Generación de variables aleatorias continuas

Se dice que una variable aleatoria X es continua, o más propiamente, absolutamente continua, si existe $f : \mathbb{R} \mapsto \mathbb{R}$ tal que

$$F(x) := P(X \leq x) = \int_{-\infty}^{x} f(t)\, dt.$$

Al igual que para las v.a. discretas, analizaremos dos métodos de generación para variables continuas, con las respectivas mejoras en cada caso particular. Estos métodos también se denominan:

- Método de la transformada inversa.
- Método de aceptación y rechazo.

1. Método de la transformada inversa

La generación o simulación de valores de una v.a. por el método de la transformada inversa, requiere conocer la función de distribución acumulada de la variable X a simular. Si X es (absolutamente) continua, entonces existe $f : \mathbb{R} \mapsto \mathbb{R}$ tal que su función de distribución está dada por:

$$F(x) = \int_{-\infty}^{x} f(t)\, dt,$$

y f se denomina la **función de densidad** de X.

La función F es continua, no decreciente, y se cumple que:

$$\lim_{x \to -\infty} F(x) = 0, \qquad \lim_{x \to \infty} F(x) = 1.$$

En esta sección nos referiremos a variables aleatorias X cuya función de distribución sea monótona creciente sobre el conjunto $F^{-1}(0,1) = \{x \mid 0 < F(x) < 1\}$. De esta manera, la función F tiene inversa sobre el intervalo $(0,1)$.

Para los casos en que no se cumple esta propiedad también es posible definir métodos de simulación de las variables, pero las demostraciones de su validez son un tanto más complejas.

Proposición 1. Sea $U \sim \mathcal{U}(0,1)$ una variable aleatoria. Para cualquier función de distribución continua F, monótona creciente en $F^{-1}(0,1)$, la variable aleatoria X definida por:

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

tiene distribución F.

Notemos que U toma valores en el intervalo $(0,1)$, excluyendo los puntos 0 y 1. La propiedad de monotonía de F asegura que F tiene inversa en $(0,1)$.

Así, si $X = F^{-1}(U)$, su función de distribución está dada por:

$$F_X(x) = P(X \leq x) = P(F^{-1}(U) \leq x).$$

Como F es creciente, entonces $a \leq b$ si y sólo si $F(a) \leq F(b)$. Tomando $a = F^{-1}(U)$ y $b = x$, resulta:

$$F_X(x) = P(F(F^{-1}(U)) \leq F(x)) = P(U \leq F(x)) = F(x).$$

Luego X tiene distribución F.

Esta proposición da un método en principio simple para simular una variable aletoria continua:

```
1 def Tinversa(F):
2     U=random()
3     return G(U)  # G = F^-1
```

Sin embargo, pueden existir algunas complicaciones. Por ejemplo, que la inversa de F involucre funciones computacionalmente costosas. ($\sqrt[n]{f(x)}$, $\log(x)$, etc.), o que no pueda ser calculada explícitamente (p. ej., distribución de la normal, de una gamma).

Así, para ciertas distribuciones deben utilizarse otras estrategias, por ejemplo expresando a F como

- distribución del mínimo y/o del máximo de v. a. independientes,
- distribución de suma de variables aleatorias independientes,
- distribución de una v. a. condicional a otra,
- u otros métodos específicos.

Ejemplo 1.1. Escribir un método para generar el valor de una v. a. X con función de densidad

$$f(x) = -\frac{x}{2} + 1, \qquad 0 \leq x \leq 2.$$

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

F resulta entonces monótona creciente en el intervalo $(0, 2)$.

Para simular X debemos determinar la inversa de F en $(0, 1)$. Es decir, para cada $u \in (0, 1)$, encontrar el valor x tal que

$$-\frac{x^2}{4} + x = u.$$

Esta ecuación tiene dos soluciones para x:

$$\begin{cases} x = 2 + 2\sqrt{1-u} \\ \text{o} \\ x = 2 - 2\sqrt{1-u} \end{cases}$$

pero sólo la segunda pertenece al intervalo $(0, 2)$. Luego el algoritmo de generación será:

```
def inversaX():
    U=random()
    return 2-2*sqrt(U)
```

Ejemplo 1.2. Escribir un método para generar el valor de una v. a. Y con función de densidad

$$f(x) = \begin{cases} 0.25 & 0 \leq x \leq 2 \\ 0.5 & 2 < x < 3 \\ 0 & \text{c.c.} \end{cases}$$

Para esta variable aleatoria, la función de distribución está dada por:

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

Notemos que Y toma valores en el intervalo $(0, 3)$, y que $F(2) = 0.5$. Luego el algoritmo de simulación de la variable es:

```
def inversaY():
    U=random()
    if U<0.5:
        return 4*U
    else:
        return 2U+1
```

Ejemplo 1.3. Consideremos $X_1, X_2, \ldots, X_n$ n variables aleatorias independientes, con funciones de distribución $F_1, F_2, \ldots, F_n$, respectivamente:

$$F_i(a) = P(X_i \leq a).$$

Llamamos X a la variable aleatoria que es el máximo de las X_i, $1 \leq i \leq n$:

$$X = \text{máx}\{X_1, X_2, \ldots, X_n\}.$$

Entonces la función de distribución de X está dada por:

$$F_X(a) = F_1(a) \cdot F_2(a) \ldots F_n(a).$$

En particular, si X_1, X_2, …, X_n son uniformes en $(0, 1)$, independientes entre sí, tenemos que $F_i(x) = x$, $0 \leq x \leq 1$ e $1 \leq i \leq n$. Luego el máximo de las X_i tiene una distribución dada por:

$$F_X(x) = x^n, \qquad 0 \leq x \leq 1.$$

Así, si se desea simular una variable aleatoria con distribución $F_X(x) = x^n$, en lugar de tomar una raíz n-ésima es conveniente generar n uniformes y tomar su máximo.

```
def raizn(n):
    Maximo=0
    for i in range(n):
        U=random()
        If Maximo<U:
            Maximo=U
    return Maximo
```

Ejemplo 1.4. Si X es una variable exponencial, $X \sim \mathcal{E}(1)$, entonces su función de distribución está dada por:

$$F(x) = \begin{cases} 1 - e^{-x} & x > 0 \\ 0 & x \leq 0. \end{cases}$$

Luego, si $u \in (0,1)$, entonces $F(x) = u$ si y sólo si $1 - e^{-x} = u$, es decir:

$$x = -\ln(1-u).$$

Dado que si $U \sim U(0,1)$ entonces $(1-U)$ también es uniforme en $(0,1)$, el algoritmo por el método de la transformada inversa es:

```
def exponencial():
    U=random()
    return -log(U)
```

Notemos además que si X es exponencial con parámetro 1, esto es, $X \sim \mathcal{E}(1)$, entonces $\frac{1}{\lambda}X \sim \mathcal{E}(\lambda)$. Luego el método para generar $Y \sim \mathcal{E}(\lambda)$ es

```
def exponencial(lambda):
    U=random()
    return -log(U)/lambda
```

1.1. Generación de una v. a. Poisson $X \sim \mathcal{P}(\lambda)$

Si $P(t)$, $t \geq 0$ es un proceso de Poisson homogéneo de parámetro λ, entonces sabemos que:

- los tiempos de llegada entre eventos son v. a. exponenciales de parámetro λ.
- el número de eventos en un intervalo de tiempo de longitud t es una v. a. Poisson de media $\lambda \cdot t$.

En particular, $N(1)$ es una v. a. Poisson de media λ que indica el número de arribos hasta el tiempo $t = 1$, y los tiempos entre arribos en el intervalo $[0,1]$ son exponenciales de parámetro λ.

Por lo tanto, si se generan variables exponenciales $X_i \sim \mathcal{E}(\lambda)$, $i \geq 1$, hasta que $X_1 + X_2 + \cdots + X_n \leq 1$ y $X_1 + X_2 + \cdots + X_n + X_{n+1} > 1$, entonces n representa el número de arribos hasta $t = 1$. Esto es:

$$N(1) = \text{máx}\{n \mid X_1 + X_2 + \cdots + X_n \leq 1\}$$

Empleando v.a. uniformes para generar las exponenciales, tenemos que

$$\begin{aligned} N(1) &= \text{máx}\{n \mid X_1 + X_2 + \cdots + X_n \leq 1\} \\ &= \text{máx}\{n \mid -\frac{1}{\lambda}\left(\log(U_1) + \log(U_2) + \cdots + \log(U_n)\right) \leq 1\} \\ &= \text{máx}\{n \mid -\frac{1}{\lambda}\left(\log(U_1 \cdot U_2 \cdots U_n)\right) \leq 1\} \\ &= \text{máx}\{n \mid \log\left(U_1 \cdot U_2 \cdots U_n\right) \geq -\lambda\} \\ &= \text{máx}\{n \mid U_1 \cdot U_2 \cdots U_n \geq e^{-\lambda}\} \end{aligned}$$

Luego:

$$N(1) = \text{mín}\{n \mid U_1 \cdot U_2 \cdots U_n < e^{-\lambda}\} - 1$$

1.2. Generación de una variable aleatoria con distribución Gamma(n,λ)

En otras secciones hemos visto que la suma de n variables aleatorias exponenciales, independientes, con parámetro λ, es una variable aleatoria Xcon distribución $Gamma(n, \lambda)$. Esta propiedad nos permite dar un algoritmo de generación de X. Notemos que

$$\begin{aligned} X &= -\frac{1}{\lambda}\log(U_1) - \frac{1}{\lambda}\log(U_2)\cdots - \frac{1}{\lambda}\log(U_n) \\ &= -\frac{1}{\lambda}\log\left(U_1 \cdot U_2 \cdots U_n\right) \end{aligned}$$

```
def Gamma(n,lambda):
    U=1
    for _ in range(n):
        U*=random()
    return -log(U)/lambda
```

Notemos que este método utiliza n variables uniformes y calcula un único logaritmo, mientras que, para generar n exponenciales es necesario calcular n logaritmos. Veamos cómo puede utilizarse la generación de una $Gamma(n, \lambda)$ para generar n exponenciales independientes de parámetro λ.

Teorema 1.1. Si $X, Y \sim \mathcal{E}(\lambda)$, independientes, entonces

$$f_{X|X+Y}(x \mid t) = \frac{1}{t}\mathbb{I}_{(0,t)}(x),$$

es decir, X condicional a $X + Y = t$ es uniforme en $(0, t)$.

La demostración de este teorema y del caso general para n exponenciales está disponible en el material complementario.

Luego, para generar X, Y exponenciales independientes, de parámetro λ podemos aplicar el siguiente algoritmo:

```
def DosExponenciales(lambda):
    t= -1/lambda * log(random() * random()) # valor de una Gamma(2,lambda)
    U=random()
    X=t*U
    Y=t-X
    return (X,Y)
```

Este algoritmo requiere el cálculo de un único logaritmo, y tres uniformes.

En el caso general, para simular n exponenciales a partir de una $X \sim Gamma(n, \lambda)$ se requiere calcular un único logaritmo y $n-1$ uniformes adicionales: $V_1, V_2, \ldots, V_{n-1}$. En primer lugar se ordenan los valores de las V_i:

$$V_{i_1} < V_{i_2} < \ldots < V_{i_{n-1}}.$$

Si $X = t$, entonces el intervalo $(0, t)$ se divide en n subintervalos no superpuestos, de longitud:

$$tV_{i_1}, \qquad t\,(V_{i_2} - V_{i_1}), \qquad t\,(V_{i_{n-1}} - V_{i_{n-2}}), \quad t(1 - V_{i_{n-1}}).$$

Cada uno de estos valores tiene distribución exponencial $\mathcal{E}(\lambda)$.

N EXPONENCIALES(λ)

1 $X = \log(U_1 \cdot U_2 \cdots U_n)$ # $U_i = random()$
2 $t = -\frac{1}{\lambda} X$
3 Generar $V_1, V_2, \ldots, V_{n-1} \sim U(0, 1)$
4 Ordenarlos de menor a mayor
5 **return** $[t\,V_1, t\,(V_2 - V_1), \ldots, t\,(V_{n-1} - V_{n-2}), t - t\,V_{n-1}]$

---

2. El método de aceptación y rechazo

Supongamos que se quiere generar una variable aleatoria X con función de densidad f:

$$F(x) = P(X \leq x) = \int_{-\infty}^{x} f(t)\,dt,$$

y que se tiene un método para generar otra variable Y, con densidad g, tal que

$$\frac{f(y)}{g(y)} \leq c, \qquad \text{para todo } y \in \mathbb{R} \text{ tal que } f(y) \neq 0.$$

El **método de rechazo** para generar X a partir de Y tiene el siguiente algoritmo:

MÉTODO DE RECHAZO

1 **repeat**
2 Generar Y, con densidad g
3 $U = random()$
4 **until** $U < f(Y)/(cg(Y))$
5 **return** Y

El siguiente teorema establece que el MÉTODO DE RECHAZO es eficaz para generar X con densidad f:

Teorema 2.1. 1. La variable aleatoria generada por el MÉTODO DE RECHAZO tiene densidad f.

2. El número de iteraciones del algoritmo es una variable aleatoria geométrica con media c.

Este algoritmo requiere determinar una cota c para el cociente $\frac{f(x)}{g(x)}$, válida para todo $x \in \mathbb{R}$. Para determinar este valor es útil recordar algunos resultados del Análisis Matemático.

En primer lugar, considerar la función

$$h(x) = \frac{f(x)}{g(x)}, \qquad x \text{ tal que } f(x) \neq 0,$$

y para esta función determinar:

1. Puntos críticos: Esto es, $f'(x) = 0$ o $f'(x)$ no existe.

2. Analizar cuáles de estos puntos corresponden a máximos locales.

3. Evaluar h en los extremos de su dominio, si es acotado, o los límites correspondientes.

Ejemplo 2.1. Se quiere generar una variable aleatoria X con función de densidad:

$$f(x) = 20x(1-x)^3, \qquad 0 < x < 1.$$

En particular, X tiene distribución $Beta(2, 4)$:

$$\boxed{f(x) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1}\mathbb{I}_{(0,1)}(x) \qquad \text{Variable } \beta(2,4)}$$

Dado que $f(x) \neq 0$ en el intervalo $(0, 1)$, se podría utilizar el método de rechazo generando una variable $Y \sim \mathcal{U}(0, 1)$. Tenemos entonces que

$$f_X(x) = 20x(1-x)^3, \qquad f_Y(x) = 1, \qquad x \in (0,1),$$

y $h(x) = \frac{f_X(x)}{f_Y(x)} = f_X(x)$.

Los pasos para analizar una cota de h son los siguientes:

$$\begin{aligned} h(x) &= 20x(1-x)^3, \qquad 0 < x < 1 \\ h'(x) &= 20(1-x)^2 \cdot (1-4x) \end{aligned}$$

- Puntos críticos: $x = 1$ y $x = 1/4$.

- $f(1) = 0$, luego $x = 1$ no es un máximo.
- $x = 1/4$ es el otro punto crítico, luego es el máximo por ser $h \geq 0$.
- $h(1/4) = f(1/4) = 135/64$ es el valor máximo de h

$$c = \frac{135}{64} = 2.109375$$

El algoritmo utilizando el MÉTODO DE RECHAZO resulta:

BETA(2,4)

1 **repeat**
2 $\quad Y = random()$
3 $\quad U = random()$
4 **until** $U < \dfrac{256 \cdot Y \cdot (1-Y)^3}{27}$
5 **return** Y

Este método requiere un número promedio de ciclos del orden de $c = \dfrac{135}{64} \approx 2.11$.

Ejemplo 2.2. Veamos cómo es posible generar una variable aleatoria X, con densidad $Gamma(\frac{3}{2}, 1)$:

$$f(x) = \begin{cases} Kx^{1/2}e^{-x} & x > 0 \\ 0 & c.c. \end{cases}, \qquad K = \frac{1}{\Gamma(\frac{3}{2})} = \frac{2}{\sqrt{\pi}}.$$

En general, si $X \sim \Gamma(\alpha, \beta)$, su función de densidad está dada por

$$g(x) = \frac{\beta^\alpha}{\Gamma(\alpha)} e^{-\beta x} x^{\alpha-1}, \qquad x > 0,$$

y su valor esperado es

$$E[X] = \frac{\alpha}{\beta}.$$

En este caso, $\alpha = \frac{3}{2}$ y $\beta = 1$, por lo cual $E[X] = \frac{3}{2}$. Dado que una variable exponencial tiene el mismo rango que X, una posibilidad entonces es utilizar el método de rechazo con una variable exponencial $Y \sim \mathcal{E}(\lambda)$ que tenga igual media que X, es decir, $Y \sim \mathcal{E}(\frac{2}{3})$.

El método de rechazo implica calcular la constante c tal que $\frac{f_X(x)}{f_Y(x)} \leq c$, si $x > 0$. Esto es, debemos hallar una cota superior para la función:

$$h(x) = \frac{Kx^{1/2}e^{-x}}{\frac{2}{3}\, e^{-\frac{2}{3}x}} = K\,\frac{3}{2}\, x^{1/2}e^{-\frac{1}{3}x}, \qquad x > 0.$$

Tenemos que $h'(x) = 0$ si y sólo si

$$(\frac{1}{2} - \frac{1}{3}x)x^{-\frac{1}{2}}e^{-\frac{1}{3}x} = 0$$

Dado que $\lim_{x\to\infty} h(x) = 0$ y $\lim_{x\to 0^+} h(x) = 0$, el valor máximo estará dado en algún punto crítico. En este caso, el valor de x tal que

$$\frac{1}{2} - \frac{1}{3}x = 0, \qquad x = \frac{3}{2}.$$

Para este punto crítico, el valor máximo alcanzado por la función es:

$$c = 3\left(\frac{3}{2\pi e}\right)^{1/2} \approx 1.257$$

Así el algoritmo resulta:

GAMMA$(3/2, 1)$

1 **repeat**
2 $\quad Y = -\frac{2}{3} \cdot \log(random())$
3 $\quad U = random()$
4 **until** $U < \left(\frac{2e}{3}\right)^{1/2} Y^{1/2}e^{-Y/3}$
5 **return** Y

Podríamos preguntarnos en el Ejemplo 2.2 si es razonable rechazar con una exponencial de igual media que la Gamma a generar. Para ello, deberíamos encontrar una cota de:

$$\frac{f(x)}{\lambda e^{-\lambda x}}$$

y determinar el valor de λ para la cual la cota es mínima. Notemos que:

$$\frac{f(x)}{\lambda e^{-\lambda x}} = \frac{Kx^{1/2}e^{-(1-\lambda)x}}{\lambda},$$

y esta función es no acotada si $\lambda \geq 1$. Luego corresponde analizar los casos en que $0 < \lambda < 1$.

Analizando puntos críticos, podemos ver que el punto de máximo está en:

$$x = \frac{1}{2(1-\lambda)}, \qquad 0 < \lambda < 1.$$

y el valor máximo es igual a

$$c_\lambda = \frac{K}{\lambda}(2(1-\lambda))^{-1/2}e^{-1/2}.$$

El valor máximo c_λ será mínimo si $\lambda(1-\lambda)^{1/2}$ es máximo, y esto ocurre si $\lambda = \frac{2}{3}$. Por lo tanto $\lambda = \frac{2}{3}$ minimiza el valor de la cota c.

3. Generación de variables aleatorias normales

Las variables aleatorias normales tienen un uso extensivo, por lo cual es importante obtener un buen método para simularlas. En primer lugar, recordemos que si $Z \sim N(0,1)$, entonces

$$X = \sigma \cdot Z + \mu \sim N(\mu, \sigma).$$

Por lo tanto es suficiente tener un buen método para generar variables normales estándar.

El **método de la transformada inversa** no es una opción, puesto que la función de distribución acumulada:

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

no tiene una expresión cerrada y por lo tanto no se conoce una forma explícita de su inversa.

Una observación es que la función de densidad de Z es par, por lo cual puede intentarse un método para generar $|Z|$ y usar luego un método de composición:

$$F_Z(x) = 0.5 \cdot F_{|Z|}(x) + 0.5 \cdot F_{-|Z|}(x),$$

con el consiguiente algoritmo:

NORMAL-MÉTODO1

1 $U = random()$
2 **if** $U < 0.5$
3 **return** $|Z|$
4 **else**
5 **return** $-|Z|$

Es necesario entonces un método para generar $|Z|$. Su densidad está dada por:

$$f_{|Z|}(x) = \begin{cases} \frac{2}{\sqrt{2\pi}} e^{-x^2/2} & x > 0 \\ 0 & c.c. \end{cases}$$

Una posibilidad es utilizar el **método de rechazo** con una exponencial, por ejemplo, $X = \mathcal{E}(1)$. Debemos encontrar una cota para:

$$\frac{f_{|Z|}(x)}{e^{-x}} = \frac{2}{\sqrt{2\pi}}\, e^{-\frac{x^2}{2}+x}.$$

Este máximo corresponde al valor máximo del exponente de la exponencial, que ocurre en $x = 1$. El valor máximo y por consiguiente la cota mínima c está dada por:

$$c = \sqrt{\frac{2e}{\pi}} \approx 1.32$$

Para describir el algoritmo, calculamos:

$$\frac{f(x)}{cg(x)} = \exp\left\{-\frac{(x-1)^2}{2}\right\}.$$

Luego el método de rechazo simula $Y_1 \sim \mathcal{E}(1)$, U uniforme, e itera hasta que:

$$U \leq \exp\left\{-\frac{(Y_1-1)^2}{2}\right\},$$

es decir:

$$\log(U) \leq -\frac{(Y_1-1)^2}{2} \qquad \text{o equivalentemente} \qquad -\log(U) \geq \frac{(Y_1-1)^2}{2}.$$

Dado que $Y_2 = -\log(U)$ genera una exponencial $\mathcal{E}(1)$, el algoritmo por el método de rechazo **para generar** Z se traduce en:

```
def Normal(mu,sigma):
    while True:
        Y1=-log(random)
        Y2=-log(random)
        if Y2 >=(Y1-1)**2/2:
            break
    if random()<0.5:
        return Y1*sigma + mu
    else:
        return -Y1*sigma + mu
```

Recordemos que una variable con distribución exponencial posee la propiedad de **falta de memoria**:

$$P(X > s+t \mid X > t) = P(X > s).$$

Es decir, condicional a que la variable X sea mayor a t, la cantidad que excede a este valor también tiene distribución exponencial. En particular, en el paso del algoritmo amterior en que se acepta el valor de Y_1, ocurre que Y_2 supera a $(Y_1-1)^2/2$ y por lo tanto la cantidad que excede:

$$Y_2 - (Y_1-1)^2/2$$

tiene distribución exponencial $\mathcal{E}(1)$.

De esta manera, el método de rechazo para generar $Z \sim N(0,1)$ permite generar también una exponencial. Esta exponencial podría ser utilizada en pasos sucesivos como el primer valor de Y_1, y así disminuir en 1 el número de exponenciales generadas. Como el método tiene un promedio de $c \sim 1.32$ iteraciones, el número de exponenciales necesarias será del orden de

$$2 \cdot 1.32 - 1 = 1.64.$$

3.1. Método polar

Otro método eficiente para generar variables normales es el **método polar**. El nombre se refiere al uso de coordenadas polares para referenciar puntos del plano:

$$x = r\cos\theta, \qquad y = r\,\mathrm{sen}\,\theta, \quad r \geq 0, \quad , 0 \leq \theta < 2\pi. \tag{1}$$

Consideramos dos variables aleatorias X e Y, normales estándar, independientes. Dado que toman cualquier valor real, entonces el par (X, Y) denota a algún punto del plano que podría ser cualquiera. Las coordenadas polares correspondientes a este punto, R y Θ, verifican:

$$R^2 = X^2 + Y^2, \qquad \Theta = \arctan\frac{Y}{X}.$$

La función de densidad conjunta de X e Y está dada por:

$$f_{X,Y}(x, y) = f_X(x) \cdot f_Y(y) = \frac{1}{2\pi}\, e^{-(x^2+y^2)}.$$

La transformación de coordenadas (r^2, θ) a (x, y) es:

$$r^2 = x^2 + y^2, \qquad \theta = \arctan\frac{y}{x},$$

y por lo tanto la matriz jacobiana está dada por:

$$\begin{pmatrix} \frac{\partial}{\partial x}(x^2+y^2) & \frac{\partial}{\partial y}(x^2+y^2) \\ \frac{\partial}{\partial x}(\arctan\frac{y}{x}) & \frac{\partial}{\partial \theta}(\arctan\frac{y}{x}) \end{pmatrix} = \begin{pmatrix} 2x & 2y \\ -\frac{y}{x^2+y^2} & \frac{x}{x^2+y^2} \end{pmatrix}$$

con determinante jacobiano igual a 2. Por lo tanto, la densidad conjunta de R^2 y Θ satisface:

$$f_{X,Y}(x, y) = 2\, f_{R^2,\Theta}(d, \theta), \qquad x^2 + y^2 = d,\ \tan(\theta) = y/x.$$

Así, formalmente resulta:

$$\begin{aligned} f_{R^2,\Theta}(d, \theta) &= \frac{1}{2}\,\frac{1}{2\pi} e^{-d/2}\, \mathbb{I}_{[0,\infty)}(d)\mathbb{I}_{[0,2\pi)}(\theta) \\ &= \underbrace{\frac{1}{2\pi}\mathbb{I}_{[0,2\pi)}(\theta)}_{\Theta\sim U(0,2\pi)} \cdot \underbrace{e^{-d/2}\,\frac{1}{2}\,\mathbb{I}_{[0,\infty)}(d)}_{R^2\sim\mathcal{E}(\frac{1}{2})} \end{aligned}$$

Es decir, R^2 y Θ resultan ser variables aleatorias independientes, con distribución exponencial $\mathcal{E}(\frac{1}{2})$ y uniforme $\mathcal{U}(0, 2\pi)$, respectivamente. El **Método polar** genera dos variables normales estándar, y el algoritmo es como sigue:

```
def DosNormales():
    Rcuadrado=-2*log(random())
    Theta= 2*Pi*random()
    X= sqrt(Rcuadrado)*cos(Theta)
    Y= sqrt(Rcuadrado)*sen(Theta)
    return (X,Y)
```

Las transformaciones:

$$\begin{cases} X = \sqrt{-2\log(U_1)}\cos(2\pi U_2) \\ Y = \sqrt{-2\log(U_1)}\,\mathrm{sen}(2\pi U_2) \end{cases} \tag{2}$$

se denominan **transformaciones de Box-Muller**. Una desventaja de estas transformacioneses que requieren el cálculo de dos funciones trigonométricas: seno y coseno. Para mejorar este paso, notemos que si (X, Y) son las coordenadas de un punto aleatorio en el círculo unitario, entonces:

$$P(X^2 + Y^2 \leq r) = P(\sqrt{X^2 + Y^2} \leq \sqrt{r}) = \frac{\pi\, r^2}{\pi} = r, \qquad 0 < r < 1,$$
$$P(0 < \arctan\frac{Y}{X} < \alpha) = \frac{\alpha/2}{\pi} = \frac{1}{2\pi}\,\alpha, \qquad 0 \leq \alpha < 2\pi.$$

Así, $S^2 = X^2 + Y^2$ y $\Theta = \arctan\frac{Y}{X}$ resultan variables independientes, uniformemente distribuidas en $[0, 1)$ y $[0, 2\pi)$ respectivamente. Además,

$$\cos\Theta = \frac{X}{\sqrt{X^2+Y^2}} = \frac{X}{\sqrt{S^2}}, \qquad \mathrm{sen}\,\Theta = \frac{Y}{\sqrt{X^2+Y^2}} = \frac{Y}{\sqrt{S^2}}. \tag{3}$$

En resumen, para generar $\cos\Theta$ y $\mathrm{sen}\,\Theta$ debemos:

1. Generar un punto aleatorio (X, Y) del círculo unitario.
2. Calcular $\cos\Theta$ y $\mathrm{sen}\,\Theta$ usando (3).

La generación de puntos aleatorios en el círculo se reduce a generar pares de puntos (V_1, V_2) en el cuadrado unitario hasta obtener uno en el círculo unitario. Las **transformaciones de Box-Muller** se reescriben entonces como:

$$X = \sqrt{-2\log U}\,\frac{V_1}{\sqrt{(V_1^2 + V_2^2)}} \tag{4}$$
$$Y = \sqrt{-2\log U}\,\frac{V_2}{\sqrt{(V_1^2 + V_2^2)}} \tag{5}$$

Ahora, como $S^2 = V_1^2 + V_2^2$ es uniforme en $(0, 1)$ y es independiente de Θ, puede ser utilizado como U en (4) y (5), resultando las ecuaciones:

$$X = \sqrt{-2\log U} \cdot \frac{V_1}{\sqrt{U}} = V_1 \cdot \sqrt{\frac{-2\log U}{U}} \tag{6}$$

$$Y = \sqrt{-2\log U} \cdot \frac{V_2}{\sqrt{U}} = V_2 \cdot \sqrt{\frac{-2\log U}{U}} \tag{7}$$

Finalmente, el método polar para generar dos variables normales resulta:

```
def DosNormales():
    #Generar un punto aleatorio en el círculo unitario.
    while True:
        V1, V2 = random(), random()
        if V1**2+V2**2<=1:
            break
    S=V1**2+V2**2
    X=V1*sqrt(-2*log(S)/S)
    Y=V2*sqrt(-2*log(S)/S)
    return (X,Y)
```

4. Generación de un Proceso de Poisson

En un proceso de Poisson homogéneo de razón λ, los tiempos de llegada entre eventos sucesivos son exponenciales con razón λ (media $1/\lambda$). Así, para generar los primeros n eventos de un Proceso de Poisson homogéneo, generamos exponenciales $X_i \sim \mathcal{E}(\lambda)$, $1 \leq i \leq n$:

- Primer evento: al tiempo X_1.
- j-ésimo evento: al tiempo $X_1 + X_2 + \cdots + X_j$, para $1 \leq j \leq n$.

Para generar los eventos en las primeras T unidades de tiempo, generamos eventos hasta que $j + 1$ excede a T. El siguiente algoritmo denota I al número de eventos que ocurren hasta el tiempo T, $S[I]$ indica el tiempo en que ocurre cada evento, $1 \leq I \leq n$ (n será el máximo valor alcanzado por I):

```
def eventosPoisson(T):
    t=0
    I=0
    S[0]=0
    while True:
        U=random()
        if t-log (U)/lambda >T:
            break
        else:
            t= t-t-log (U)/lambda
        I+=1
        S[I]=t
```

Ahora bien, sabemos que $N(T)$, el número de eventos hasta el tiempo T, es una variable aleatoria con distribución Poisson $\mathcal{P}(\lambda T)$. Además, conocido $N(T)$, los tiempos de arribo se distribuyen uniformemente en $(0, T)$:

Proposición 2. Dado $N(T)$, la distribución de los tiempos de arribo en un Proceso de Poisson homogéneo de razón λ es uniforme en $(0, T)$.

Por lo tanto, un método alternativo para generar los tiempos de arribo hasta el tiempo T consiste en:

- Generar una variable aleatoria Poisson de media λT, y tomar $n = N(T)$.
- Generar n variables aleatorias uniformes $U_1, U_2, \ldots, U_n$.
- Ordenarlas: $U_{i_1} < U_{i_2} < \ldots < U_{i_n}$.
- Los tiempos de arribo son: $TU_{i_1}, TU_{i_2}, \ldots, TU_{i_n}$.

Si bien este algoritmo posee la ventaja de no generar una secuencia de exponenciales, requiere realizar un ordenamiento de $n = N(T)$ números.

5. Generación de un proceso de Poisson no homogéneo

Consideremos un proceso de Poisson no homogéneo con función de intensidad $\lambda(t)$, y sea λ tal que

$$\lambda(t) \leq \lambda.$$

Si ahora $N_1(t)$ es un proceso homogéneo, de tasa λ, y consideramos $N(t)$ el proceso que cuenta los eventos de $N_1(t)$ con probabilidad $\frac{\lambda(t)}{\lambda}$, entonces $N(t)$ verifica:

- $N(0) = 0$, pues $N_1(0) = 0$.
- Los incrementos son independientes, pues los incrementos de N_1 lo son.
-

$$\lim_{h\to 0} \frac{P(N(t+h) - N(t) = 1)}{h} = \lim_{h\to 0} \frac{P(N_1(t+h) - N_1(t) = 1, U \leq \frac{\lambda(t)}{\lambda}}{h}$$
$$= \lambda \cdot \frac{\lambda(t)}{\lambda} = \lambda(t).$$

$$\lim_{h\to 0} \frac{P(N(t+h) - N(t) \geq 2)}{h} = \lim_{h\to 0} \frac{P(N_1(t+h) - N_1(t) \geq 2, U \leq \frac{\lambda(t)}{\lambda}}{h} = 0$$

Luego $N(t)$ es un proceso de Poisson no homogéneo con función de intensidad $\lambda(t)$. Por lo tanto, estas propiedades nos permiten deducir el siguiente algoritmo de generación de un proceso de Poisson no homogéneo con función de intensidad $\lambda(t)$:

PROCESO POISSON NO HOMOGÉNEO($\lambda(t)$)

```
1  t = 0
2  I = 0
3  while True
4      U = random()
5      if t − (1/λ) log U > T
6          break
7      else
8          t = t − (1/λ) log U
9      V = random()
10     if V < λ(t)/λ
11         I = I + 1
12         S[I] = t
```

Nuevamente, I finaliza con el número de eventos hasta el tiempo T, es decir, $N(T)$, y $S[0]$, $S[1]$, ..., $S[T]$ son los tiempos de cada evento.

Notemos que el algoritmo es más eficiente cuánto más cerca esté λ de $\lambda(t)$, ya que cuanto más grande sea λ más iteraciones tendrá el método. Una mejora de este algoritmo es particionar el intervalo $(0, T)$ en subintervalos, y aplicar el algoritmo anterior en cada uno de ellos con un valor λ_i adecuado. Esto es, considerar k intervalos consecutivos $[t_{i-1}, t_i)$, $1 \leq i \leq k$ tales que:

$$0 = t_0 < t_1 < \ldots < t_k < t_k = T,$$

y valores $\lambda_1, \lambda_2, \ldots, \lambda_k$ que cumplan

$$\lambda(s) \leq \lambda_i, \qquad s \in [t_{i-1}, t_i).$$

El método resulta entonces:

```
1  t = 0   # acumula tiempos
2  J = 1   # recorre intervalos
3  I = 0   # acumula cantidad de arribos
4  while True
5      U = random()
6      X = -(1/λ_J) log U
7      if t + X > t_J
8          if J = k + 1
9              break
10         else
11             X = (X - t_J + t)(λ_J / λ_{J+1})
12             t ← t_J
13             J = J + 1
14     t = t + X
15     V = random()
16     if V < λ(t)/λ_J
17         I ← I + 1
18         S[I] = t
```