generar un valor de f1
otro si generar un valor de f2

En R,

```
# Generación variable X
gen.x <- function(){
  U <- runif(1)
  if(U < 0.75) X <- gen.f1()
  else         X <- gen.f2()
  return(X)
}
```

por ejemplo

```
# Semilla
set.seed(544)

gen.x()
```

```
## [1] 0.508971
```

por ejemplo

2.3.2 Caso discreto

Dada una variable aleatoria discreta X con función de distribución F_X si existen n variables aleatorias con funciones de distribución $F_1, F_2, \ldots, F_n$ tal que,

$$F_X(x) = \alpha_1 F_1(x) + \alpha_2 F_2(x) + \cdots + \alpha_n F_n(x), \qquad \sum_{i=1}^{n} = 1, \alpha_i > 0$$

se puede generar valores de X haciendo:

Paso 1: Generar $I \sim U_D[1, n]$
Paso 2: Generar F_I

Ejemplo 11: Sea X la variable aleatoria con función de masa de probabilidad

X	1	2	3	4	5	6	7	8	9	10
$\Pr\{X = j\}$	0.05	0.05	0.05	0.05	0.15	0.15	0.15	0.15	0.15	0.15

Descomponemos X como una mixtura de dos variables aleatorias:

$$X_1: \quad p_j^{(1)} = 0.1, \qquad j = 1, \ldots, 10$$

$$X_2: \quad p_j^{(2)} = \begin{cases} 0 & j = 1, \ldots, 5 \\ 0.2 & j = 6, \ldots, 10 \end{cases}$$

luego, $X = 0.5X_1 + 0.5X_2$

El algoritmo queda,

1. Generar $U1 \sim U(0, 1)$
2. Generar $U2 \sim U(0, 1)$
3. Si $U1 < 0.5$, hacer $X = Int(10 * U2) + 1$
4. Otro $X = Int(5 * U2) + 6$
5. Devolver X

Ejemplo 12: Método de composición.

Sea la VA X con función de masa de probabilidad,

$$\Pr(X = j) = p_j = \begin{cases} 0.05 & \text{para} j = 1, 2, 3, 4, 5 \\ 0.15 & \text{para} j = 6, 7, 8, 9, 10 \end{cases}$$

Se pide generar una muestra aleatoria de X.

Primero vemos que,

$0.05 \times 5 = 0.25$

$0.15 \times 5 = 0.75$

definimos las variables aleatorías, Y_1 y Y_2:

$Y_1 : \{p_j = 0.05/0.25 j = 1, 2, 3, 4, 5\}$

$Y_2 : \{p_j = 0.15/0.75 j = 6, 7, 8, 9, 10\}$

de esta forma podemos expresar X como una mixtura de Y_1 y Y_2, $X = 0.25Y_1 + 0.75Y_2$

El algoritmo de composición es,

```
Generar U ~ U(0,1)
Si      U < 0.75 entonces X = Generar Y1
Otrosi                   X = Generar Y2
```

En R,

```
# Generar valores de Y1
V <- runif(1)
Y1 <- floor(5*V) + 1
Y1
```

```
## [1] 1
```

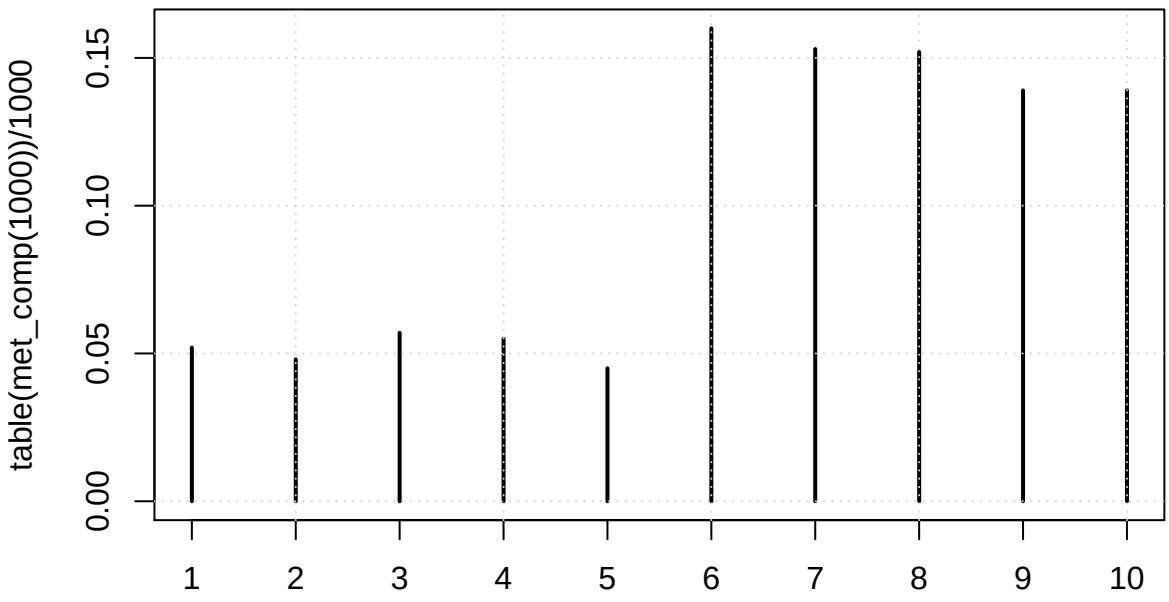
```
# Generar valores de Y2
V <- runif(1)
Y2 <- floor(5*V) + 6
Y2
```

```
## [1] 7
```

```
U <- runif(1)
if(U < 0.75){
  V <- runif(1)
  X <- floor(5*V) + 6
} else {
  V <- runif(1)
  X <- floor(5*V) + 1
  }
```

En forma de función

```
met_comp <- function(n){
  # Entrada
  # n - cantidad de números aleatorios
  # Salida
  # vec - vector con la muestra aleatoria
  U <- runif(n)
  V <- runif(n)
  vec <- ifelse(U < 0.75, floor(5*V) + 6, floor(5*V) + 1)
  vec
}

plot(table(met_comp(1000))/1000)
grid()
```


table(met_comp(1000))/1000
0.00
0.05
0.10
0.15
1
2
3
4
5
6
7
8
9
10

2.3.3 Ejercicios