Práctica 10: Simulación con R

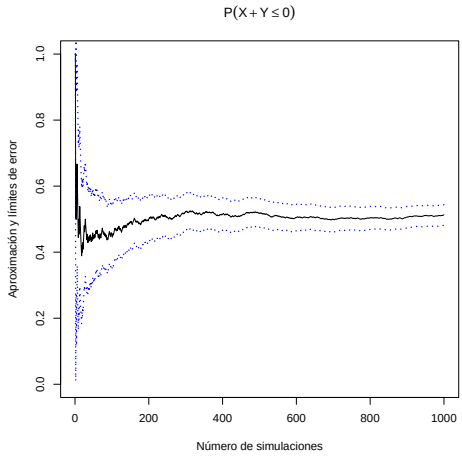
A continuación se proponen varias tareas a realizar en R sobre los contenidos del Tema 5. Las soluciones detalladas (incluyendo código, resultados de su ejecución y comentarios oportunos) deberás describirlas en un informe estadístico en HTML confeccionado con R Markdown. El fichero HTML resultante deberás subirlo a PRADO en la tarea habilitada para esta sesión práctica.

1. Tarea 1: Aproximación de una probabilidad

Sea (X, Y) una variable aleatoria bidimensional con distribución uniforme en el cuadrado unidad, $(X, Y) \sim \mathcal{U}([-1, 1] \times [-1, 1])$.

Se trata de aproximar mediante simulación la probabilidad $P(X + Y \leq 0)$. El siguiente código nos permite resolver el problema. Además de la aproximación se calcula el error de estimación y se visualiza la convergencia construyendo un gráfico que incluya los límites de error.

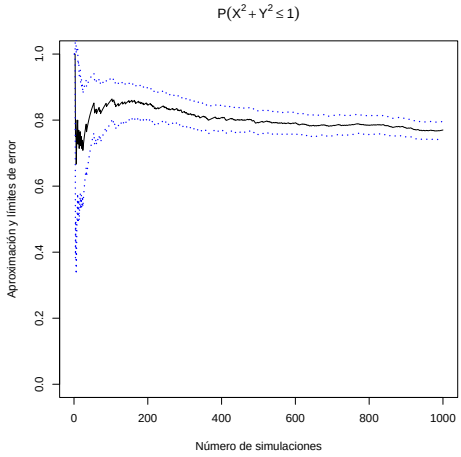
```
> nsim<-1000
> set.seed(1)
> x<-runif(nsim,-1,1)
> y<-runif(nsim,-1,1)
> suceso<-(x+y<=0)
> # Aproximación
> mean(suceso)
[1] 0.513
> # Error de estimación
> sd(suceso)/sqrt(nsim)
[1] 0.01581395
> ## Gráfico de convergencia
> # aproximaciones para $n=1,...,nsim$
> estim<-cumsum(suceso)/(1:nsim)
> # errores de estimación correspondientes
> estim.err<-sqrt(cumsum((suceso-estim)^2))/(1:nsim)
> plot(1:nsim,estim,type='l',ylab='Aproximación y límites de error',
+      xlab='Número de simulaciones',main=expression(P(X+Y<=0)),
+      ylim=c(0,1))
> z<-qnorm(0.025,lower.tail = FALSE)
> lines(estim - z*estim.err,col='blue',lwd=2,lty=3)
> lines(estim + z*estim.err,col='blue',lwd=2,lty=3)
```



Podemos comparar con el valor exacto de dicha probabilidad que en este caso es $1/2$.

Aproximar mediante simulación la probabilidad $P(X^2 + Y^2 \leq 1)$. Compara con el valor exacto que vale $\pi/4$.

```
> nsim<-1000
> set.seed(1)
> x<-runif(nsim,-1,1)
> y<-runif(nsim,-1,1)
> suceso<-(x^2+y^2<=1)
> # Aproximación
> mean(suceso)

[1] 0.77

> # Error de estimación
> sd(suceso)/sqrt(nsim)

[1] 0.01331455

> ## Gráfico de convergencia
> # aproximaciones para $n=1,...,nsim$
> estim<-cumsum(suceso)/(1:nsim)
> # errores de estimación correspondientes
> estim.err<-sqrt(cumsum((suceso-estim)^2))/(1:nsim)
> plot(1:nsim,estim,type='l',ylab='Aproximación y límites de error',
+      xlab='Número de simulaciones',main=expression(P(X^2+Y^2<=1)),
+      ylim=c(0,1))
> z<-qnorm(0.025,lower.tail = FALSE)
> lines(estim - z*estim.err,col='blue',lwd=2,lty=3)
> lines(estim + z*estim.err,col='blue',lwd=2,lty=3)
```



2. Tarea 2: Aproximación de una integral

Consideramos las dos integrales siguientes:

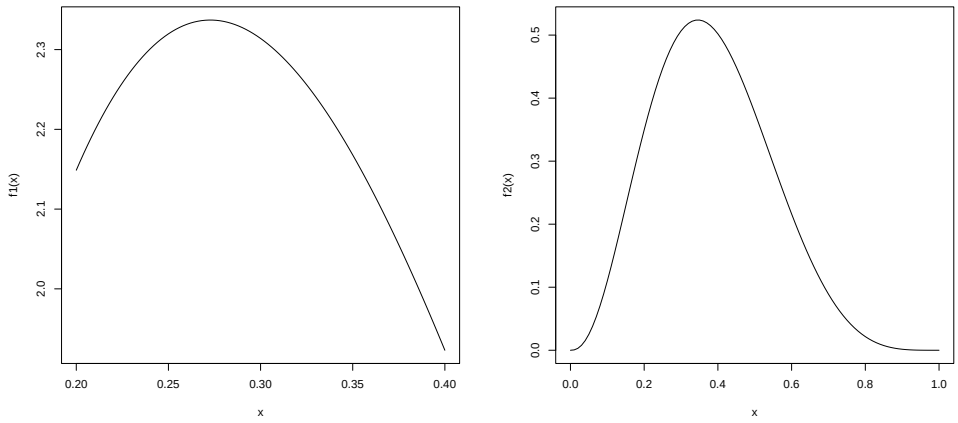
$$\begin{aligned} I_1 &= \int_{0.2}^{0.4} f(x)\ dx = \int I(0.2 < x < 0.4) f(x)\ dx \\ I_2 &= \int sin(x) e^{-x} f(x)\ dx, \end{aligned}$$

donde

$$f(x) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} x^{a-1}(1-x)^{b-1}, \quad 0 \leq x \leq 1,$$

es la función de densidad de una distribución Beta con parámetros $a = 2.5$ y $b = 5$. A continuación generamos gráficos que muestran las funciones a integrar:

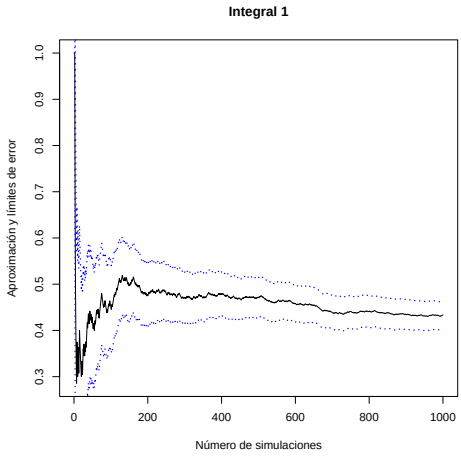
```
> # primera integral
> f1<-function(x) dbeta(x,2.5,5)
> curve(f1(x),0.2,0.4)
> # segunda integral
> f2<-function(x) sin(x)*exp(-x)*dbeta(x,2.5,5)
> curve(f2(x),0,1)
```



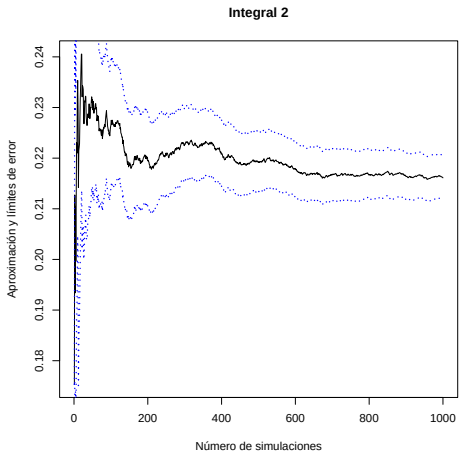
Una vez visualizados los problemas, el objetivo es aproximar las dos integrales anteriores utilizando simulación. Para ello te sugiero que consideres 1000 simulaciones. Además de la aproximación obtenida calcula su error de estimación y construye un gráfico que muestre la convergencia junto con los límites de error.

```
> # Integral 1
> nsim<-1000
> set.seed(1)
> x<-rbeta(nsim,shape1=2.5,shape2 =5)
> hx<-(0.2<x) & (x<0.4)
> # aproximaciones para $n=1,...,nsim$
> estim<-cumsum(hx)/(1:nsim)
> # errores de estimación correspondientes
> estim.err<-sqrt(cumsum((hx-estim)^2))/(1:nsim)
> plot(1:nsim,estim,type='l',ylab='Aproximación y límites de error',
+      xlab='Número de simulaciones',main='Integral 1')
> z<-qnorm(0.025,lower.tail = FALSE)
> lines(estim - z*estim.err,col='blue',lwd=2,lty=3)
> lines(estim + z*estim.err,col='blue',lwd=2,lty=3)
> # Aproximación final y su error de estimación
> estim[nsim]

[1] 0.433

> estim.err[nsim]

[1] 0.01561774

> ## Otra forma de aproximar la integral 1:
> nsim<-1000
```

```
> set.seed(1)
> u<-runif(nsim,0.2,0.4)
> (0.4-0.2)*mean(f1(u))

[1] 0.4443133
```




```
> # Integral 2
> nsim<-1000
> set.seed(1)
> x<-rbeta(nsim,shape1=2.5,shape2 =5)
> hx<-sin(x)*exp(-x)
> # aproximaciones para $n=1,...,nsim$
> estim<-cumsum(hx)/(1:nsim)
> # errores de estimación correspondientes
> estim.err<-sqrt(cumsum((hx-estim)^2))/(1:nsim)
> plot(1:nsim,estim,type='l',ylab='Aproximación y límites de error',
+      xlab='Número de simulaciones',main='Integral 2')
> z<-qnorm(0.025,lower.tail = FALSE)
> lines(estim - z*estim.err,col='blue',lwd=2,lty=3)
> lines(estim + z*estim.err,col='blue',lwd=2,lty=3)
> # Aproximación final y su error de estimación
> estim[nsim]

[1] 0.2161039

> estim.err[nsim]

[1] 0.002228571
```



Finalmente puedes comparar tus aproximaciones con las aproximaciones numéricas que calcula la función `integrate`:

```
> # primera integral
> f1<-function(x) dbeta(x,2.5,5)
> integrate(f1,0.2,0.4)

0.4441656 with absolute error < 4.9e-15

> # también podemos evaluarla con la función pbeta()
> pbeta(0.4,2.5,5)-pbeta(0.2,2.5,5)

[1] 0.4441656

> # segunda integral
> f2<-function(x) sin(x)*exp(-x)*dbeta(x,2.5,5)
> integrate(f2,-Inf,Inf)

0.2172255 with absolute error < 7.9e-05
```

3. Tarea 3: Aproximación de una distribución de probabilidad

La simulación también nos permite aproximar la distribución completa de una variable aleatoria X. Si somos capaces de simular un número suficientes de valores de la variable, $X_1, \ldots, X_n$, entonces podremos usar dichos valores para aproximar la distribución utilizando por ejemplo un histograma. También podremos aproximar otras características de la distribución como por ejemplo la media (calculando la media de los valores generados),

la varianza, incluso un percentil. En esta parte vamos a utilizar esta herramienta para aproximar la distribución y algunas características de una variable aleatoria especial, la denominada **distribución de Poisson compuesta**.

> En la Teoría de la Probabilidad una distribución de Poisson compuesta (*compound Poisson*) es la distribución de probabilidad de la suma de N variables aleatorias independientes e idénticamente distribuidas, $X_1, \ldots, X_N$, donde N es una variable con distribución de Poisson independiente de las X_i:
>
> $$S_N = \sum_{i=1}^{N} X_i, \quad N \sim \mathcal{P}(\lambda).$$
>
> Los dos primeros momentos de S_N están dados por las siguientes expresiones exactas:
>
> $$\mathbb{E}[S_N] = \mathbb{E}[N]\mathbb{E}[X_1] \quad (1)$$
> $$\mathbb{V}[S_N] = \mathbb{E}[N]\mathbb{V}(X_1) + \mathbb{V}(N)\mathbb{E}[X_1]^2 \quad (2)$$
>
> A parte de las expresiones de estos momentos, en general no es posible expresar la distribución compuesta de forma analítica y es necesario recurrir a su aproximación.

La distribución de Poisson compuesta es muy utilizada en aplicaciones actuariales. Para este ejercicio vamos a considerar un problema de este contexto:

Cada mes el número de reclamaciones que se resuelven en una aseguradora, N, sigue una distribución de Poisson con media $\lambda = 17$. Además cada reclamación resuelta implica un pago por parte de la seguradora, X_i $(i = 1, \ldots, N)$, con distribución de probabilidad lognormal con parámetros $\mu = 3.5$ y $\sigma = 1.1$. La cuantía total que paga la aseguradora en un mes viene dada por la siguiente variable aleatoria:

$$S_N = \sum_{i=1}^{N} X_i,$$

la cual sigue una distribución de Poisson compuesta (asumiendo que las reclamaciones son independientes entre sí).

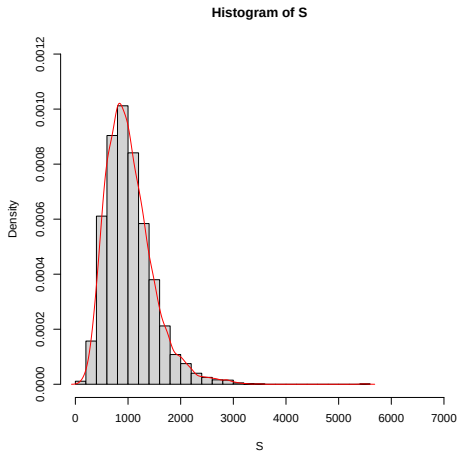
Las expresiones (1) y (2) nos permiten calcular de manera exacta la media y la varianza de S_N:

```
> # Parámetros de la lognormal (X)
> mu <- 3.5
> sig <- 1.1
> # a partir de ellos calculamos E[X] y V(X)
> EX<- exp(mu+sig^2/2) ; EX
```

```
[1] 60.64274

> VX<- EX^2*(exp(sig^2)-1) ; VX

[1] 8655.04

> # Parámetro de la Poisson (N)
> lam<- 17 # coincide con E[N] y V(N)
> # Media de la Poisson compuesta
> ES<-lam*EX ; ES

[1] 1030.927

> # Varianza de la Poisson compuesta
> VS<- lam*VX + lam*EX^2 ; VS

[1] 209653.9
```

Ya tenemos los dos primeros momentos, nuestro objetivo ahora es la distribución de probabilidad completa de S_N. Como comentamos antes esto no es posible analíticamente así que vamos a hacerlo usando simulación. Para ello necesitamos generar suficientes valores de S_N (e.g. 5000) y a partir de ellos obtendremos una aproximación de la función de densidad representando un histograma de los valores generados usando la función `hist()` (o incluso un estimador de la densidad suavizada que hemos hecho en alguna práctica anterior usando `density()`). Lo hacemos como sigue:

```
> nsim<-5000 # número de simulaciones
>
> # simulamos ahora nsim valores de S_N:
> S <- double(nsim) # para almacenar los valores simulados
> set.seed(1)
> for (i in 1:nsim)
+ {
+    n <- rpois(1,lam)
+    if (n>0) S[i] <- sum(rlnorm(n,mu,sig))
+ }
> # el vector S contiene los valores simulados de S_N
> # un histograma de S nos da una aproximación de la distribución
> hist(S,xlim=c(0,7000),breaks=20,prob=TRUE, ylim=c(0,1.2e-3))
> # podemos superponer la densidad suavizada que calcula density()
> lines(density(S),col="red")
```



En el código hemos utilizado un bucle para generar uno a uno cada valor de S_N en dos pasos: generando primero un valor `n` desde una Poisson (usando e `rpois()`) y luego `n` valores desde una lognormal (usado la función `rlnorm()`). La suma de estos últimos, `S`, nos da un valor aleatorio de S_N.
Por otro lado el gráfico de la distribución (tanto el histograma como la densidad suavizada) nos muestran que la distribución es ligeramente asimétrica hacia la derecha.

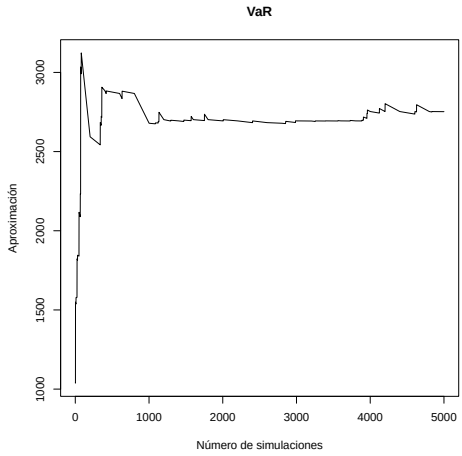
Calcula la media y la varianza de los `nsim` valores simulados de S_N. Estos nos dan aproximaciones de la media y la varianza exactas que calculamos antes de forma exacta.

```
> mean(S)

[1] 1028.819

> var(S)

[1] 206025.3
```

Un valor muy importante en aplicaciones actuariales y en finanzas es lo que se denomina VaR (del inglés *Value at Risk*). Se trata de una medida de riesgo y consiste en un percentil elevado de la distribución, habitualmente el 99.5 %. En el problema que hemos planteado, calcular el VaR para S_N nos dará la cuantía máxima que la compañía tendrá que pagar en un mes para el 99.5 % de las reclamaciones.

Utilizando los valores que hemos simulado de S_N, aproximar el VaR. [Nota: Dado que se trata de calcular un cuantil de un vector de "datos" puedes usar la función `quantile()`.]

```
> quantile(S,0.995)

   99.5%
2752.155

> # podemos pintar el gráfico de convergencia (sin límites de error)
> VaRs<-sapply(1:nsim,function(i) quantile(S[1:i],0.995))
> plot(1:nsim,VaRs,type='l',ylab='Aproximación',
+      xlab='Número de simulaciones',main='VaR')
```



Finalmente podemos comparar la aproximación del VaR obtenida con simulación con la que nos daría una aproximación Normal, usando la media y la varianza de la distribución que calculamos antes. Para ello usamos la función `qnorm()`:

```
> # aproximación por una Normal con media ES y varianza VS
> qnorm(0.995, mean=ES, sd=sqrt(VS))

[1] 2210.347
```