Simulación ITAM
18 de noviembre de 2018 2018-II

Tarea 3.

La fecha de entrega es el viernes **5 de octubre de 2018**.

Lecturas

- Casella y Robert, leer la sección 1.5 y el capítulo 2.
- Fast Generation of Discrete Random Variables Artículo de *Journal of Statistical Software, July 2004, Volume 11, Issue 3.*
- Introduction to simulation using R Capítulo 13
- Marc C. Bove, et. al Effect of El Niño on U.S. Landfalling Hurricanes, Revisited

Problemas

1. Un estadístico está interesado en el número N de peces en un estanque. El captura 250 peces, los marca y los regresa al estanque. Unos cuantos días después regresa y atrapa peces hasta que obtiene 50 peces marcados, en ese punto también tiene 124 peces no marcados (la muestra total es de 174 peces).

   - ¿Cuál es la estimación de N?
   - Hagan un programa (en excel o en R), que permita simular el proceso de obtener la primera y segunda muestra considerando como parámetros el tamaño N de la población de interés, el tamaño de la primera y segunda muestra y como datos a estimar son: de qué tamaño debe ser n_1 y n_2 para obtener una buena aproximación y ver cómo se afecta por el tamaño N.

   Solución.

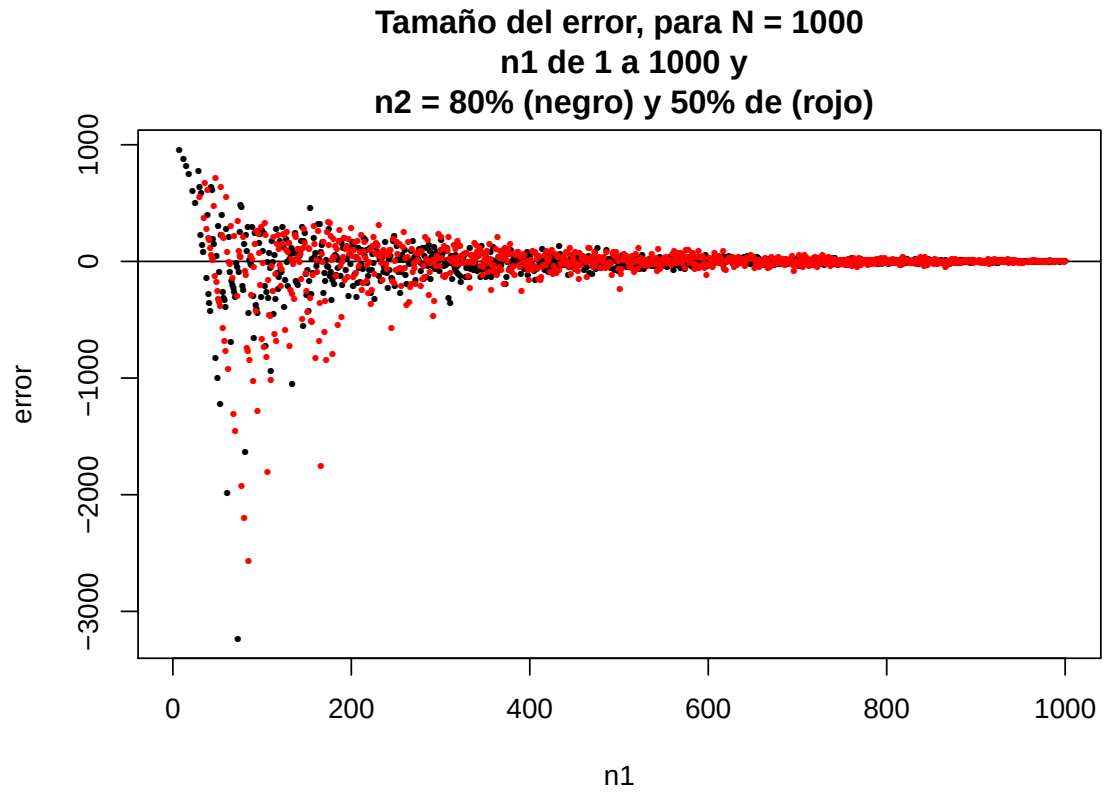
   De acuerdo a lo que vimos en clase, la primera captura es $n_1 = 250$ y la segunda muestra es $n_2 = 174$ y los marcados son $r = 50$. La estimación de N es entonces $N = \frac{n_1 n_2}{r} = 870$.

   Ahora podemos hacer un ejercicio de simulación tomando un valor fijo de N. Supongamos que n_1 es el tamaño de la primera muestra y n_2 el de la segunda muestra. Estos tres valores son dados, pero el $r =$ número de peces marcados en la segunda

muestra es una variable aleatoria que sigue una distribución hipergeométrica, con parámetros (N, n_1, n_2). Sin embargo, en el ejercicio de simulación que se propone, debemos estimar N a partir del valor que obtengamos de r.

La siguiente función calcula el valor estimado a partir de realizar las muestras. A partir de esta función se pueden hacer curvas de nivel para analizar el comportamiento del error de acuerdo a los valores de n_1 y n_2.

```
CapRecap <- function(N, n1, n2){
marca1 <- sample(1:N, n1) #obten primera muestra
marca2 <- sample(1:N, n2) #obten segunda muestra
r <- length(intersect(marca1, marca2)) #recapturados
Nhat <- n1*n2/r
return(list(N = N, Nhat = Nhat, error = N - Nhat))
}
N <- 1000 #Consideremos un ejemplo,
error <- NULL
for(i in 1:N) error[i] <- CapRecap(N, i, round(i*0.8, 0))$error
plot(error, pch = 16, cex = 0.5,
main = "Tamaño del error, para N = 1000\n n1 de 1 a 1000 y\n n2 = 80% (negro) y 50% de (rojo)",
xlab = "n1")
abline(h = 0)
for(i in 1:N) error[i] <- CapRecap(N, i, round(i*0.5,0))$error
points(error, pch = 16, cex = 0.5, col = "red")
```



□

2. Este problema es una versión simplificada de dos problemas comunes que enfrentan las compañías de seguros: calcular la probabilidad de quebrar y estimar cuánto dinero podrán hacer.

Supongan que una compañía de seguros tiene activos (todo en dólares) por \$ 10^6. Tienen $n = 1,000$ clientes que pagan individualmente una prima anual de \$5,500, al principio de cada año. Basándose en experiencia previa, se estima que la probabilidad de que un cliente haga un reclamo es $p = 0.1$ por año, independientemente de reclamos previos de otros clientes. El tamaño X de los reclamos varía, y tiene la siguiente densidad con $\alpha = 5$ y $\beta = 125,000$:

$$f(x) = I(x \geq 0)\frac{\alpha\beta^{\alpha}}{(x+\beta)^{\alpha+1}}$$

(Tal X se dice que tiene una distribución Pareto, y en el mundo real no es un modelo infrecuente para el tamaño de los reclamos a las aseguradoras). Suponemos las fortunas de la compañía aseguradora sobre un horizonte de 5 años. Sea $Z(t)$ los activos de la compañía al final del año t, así que $Z(0) = 1,000,000$, y $Z(t) = I(Z(t-1) > 0)max\{Z(t-1) + \text{primas} - \text{reclamos}, 0\}$

Noten que si $Z(t)$ cae bajo 0 entonces se queda ahí. Si la compañía se va a bancarrota, deja de operar.

- Calcular la función de distribución F_X, $E[X]$, y $Var[X]$. Obtengan por simulación una muestra de X y su función de distribución, y comparen su estimado con la verdadera
- Escriban una función para simular los activos de la compañía por cinco años y estimen: (1) la probabilidad de que la compañía se vaya a bancarrota,y (2) Los activos esperados al final de cinco años.
- Toma de ganancias. Supongan ahora que la compañía toma ganancias al final de cada año. Esto es, si $Z(t) > 1,000,000$ entonces $Z(t) - 1,000,000$ se le paga a los accionistas. Si $Z(t) \leq 1,000,000$ entonces los accionistas no reciben nada ese año. Usando este nuevo esquema, estimar: (1) la probabilidad de irse a la quiebra, (2) los activos esperados después de 5 años, y (3) Las ganancias totales esperadas después de cinco años.

Solución.

- El primer inciso es un ejercicio estándar en probabilidad. Para obtener la fun-

ción de distribución:

$$\begin{aligned} F(x) &= \int_0^x \frac{\alpha\beta^\alpha}{(u+\beta)^{\alpha+1}}du \\ &= \alpha\beta^\alpha \int_0^x \frac{du}{(u+\beta)^{\alpha+1}} \\ &= \alpha\beta^\alpha \frac{(u+\beta)^{-\alpha-1+1}}{-\alpha-1+1}\Big|_0^x = \frac{\alpha\beta^\alpha}{-\alpha(u+\beta)^\alpha}\Big|_0^x \\ &= \frac{\beta^\alpha}{(x+\beta)^\alpha} - \frac{\beta^\alpha}{-\beta^\alpha} \\ &= 1 - \frac{\beta^\alpha}{(x+\beta)^\alpha} \end{aligned}$$

Para encontrar la esperanza y la varianza, hay que realizar integración por partes y para valuar una integral se requiere aplicar el Teorema de L'Hôpital. Para evlaur la varianza, se requiere también integración por partes, y la valuación de un límite, así como incorporar la condición de que $\alpha > 2$. Los resultados indican que:

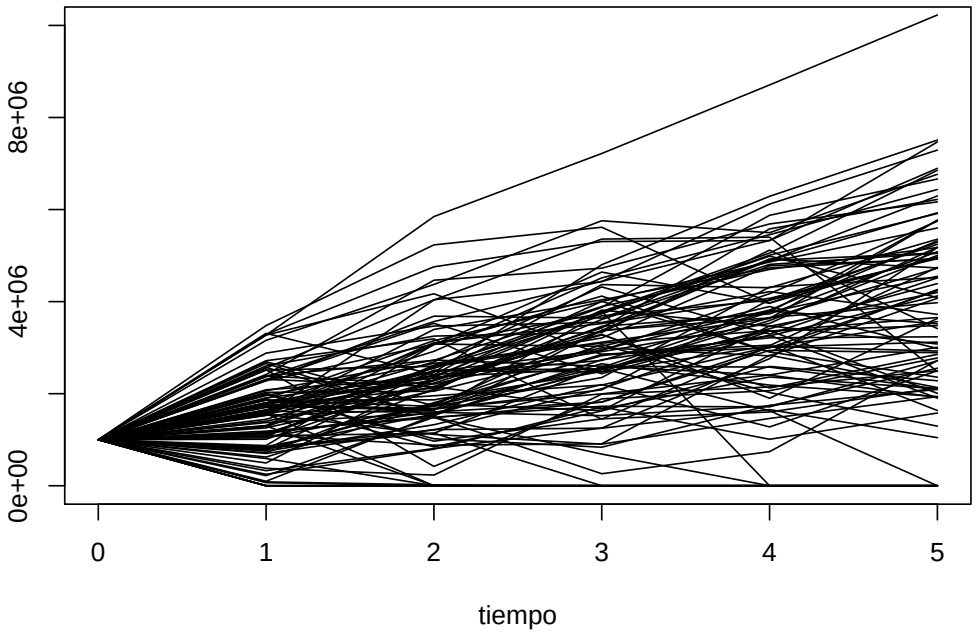
$$\mathrm{E}(X) = \frac{\beta}{\alpha - 1}$$

y

$$\mathrm{Var}(X) = \frac{\beta^2\alpha}{(\alpha-1)(\alpha-2)}$$

- A continuación construyo la función que simula el proceso estocástico. Consideramos al evento de que un cliente haga un reclamo como una variable Bernoulli con probabilidad $p = 0.1$, y como son independientes, podemos suponer que el número de reclamos anual es una variable Binomial. Por lo tanto, este proceso tiene dos tipos de variación: una para saber cuántos clientes hacen reclamos, y otra para determinar el tamaño de la reclamación.

```
library(actuar) #para muestrear de una distribución Pareto


Attaching package: 'actuar'
The following object is masked from 'package:grDevices':
    cm

Proceso <- function(ActIni = 1e6, clientes = 1000, t = 5, prima = 5500){
Z <- NULL
Z0 <- ActIni
for(i in 1:t){
reclamos <- sum(rbinom(clientes,1,0.1)) #clientes que generan reclamos, binomial
Z[i] <- max(ifelse(i==1,Z0,Z[i-1]) + clientes*prima - sum(rpareto(reclamos,3,100000)),0)
if(Z[i]==0){Z[i:t] <-0;break} #verifica si se alcanzó 0, y todos los consecutivos 0.
}
return(c(Z0,Z))
}
y <- NULL
plot(0:5,Proceso(),type="l",ylim=c(0,10e6),xlab="tiempo",ylab="")
for(i in 1:100){y[[i]] <- Proceso();lines(0:5,y[[i]])}
```

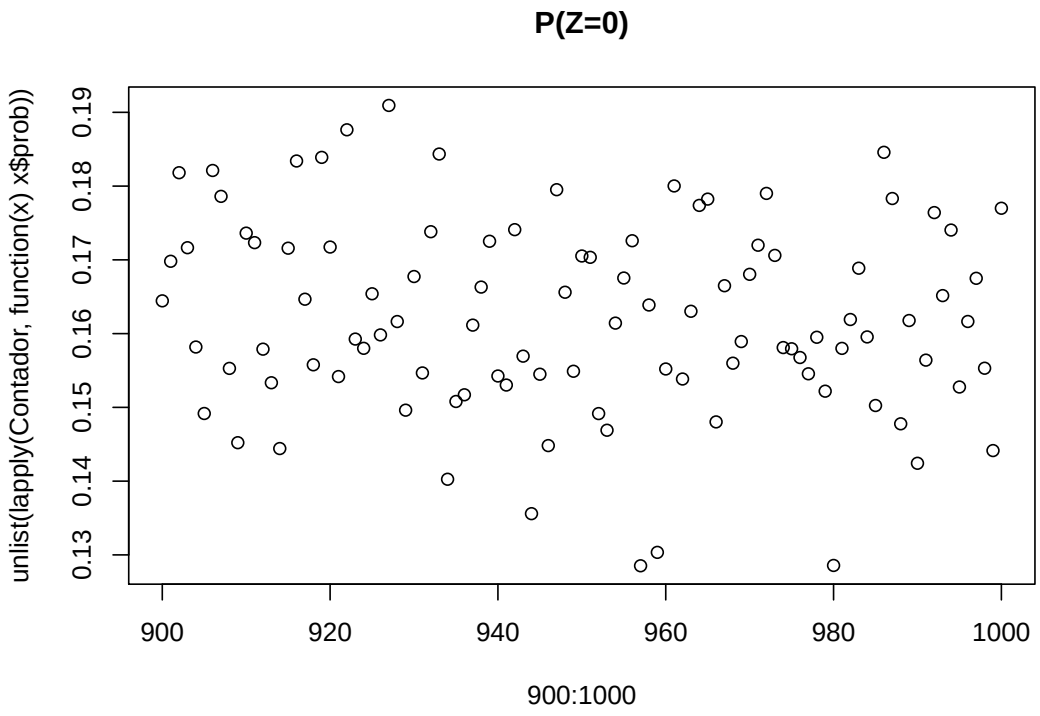



```
Trayectorias <- matrix(unlist(y),ncol=6,byrow=T)
head(Trayectorias)
```

```
      [,1]    [,2]    [,3]      [,4]      [,5]     [,6]
[1,] 1e+06 1799505 2919356 4515533.7 5453996.2  6435308
[2,] 1e+06 2716305 3190883 1447216.2 3026876.5  2926738
[3,] 1e+06 3273040 5847348 7217497.7 8706795.1 10226651
[4,] 1e+06 2657807 1487349  261202.4  740478.4  2723027
[5,] 1e+06 1558606 3395462 4038467.9 4731305.7  4936130
[6,] 1e+06 1701021 1955510 2188090.5 3015325.3  3565862
```

Para estimar la probabilidad de que la compañía se vaya a la bancarrota, podemos contar el número de trayectorias en donde la serie toca 0, para un número N de simulaciones. También se calculan los activos esperados después de 5 años. Como se puede apreciar, este proceso no se estabiliza incrementando el número de trayectorias simuladas.

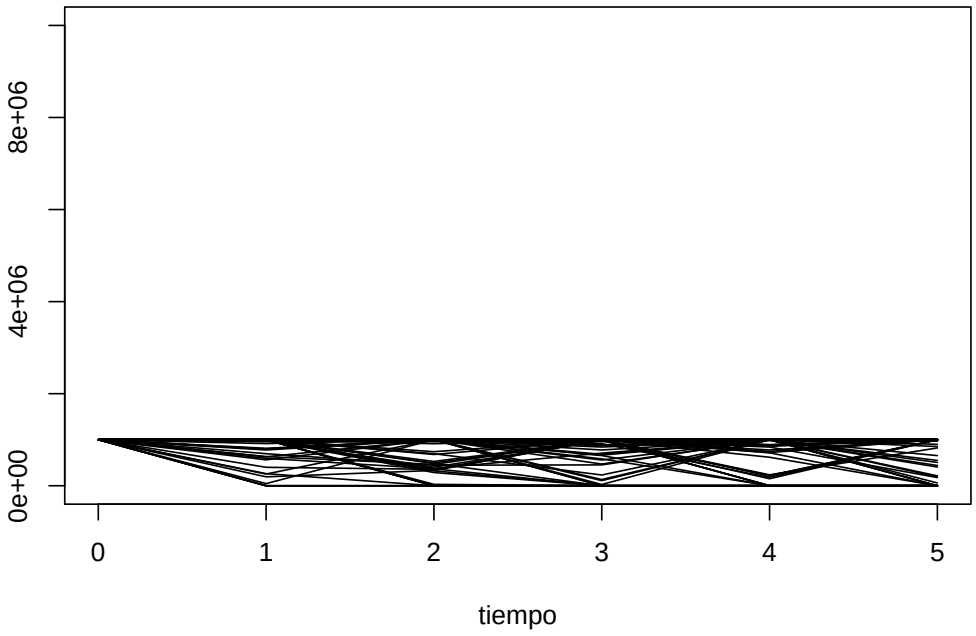
```
simula <- function(N){
Tr <- NULL
for(i in 1:N)Tr[[i]] <- Proceso()
Tr <- matrix(unlist(Tr),ncol=6,byrow = T)
prob <- sum(apply(Tr,1,function(x)0 %in% x))/N
acti <- mean(Tr[,6])
return(list(prob=prob,activos=acti))
}
Contador <- list(NULL)
for(n in 900:1000) Contador[[n-899]] <- simula(n)
plot(900:1000,unlist(lapply(Contador,function(x)x$prob)),main="P(Z=0)")
```




```
mean((unlist(lapply(Contador,function(x)x$acti)))) #Activos promedio después de 5 años

[1] 3394131
```

- Ahora modificamos el ejercicio para toma de ganancias.

```
#Considera ahora la toma de ganancias.
Proceso2 <- function(ActIni=1e6,clientes=1000,t=5,prima=5500){
Z <- NULL
G <- NULL #Ganancias
Z0 <- ActIni
for(i in 1:t){
reclamos <- sum(rbinom(clientes,1,0.1)) #clientes que generan reclamos, binomial
Z[i] <- max(ifelse(i==1,Z0,Z[i-1]) + clientes*prima - sum(rpareto(reclamos,3,100000)),0)
if(Z[i]==0){Z[i:t] <-0;break} #verifica si se alcanzó 0, y todos los consecutivos 0.
if(Z[i]>=1e6){G[i] <- Z[i]-1e6;Z[i] <- 1e6} # Toma de ganancias
}
is.na(G) <- 0
return(list(Z=c(Z0,Z),G=G))
}
y <- NULL
plot(0:5,Proceso2()$Z,type="l",ylim=c(0,10e6),xlab="tiempo",ylab="")
for(i in 1:100){y[[i]] <- Proceso2()$Z;lines(0:5,y[[i]])}
```

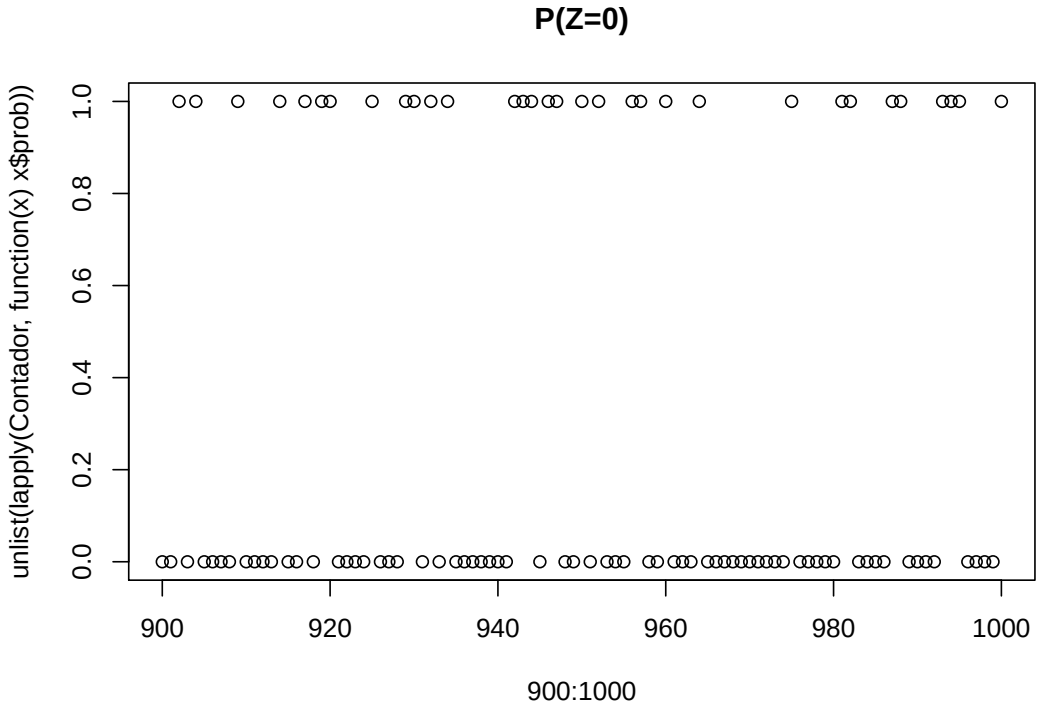



```
Trayectorias <- matrix(unlist(y),ncol=6,byrow=T)
head(Trayectorias)
```

```
      [,1]       [,2]      [,3]    [,4]      [,5]      [,6]
[1,] 1e+06 1000000.00 1000000.0 1000000 1000000.0  990485.9
[2,] 1e+06 1000000.00 1000000.0 1000000 1000000.0       0.0
[3,] 1e+06  977918.83  405271.8  980393  152732.3 1000000.0
[4,] 1e+06   47289.96 1000000.0 1000000 1000000.0 1000000.0
[5,] 1e+06 1000000.00 1000000.0 1000000 1000000.0 1000000.0
[6,] 1e+06  696559.49  446408.7  452827 1000000.0 1000000.0
```

Las estadísticas requerida son ahora:

```
#función que depende del número de trayectorias generadas
simula2 <- function(N){
Tr <- NULL #Trayectorias
Gn <- NULL #Ganancias
y <- Proceso2()
for(i in 1:N){Tr[[i]] <- y$Z;Gn[[i]] <- y$G}
Tr <- matrix(unlist(Tr),ncol=6,byrow = T)
prob <- sum(apply(Tr,1,function(x)0 %in% x))/N
acti <- mean(Tr[,6])
return(list(prob = prob, activos = acti,
ganancias = sum(unlist(lapply(Gn,sum,na.rm=T))/N,na.rm = T)))
}
Contador <- list(NULL)
for(n in 900:1000) Contador[[n-899]] <- simula2(n)
plot(900:1000,unlist(lapply(Contador,function(x)x$prob)),main="P(Z=0)")
```




```
mean((unlist(lapply(Contador,function(x)x$acti)))) #Activos promedio después de 5 años
```

```
[1] 647466.4
```

```
mean((unlist(lapply(Contador,function(x)x$ganancias)))) #Ganancia promedio después de 5 años
```

```
[1] 3336776
```

□

3. Proponer algoritmos (método y pseudocódigo o código, así como una corrida) para generar muestras de las siguientes densidades:

   (a) Cauchy

   $$f(x) = \frac{1}{\pi\beta\left[1 + \left(\frac{x-\gamma}{\beta}\right)^2\right]}$$

   donde $\gamma, x \in \Re$, $\beta > 0$.

   (b) Gumbel (o de valor extremo)

   $$f(x) = \frac{1}{\beta}\exp\left(-e^{-(x-\gamma)/\beta} - \frac{x-\gamma}{\beta}\right)$$

   donde $\gamma, x \in \Re$, $\beta > 0$.

(c) Logística

$$f(x) = \frac{(1/\beta)e^{-(x-\gamma)/\beta}}{(1+e^{-(x-\gamma)/\beta})^2}$$

donde $\gamma, x \in \Re$, $\beta > 0$.

(d) Pareto

$$f(x) = \frac{\alpha_2 c^{\alpha_2}}{x^{\alpha_2+1}}$$

donde $c > 0, \alpha_2 > 0, x > c$.

Para $\gamma = 0$ y $\beta = 1$ en cada inciso (a), (b) y (c), usen los algoritmos que obtuvieron para generar una muestra aleatoria de 5000 y obtengan $\bar{X}(n) = \sum_{i=1}^{n} X_i/n$ para $n = 50, 100, 150, \ldots, 5000$ para verificar empíricamente la ley fuerte de los grandes números. Hacer lo mismo para (d) con $c = 1$ y $\alpha_2 = 2$.

Solución.

(a) Este caso ya lo hicimos como ejemplo de clase, con una distribución Cauchy estandarizada. Basta con estandarizar la variable $Z = \frac{X-\gamma}{\beta}$. Aquí nos sirve directamente el método de la transformación inversa.

(b) La distribución Gumbel puede invertirse fácilmente. La distribución es de la forma $F(x) = e^{-e^{-\frac{x-\mu}{\beta}}}$ y entonces $X = -\beta log(-log(u)) + \mu$. El método de la transformación inversa se puede aplicar de manera directa.

(c) La distribución logística también es muy fácil de invertir. Tiene distribución $F(x) = \frac{1}{1+e^{-(x-\mu)/\beta}}$. Entonces $X = -\beta log(1/u - 1) + \mu$.

(d) La función de distribución está dada por $F(x) = 1 - \left(\frac{c}{x}\right)^{\alpha_2}$ Lo que resulta en la inversa $X = \frac{c}{(1-u)^{1/\alpha_2}}$.

El resto del ejercicio es un ejercicio estándar fácil de completar.

□

4. Grafiquen las siguientes densidades. Dar los algoritmos de transformación inversa, composición y aceptación-rechazo para cada una de las siguientes densidades. Discutir cuál algoritmo es preferible para cada densidad.

(a)

$$f(x) = \frac{3x^2}{2} I(x)_{[-1,1]}$$

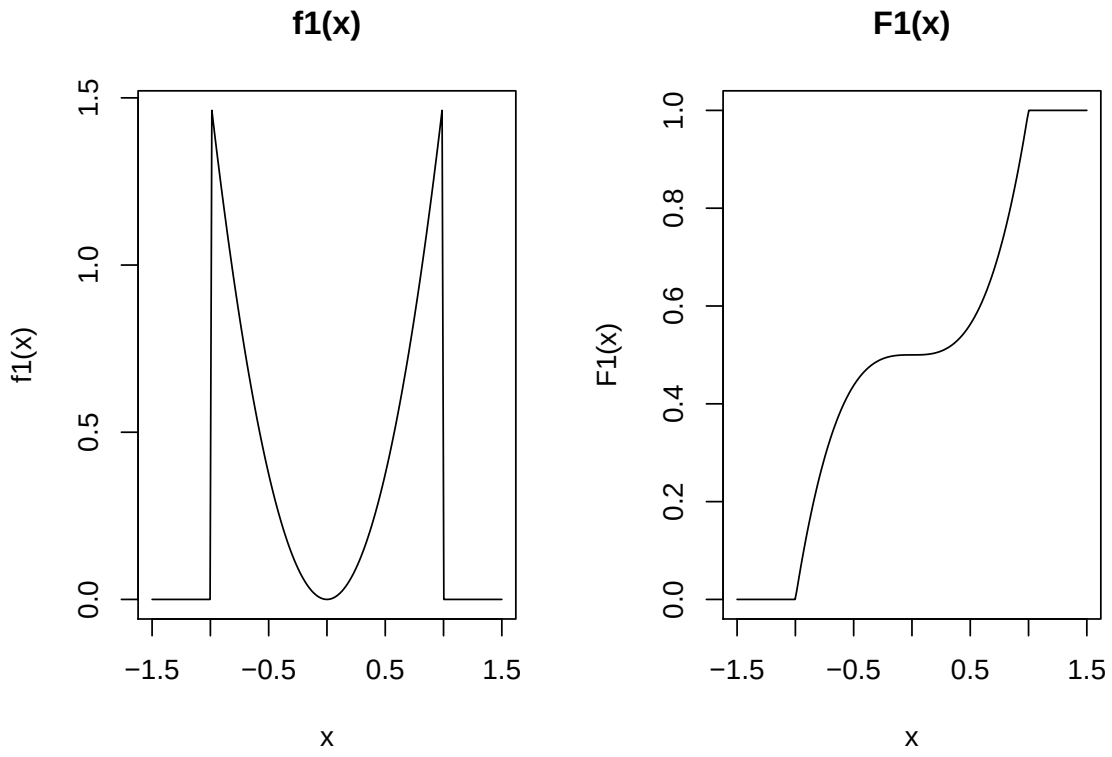
(b) Para $0 < a < \frac{1}{2}$,

$$f(x) = \begin{cases} 0 & x \leq 0 \\ \frac{x}{a(1-a)} & 0 \leq x \leq a \\ \frac{1}{1-a} & a \leq x \leq 1-a \\ \frac{1-x}{a(1-a)} & 1-a \leq x \leq 1 \\ 0 & x \geq 1 \end{cases}$$

Solución.

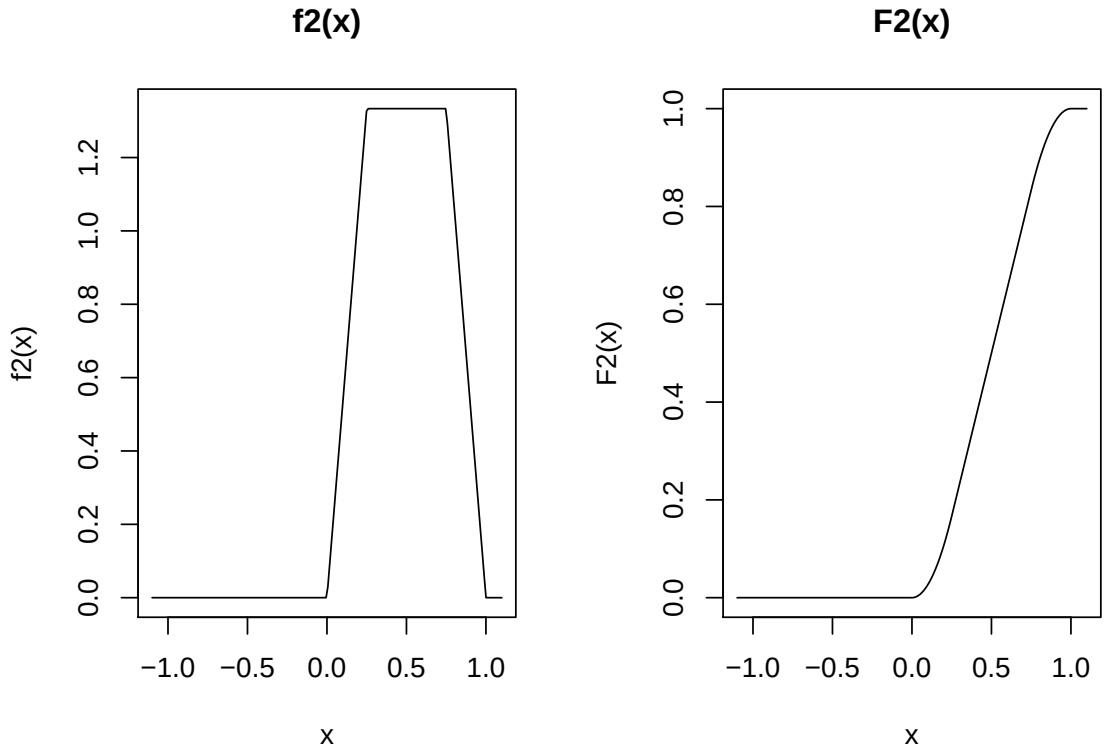
Para el inciso (a), consideremos las funciones de densidad y distribución dadas a continuación:

```
indicator <- function(x,a,b){ifelse(x <=b & x >=a,1,0)} #función indicadora en el intervalo (a,b)
f1 <- function(x){3*x^2/2*indicator(x,-1,1)}
F1 <- function(x){ifelse(x<=-1,0,ifelse(x<=1,0.5*(x^3+1),1))}
x <- seq(-1.5,1.5,length=200)  #intervalo de graficación
par(mfrow=c(1,2))
plot(x,f1(x),type="l",main="f1(x)")
plot(x,F1(x),type="l",main="F1(x)")
```



Para el inciso (b)

```r
f2 <- function(x,a=0.25){indicator(x,-1,1)*(indicator(x,0,a)*(x/(a*(1-a))) +
indicator(x,a,1-a)/(1-a) +
indicator(x,1-a,1)*(1-x)/(a*(1-a)) )}
F2 <- function(x,a=0.25){indicator(x,0,a)*x^2/(2*a*(1-a)) +
(x-a/2)/(1-a)*indicator(x,a,1-a) +
((1-3*a/2)/(1-a) + (x*(1-x/2)-(1-a)*(1+a)/2)/(a*(1-a)))*indicator(x,1-a,1) +
indicator(x,1,100)
}
par(mfrow=c(1,2))
x <- seq(-1.1,1.1,length=200)  #intervalo de graficación
plot(x,f2(x),type="l",main="f2(x)")
plot(x,F2(x),type="l",main="F2(x)")
```



□

5. Considerando la transformación polar de Marsaglia para generar muestras de normales estándar, muestren que la probabilidad de aceptación de $S = V_1^2 + V_2^2$ en el paso 2 es $\pi/4$, y encuentren la distribución del número de rechazos de S antes de que ocurra una aceptación. ¿Cuál es el número esperado de ejecuciones del paso 1?

Solución.

Para la primera parte, basta un algoritmo geométrico. La región en donde se rechazan los puntos corresponden al área sobrante del cuadrado que circunscribe el círculo con radio unitario. Esa región tiene área $\frac{4-\pi}{4} = 1 - \pi/4 = 0.215$. Entonces se

rechaza 21.5 % del tiempo. Ahora bien, si $X =$ número de rechazos antes de aceptar, sabemos que $X \sim geom(\pi/4)$. Entonces $E(X) = 1/\frac{\pi}{4} = 4/\pi \approx 1.2732395$

□

6. Obtengan una muestra de 1,000 números de la siguiente distribución discreta:

$$p(x) = \frac{2x}{k(k+1)}, x = 1, 2, \ldots, k$$

para $k = 100$.

Solución.

Noten que la función de distribución tiene la forma: $F(j) = P(X \leq j) = \frac{2}{k(k+1)} \sum_{i=1}^{j} x = \frac{2}{k(k+1)} \times \frac{j(j+1)}{2}$ ya que la suma corresponde a la suma de los primeros j naturales. Entonces podemos generar los "cajones" de la función acumulativa relativamente simple.

```
set.seed(1)
k <- 100   #parámetro de la distribución
n <- 10000 #tamaño de muestra
Fn <- ((1:k)*(1+(1:k)) )/(k*(k+1)) #Escalones de distribución
u <- runif(10000) #bten unifotmes
muestra <- findInterval(u,Fn,left.open=T) #muestra solicitada
hist(muestra,prob=T,main="histograma de la muestra")
lines(1:100,2*(1:100)/(100*101),col="red") #función de probabilidad
points(muestra,rep(0,10000),pch="|",cex=0.5) #puntos de la muestra
```

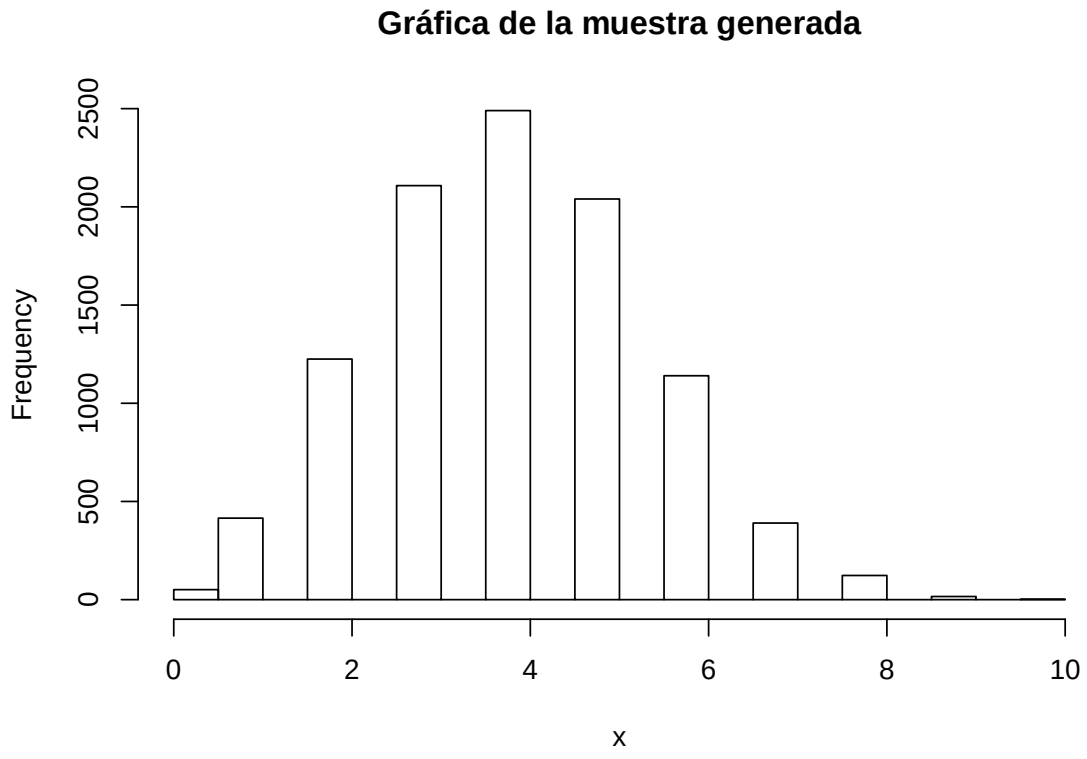
□

7. Desarrollen un algoritmo para generar una variable aleatoria binomial, usando la técnica de convolución (Hint: ¿cuál es la relación entre binomiales y Bernoullis?) Generar una muestra de 100,000 números. ¿Qué método es más eficiente, el de convoluciones o la función `rbinom` en R?

Solución.

El método de convolución es el que genera sumas de variables aleatorias. En este caso, una binomial con parámetros n y p es la suma de n variables Bernoulli con parámetro p. Por lo tanto, basta con generar n Bernoullis y agregar

```
muestraBinomial <- function(N,n,p){
x <- NULL
for(i in 1:N){
u <-runif(n)
y <- ifelse(u<=p,1,0)
x <- c(x,sum(y))
}
return(x)
}
x <- muestraBinomial(10000,10,0.4)
hist(x,main="Gráfica de la muestra generada")
```



Por último, para ver cuál es más eficiente, podemos tomar el tiempo de ejecución de ambos métodos

```
system.time(x <- muestraBinomial(1000,10,0.4))
```

```
   user  system elapsed 
   0.01    0.00    0.01 
```

```
system.time(x <- rbinom(1000,10,0.4))
```

```
   user  system elapsed 
      0       0       0 
```

Claramente, el método de R es mucho más eficiente.

□

8. Para un proceso Poisson no homogéneo con función de intensidad dada por

$$\lambda(t) = \begin{cases} 5, t \in (1,2], (3,4], \cdots \\ 3, t \in (0,1], (2,3], \cdots \end{cases}$$

a) Grafiquen una ejemplo del proceso considerando el intervalo de tiempo [0,100].

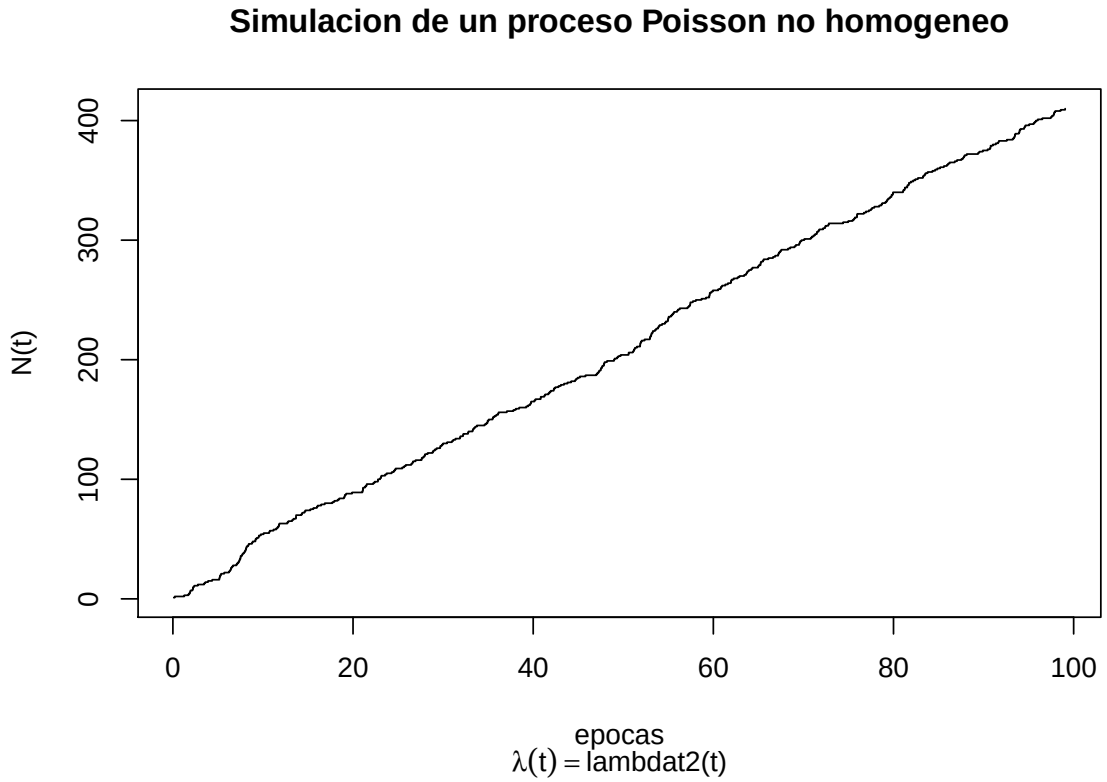
b) Grafiquen el proceso hasta obtener 100 eventos

c) Estimen la probabilidad de que el número de eventos observados en el periodo de tiempo (1.25,3] es mayor que 2.

Solución.

Para el primer problema, tenemos que definir la función pulso. Debido a que en el proceso de aceptación-rechazo se pierden algunas observaciones, el numero de observaciones necesarios para llegar al tiempo 100 se obtiene por ensayo y error, o bien, cambiar la programación y primero generar todas las exponenciales hasta que se acumule el número que necesitamos. Yo hice la otra programación.

```
lambdat2 <- function(t){
x <- paste("","{",0,"<= t & t <",1,"}",sep="")
for(i in seq(2,100,2)){x <- paste(x,paste("","{",i,"<= t & t <",i+1,"}",sep=""),sep="|")}
return(ifelse(eval(parse(text=x)),3,5))
}

poisson.nohomogeneo<-function(lambdat,n,pic=T){
lambda <- 5 #mayoriza la función lambdat
TT <- rexp(n,lambda) #genera variables exponenciales para los tiempos.
s <- cumsum(TT) #acumula los tiempos en el vector s
u <- runif(n) #obten n uniformes
ss <- s[u <= lambdat(s)/lambda] #obten los tiempos que cumplen la condición de aceptación
Ns <- 1:length(ss) # Conteo
if(pic==T){
plot(ss, Ns, type = "s", xlab = "epocas", ylab = "N(t)",
main = "Simulacion de un proceso Poisson no homogeneo",
sub = expression(lambda(t) == paste(lambdat2,"(t)")))
}
return(list(epocas = ss, cuenta=  Ns))
}

poisson.nohomogeneo(lambdat2,510)
```

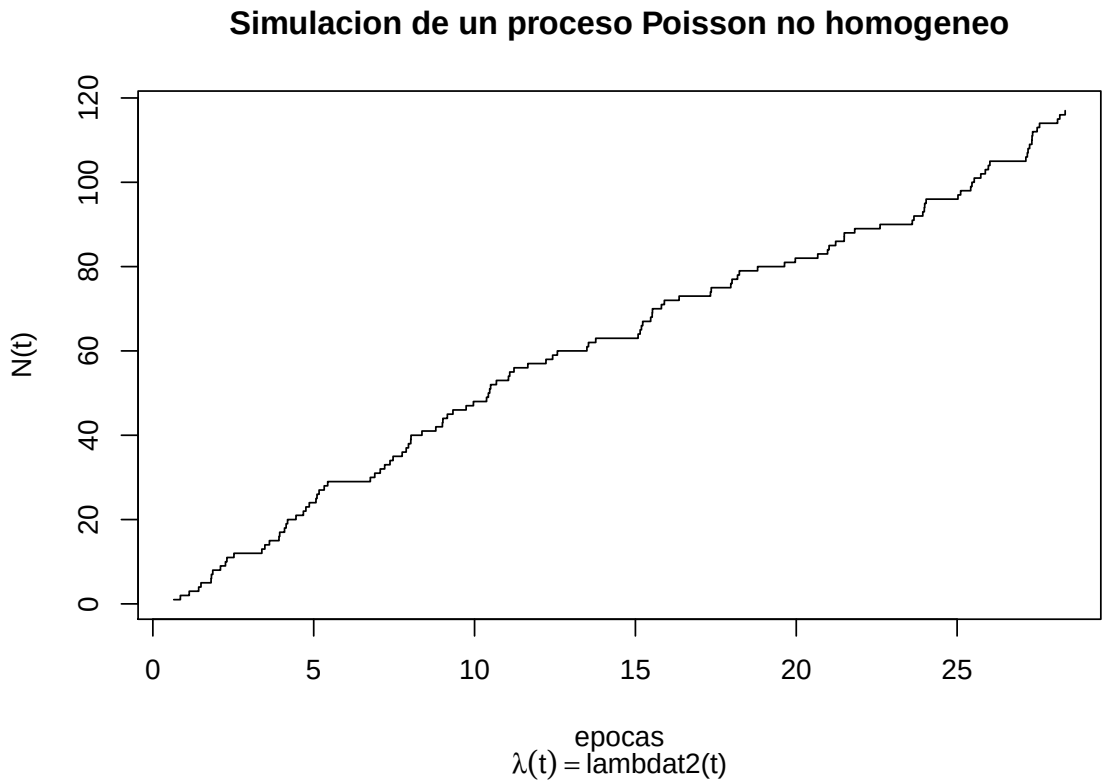



```
$epocas
  [1]  0.09647535  0.19966398  1.26534055  1.73859098  1.83836290
  [6]  1.91924298  1.94714227  2.18412605  2.22151653  2.25276250
 [11]  2.36560841  2.76687025  3.47025866  3.60490732  3.90627729
 [16]  4.36191440  5.15781657  5.17913376  5.21587546  5.27973665
 [21]  5.34572936  5.70154928  6.22644203  6.34304028  6.42808867
 [26]  6.44519970  6.56103652  6.67976302  7.06519288  7.15709343
 [31]  7.27395111  7.36573581  7.40670312  7.48474501  7.51332029
 [36]  7.52843210  7.68913048  7.74600532  7.89424526  7.99528757
 [41]  8.06275163  8.10099840  8.14956107  8.19377332  8.36418368
 [46]  8.41997070  8.78261798  8.85499836  9.16263774  9.17778107
 [51]  9.29533823  9.50840638  9.55017569  9.75908610 10.07664931
 [56] 10.69499772 10.75341852 11.17050162 11.48302123 11.62338643
 [61] 11.72448330 11.78015561 11.81215776 12.71476134 12.82464931
 [66] 13.24159613 13.37812559 13.65775829 13.67199527 13.67341926
 [71] 14.27877947 14.35401069 14.60486129 14.70810560 15.27133923
 [76] 15.60146160 15.98545790 16.06340889 16.46881655 16.86153619
 [81] 17.72231284 17.91928487 18.28577827 18.46902930 18.99252086
 [86] 19.07010660 19.08698892 19.23014749 19.89803555 21.04287112
 [91] 21.07537249 21.10110562 21.10913489 21.32670883 21.46772331
 [96] 21.51996107 22.25142971 22.38865223 22.76070820 22.77009554
[101] 23.01128927 23.10992321 23.13906582 23.56196005 23.72792909
[106] 24.29509361 24.51836132 24.62065320 24.71905205 25.47634806
[111] 25.68425122 25.84748009 26.46443823 26.59305833 26.65327292
[116] 26.94334908 27.60822028 27.63350785 27.81402614 27.94765614
[121] 27.99333804 28.28388394 28.83505913 28.90959772 29.08300428
[126] 29.27831477 29.67849236 29.70026984 29.86327730 29.99480582
[131] 30.44565856 30.98257553 31.12111729 31.36280920 31.86514334
[136] 31.91824806 32.21658254 32.24436160 32.77005267 32.80942102
[141] 33.27703347 33.30249350 33.42492587 33.55335652 33.74118740
[146] 34.61302969 34.80572906 34.93015530 34.93854522 35.05055890
[151] 35.49607528 35.53774644 35.70741608 35.90241525 36.11112715
[156] 36.15680388 37.07526410 37.80924105 38.05570896 38.44584466
[161] 39.23725611 39.45790330 39.65176183 39.73244728 39.76553302
[166] 40.14039849 40.25341392 40.82977869 40.83534514 41.23328528
[171] 41.33105777 41.69953921 41.81459068 41.94005851 42.27664278
[176] 42.29276813 42.45863073 42.81917816 43.06255071 43.42418520
[181] 43.80633070 44.22037794 44.65819557 44.76204262 44.92396635
[186] 45.17664385 45.81054911 47.02462049 47.12772046 47.25243847
[191] 47.34129667 47.50627744 47.51195792 47.64218108 47.68684453
[196] 47.85782118 47.86112380 47.93421813 48.19417722 48.97266180
[201] 49.03786401 49.31104332 49.45213927 49.70716688 50.58867883
```

```
[206] 50.59780242 51.06486697 51.17260092 51.22421987 51.29506197
[211] 51.51357762 51.87379355 51.88468333 51.89488775 51.96005138
[216] 52.03356511 52.36458590 52.95608845 52.96965049 53.03518384
[221] 53.10663297 53.12694077 53.24033986 53.28311786 53.52003843
[226] 53.65606217 53.86176594 53.92803423 54.03729453 54.30832700
[231] 54.62019725 54.71735623 54.94601517 54.97374605 55.00451357
[236] 55.03940882 55.29593233 55.43481277 55.54809836 55.63926470
[241] 55.97140015 56.08272890 56.26638132 57.14774987 57.27484850
[246] 57.41440909 57.44173348 57.45125301 57.72921610 58.05451022
[251] 58.68049894 59.18736845 59.54688129 59.55535207 59.60033842
[256] 59.60224442 59.78221198 59.94472523 60.54387446 60.80169316
[261] 60.84578982 60.97380633 61.33598570 61.61700368 61.94348153
[266] 61.96585774 62.02317632 62.24890432 62.69749844 62.89592194
[271] 63.43328042 63.63831101 63.71073074 63.79531760 63.94535061
[276] 64.21031654 64.32880542 64.95959734 65.03572799 65.23311129
[281] 65.23978308 65.28735072 65.49877015 65.58348637 66.11477615
[286] 66.62827449 66.85534203 67.20249057 67.23535873 67.31931896
[291] 67.43495251 67.53608483 68.29645135 68.54303770 69.02733536
[296] 69.16046075 69.52802958 69.57313170 69.63927641 69.78677786
[301] 70.09647074 70.81449547 70.93057853 71.12796742 71.28652365
[306] 71.47688699 71.49854520 71.66525580 71.76162395 72.19309852
[311] 72.40392040 72.45295744 72.76740950 72.81254658 74.41070149
[316] 74.98122199 75.46860695 75.61175962 75.71132510 75.95033174
[321] 75.95522506 75.97936608 76.70817895 76.93306885 77.26750875
[326] 77.50139455 77.62323666 77.86495280 78.42210800 78.66395790
[331] 78.75179186 79.10972710 79.23253674 79.31533740 79.44265784
[336] 79.60938605 79.76560380 79.79250502 79.93310496 79.94804258
[341] 81.06184368 81.10751607 81.24366762 81.25986408 81.52944782
[346] 81.55640184 81.69729539 81.71215561 81.92536862 82.19197822
[351] 82.43155946 82.71014828 83.26542440 83.31869150 83.46617070
[356] 83.55408406 83.79463674 84.31975417 84.62574458 84.87184015
[361] 85.18412319 85.59704695 85.92400406 86.05654608 86.22046690
[366] 86.80219988 87.05372513 87.54866885 87.72599094 87.80935395
[371] 87.89648650 88.13238959 89.39022241 89.52238559 89.90809255
[376] 90.45425700 90.70962906 90.77762436 90.77928854 91.02846058
[381] 91.35902778 91.62592668 91.69517340 92.57709536 93.23780883
[386] 93.37519285 93.45178891 93.47469338 93.51990060 93.91147857
[391] 93.98933483 94.00014645 94.09080247 94.50348817 94.56046888
[396] 94.64595541 95.07167292 95.56821097 95.67754062 95.81070539
[401] 96.01283380 96.53073175 97.47036947 97.58902107 97.76293548
[406] 97.80595370 97.85620601 97.95484100 98.55162484 99.05921421

$cuenta
  [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17
 [18]  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34
 [35]  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51
 [52]  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68
 [69]  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85
 [86]  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100 101 102
[103] 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119
[120] 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136
[137] 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153
[154] 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170
[171] 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187
[188] 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204
[205] 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221
[222] 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238
[239] 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255
[256] 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272
[273] 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289
[290] 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306
[307] 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323
[324] 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340
[341] 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357
[358] 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374
[375] 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391
[392] 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408
[409] 409 410
```

Para obtener 100 eventos es similar, con un tamaño de muestra mucho menor:

```
poisson.nohomogeneo(lambdat2,140)
```




```
$epocas
  [1]  0.6495207  0.8551147  1.1306859  1.4245332  1.4960867  1.8099938
  [7]  1.8215831  1.8627243  2.1013309  2.2575760  2.3019134  2.5208716
 [13]  3.3880776  3.4816431  3.6229160  3.9211407  3.9431485  4.0927503
 [19]  4.1426237  4.1873363  4.4495045  4.6788720  4.7575189  4.8622701
 [25]  5.0711121  5.1026157  5.1673765  5.3261923  5.4349111  6.7602805
 [31]  6.8976522  7.0679406  7.2057852  7.3717514  7.4674499  7.7499314
 [37]  7.8747064  7.9454235  8.0178587  8.0245204  8.3646546  8.7959443
 [43]  9.0007302  9.0159249  9.1540343  9.3285364  9.7378816  9.9648822
 [49] 10.3735202 10.4293317 10.4703660 10.4992405 10.6792510 11.0544583
 [55] 11.0932237 11.2261474 11.6565272 12.2199936 12.4278841 12.5737221
 [61] 13.4923461 13.5420265 13.7667056 15.0839688 15.1475876 15.1878354
 [67] 15.2276129 15.4747643 15.5216128 15.5302446 15.8082062 15.9001612
 [73] 16.3592022 17.3334965 17.3577265 17.9653519 18.0032273 18.1747671
 [79] 18.2305200 18.7985522 19.6351224 19.9704809 20.6683124 20.9736992
 [85] 21.0204416 21.2247149 21.4904578 21.4921751 21.8169427 22.6056892
 [91] 23.6070184 23.6583153 23.9333978 23.9802925 23.9929013 24.0365230
 [97] 25.0269937 25.1095131 25.4232099 25.4616524 25.5325810 25.7364999
[103] 25.8767730 25.9713839 26.0193421 27.1375817 27.1832063 27.2056717
[109] 27.2534219 27.3235952 27.3303713 27.3477890 27.4883544 27.5624223
[115] 28.1226934 28.1914793 28.3592044

$cuenta
  [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17
 [18]  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34
 [35]  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51
 [52]  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68
 [69]  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85
 [86]  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100 101 102
[103] 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117
```

Por último, para estimar la probabilidad en el intervalo dado, lo que podemos hacer es obtener N simulaciones, y calcular la proporción de esas simulaciones que dan un valor de conteo mayor a 2 en ese intervalo.

```
fr <- NULL
N <- 10000
for(i in 1:N){
x <- poisson.nohomogeneo(lambdat2,10,pic=F);
fr <- c(fr,ifelse(1.25 < x$epocas[x$cuenta==2] & x$epocas[x$cuenta==2] <3,1,0))}
sum(fr)/N
```

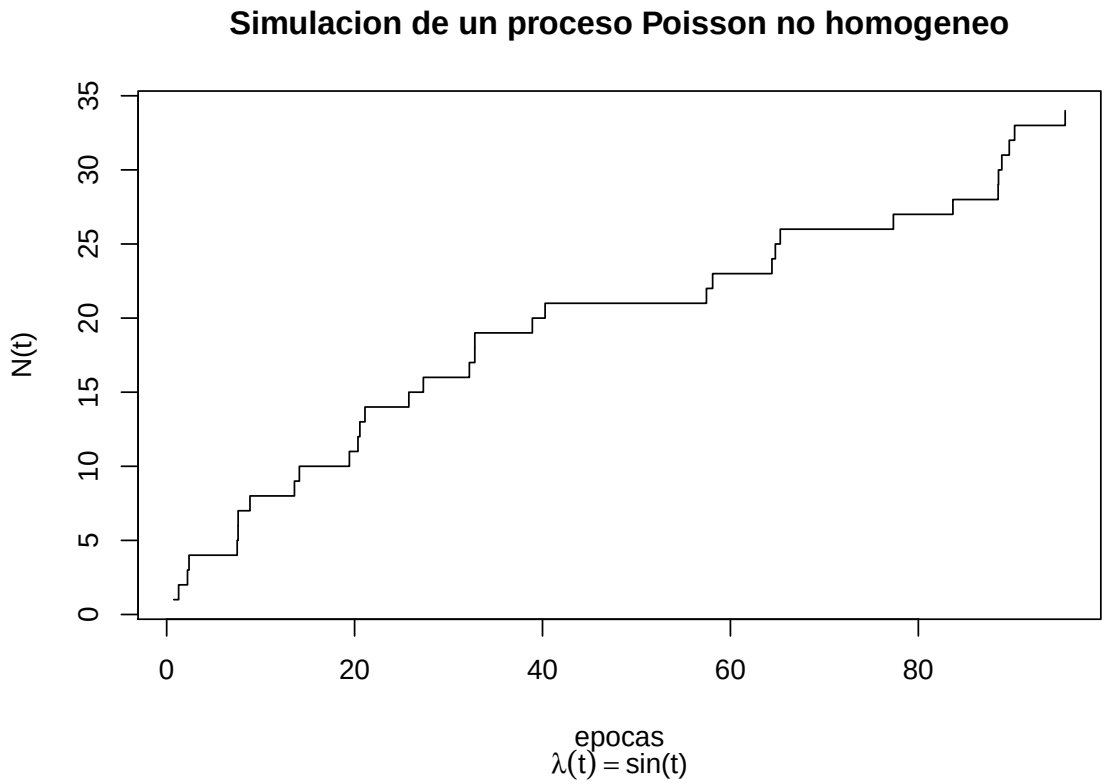
```
[1] 0.0731
```

□

9. Simular un proceso Poisson no homogéneo con función de intensidad dada por $\lambda(t) = sen(t)$

Solución.

```
lambdat <- function(t){sin(t)}

poisson.nohomogeneo<-function(lambdat,n){

lambda <- 1 #mayoriza la función lambdat
TT <- rexp(n,lambda) #genera variables exponenciales para los tiempos.
s <- cumsum(TT) #acumula los tiempos en el vector s
u <- runif(n) #obten n uniformes
ss <- s[u <= lambdat(s)/lambda] #obten los tiempos que cumplen la condición de aceptación
Ns <- 1:length(ss) # Conteo
plot(ss, Ns, type = "s", xlab = "epocas", ylab = "N(t)",
main = "Simulacion de un proceso Poisson no homogeneo",
sub = expression(lambda(t) == paste("sin","(t)")))
return(list(epocas = ss, cuenta=  Ns))
}
x <- poisson.nohomogeneo(lambdat,100)
```




```
x


$epocas
 [1]  0.7542586  1.2729743  2.2167611  2.3792362  7.5077376  7.6003239
 [7]  7.6075017  8.8673948 13.5989677 14.1225588 19.4474747 20.3609274
[13] 20.5634877 21.1082388 25.7737170 27.3197314 32.2146386 32.7947299
[19] 32.7953502 38.9191685 40.2776613 57.4490622 58.1078004 64.4179828
[25] 64.7828805 65.3042308 77.3450111 83.6799027 88.4953600 88.5330879
[31] 88.8863265 89.6778599 90.2479395 95.6279800

$cuenta
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
[24] 24 25 26 27 28 29 30 31 32 33 34
```

□

10. Una compañía de seguros tiene 1000 asegurados, cada uno de los cuales presentará de manera independiente una reclamación en el siguiente mes con probabilidad $p = 0.09245$. Suponiendo que las cantidades de los reclamos hechos son variables aleatorias normales con media \$7,000 y desviación estándar \$5000, hagan simulación para estimar la probabilidad de que la suma de los reclamos exceda \$500,000.

Solución.

En este ejercicio está claro que la reclamación presentada es una variable Bernoulli con probabilidad $p = 0.05$, y el total de reclamos es Binomial con parámetros 1000 y 0.05. Si la media de los reclamos es \$8,000, entonces el reclamo total promedio es

$8000Y$ donde $Y \sim Bin(1000, 0.05)$. Nos piden calcular $P(8000Y > 500000)$ o $P(Y > 500000/8000) = P(Y > 62.5)$.

```
N <- 1e6
y <- rbinom(N,1000,0.05)
Prob <- length(which(y>62.5))/N
Prob


[1] 0.037954


#En este caso podemos conocer el valor exacto de la probabilidad
pbinom(62.5,1000,0.05,lower.tail=F)


[1] 0.03839324
```

□

11. Escribir una función para generar una mezcla de una distribución normal multivariada con dos componentes con medias μ_1 y μ_2 y matrices de covarianzas S_1 y S_2 respectivamente.

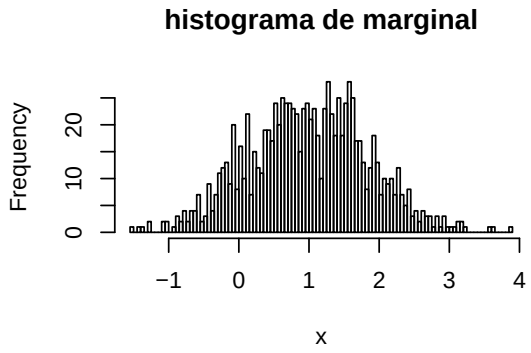
    a. Con el programa, generar una muestra de tamaño $n = 1000$ observaciones de una mezcla $50\,\%$ de una normal 4-dimensional con $\mu_1 = (0, 0, 0, 0)$ y $\mu_2 = (2, 3, 4, 5)$, y matrices de covarianzas $S_1 = S_2 = I_4$.

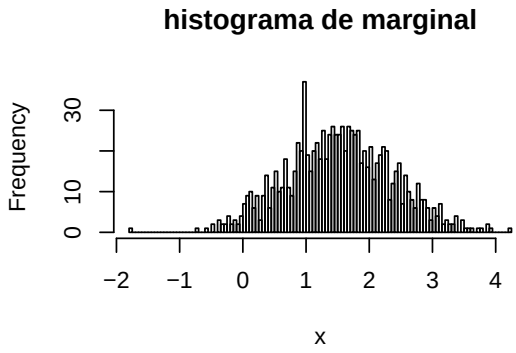
    - Obtener los histogramas de las 4 distribuciones marginales.

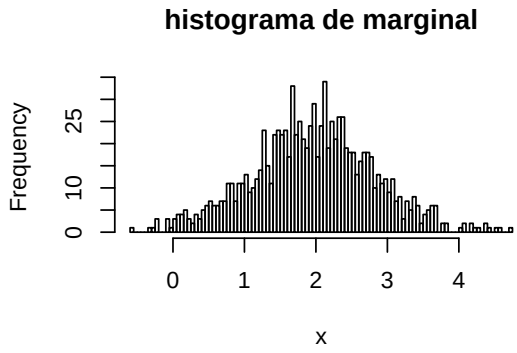
Solución.

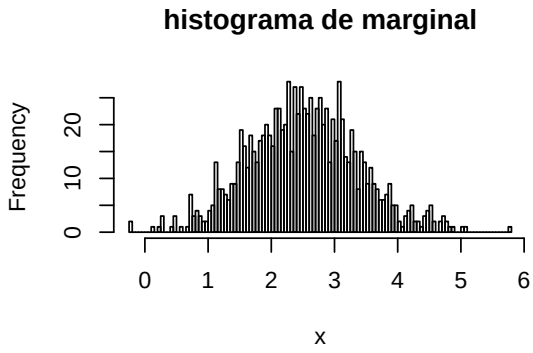
El siguiente script hace lo solicitado. Aquí estoy usando la función `mvrnorm` del paquete `MASS`, pero también se puede usar el método de Box-Müller para generar los valores normales de los vectores.

```
library(MASS)

mezclaNor <- function(n,mu1,mu2,S1,S2,p){
#Esta función genera una mezcla de dos normales p*N(mu1,S1) + (1-p)*N(mu2,S2)
p*mvrnorm(n,mu=mu1,Sigma=S1) + (1-p)*mvrnorm(n,mu=mu2,Sigma=S2)
}

mu1 <- c(0,0,0,0)
mu2 <- c(2,3,4,5)
S1 <- diag(4)
S2 <- 2*diag(4)
p <- 0.5

Z <- mezclaNor(1000,mu1,mu2,S1,S2,p)
par(mfrow=c(2,2))
apply(Z,2,hist,breaks=100, main=paste("histograma de marginal"),xlab="x")
```







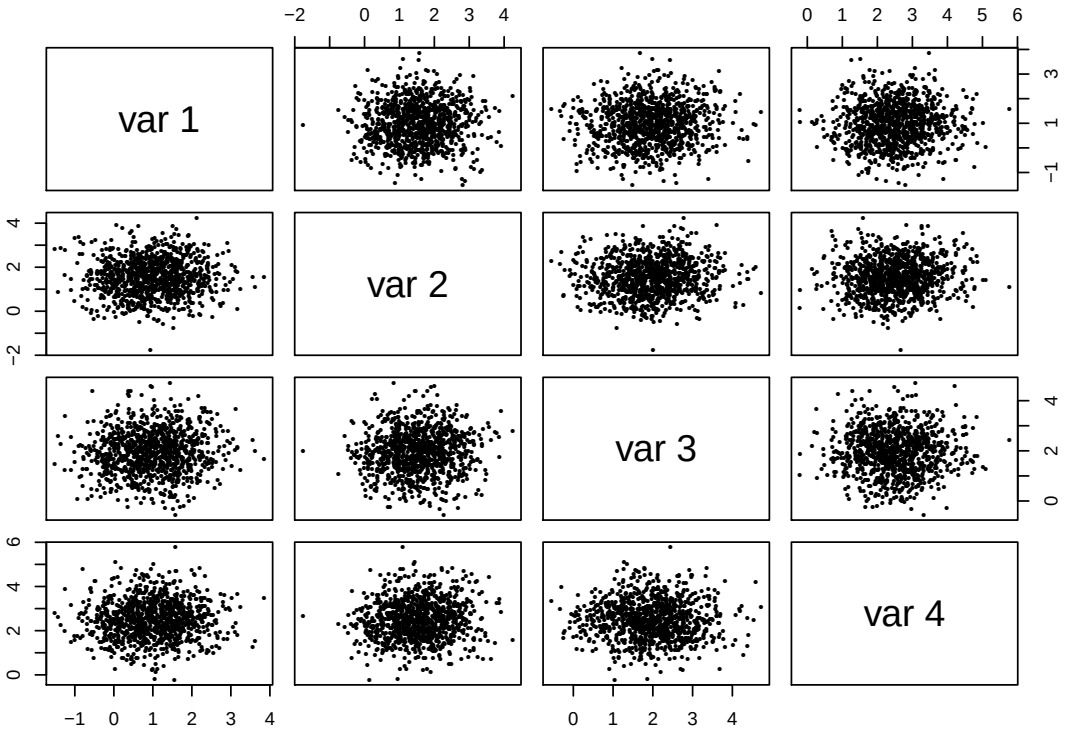



```
[[1]]
$breaks
  [1] -1.55 -1.50 -1.45 -1.40 -1.35 -1.30 -1.25 -1.20 -1.15 -1.10 -1.05
 [12] -1.00 -0.95 -0.90 -0.85 -0.80 -0.75 -0.70 -0.65 -0.60 -0.55 -0.50
 [23] -0.45 -0.40 -0.35 -0.30 -0.25 -0.20 -0.15 -0.10 -0.05  0.00  0.05
 [34]  0.10  0.15  0.20  0.25  0.30  0.35  0.40  0.45  0.50  0.55  0.60
 [45]  0.65  0.70  0.75  0.80  0.85  0.90  0.95  1.00  1.05  1.10  1.15
 [56]  1.20  1.25  1.30  1.35  1.40  1.45  1.50  1.55  1.60  1.65  1.70
 [67]  1.75  1.80  1.85  1.90  1.95  2.00  2.05  2.10  2.15  2.20  2.25
 [78]  2.30  2.35  2.40  2.45  2.50  2.55  2.60  2.65  2.70  2.75  2.80
 [89]  2.85  2.90  2.95  3.00  3.05  3.10  3.15  3.20  3.25  3.30  3.35
[100]  3.40  3.45  3.50  3.55  3.60  3.65  3.70  3.75  3.80  3.85  3.90

$counts
  [1]  1  0  1  1  0  2  0  0  0  2  2  0  1  3  2  4  2  4  4  7  2  3  9
 [24]  4  7 11 12 13  9 20  8 16 10 22  7 15 12 11 19 19 17 24 20 25 24 24
 [47] 23 22 15 23 24 21 23 18 10 23 28 22 18 25 18 24 28 25 17 17 13  8 12
 [70] 18 13  7 10  9 10  7 12  7  5  8  3  4  2  4  3  3  1  3  1  3  1  1
 [93]  1  2  2  1  0  0  0  0  0  0  1  1  0  0  0  0  1

$density
  [1] 0.02 0.00 0.02 0.02 0.00 0.04 0.00 0.00 0.00 0.04 0.04 0.00 0.02 0.06
 [15] 0.04 0.08 0.04 0.08 0.08 0.14 0.04 0.06 0.18 0.08 0.14 0.22 0.24 0.26
 [29] 0.18 0.40 0.16 0.32 0.20 0.44 0.14 0.30 0.24 0.22 0.38 0.38 0.34 0.48
 [43] 0.40 0.50 0.48 0.48 0.46 0.44 0.30 0.46 0.48 0.42 0.46 0.36 0.20 0.46
 [57] 0.56 0.44 0.36 0.50 0.36 0.48 0.56 0.50 0.34 0.34 0.26 0.16 0.24 0.36
 [71] 0.26 0.14 0.20 0.18 0.20 0.14 0.24 0.14 0.10 0.16 0.06 0.08 0.04 0.08
 [85] 0.06 0.06 0.02 0.06 0.02 0.06 0.02 0.02 0.02 0.04 0.04 0.02 0.00 0.00
 [99] 0.00 0.00 0.00 0.00 0.02 0.02 0.00 0.00 0.00 0.00 0.02

$mids
  [1] -1.525 -1.475 -1.425 -1.375 -1.325 -1.275 -1.225 -1.175 -1.125 -1.075
 [11] -1.025 -0.975 -0.925 -0.875 -0.825 -0.775 -0.725 -0.675 -0.625 -0.575
 [21] -0.525 -0.475 -0.425 -0.375 -0.325 -0.275 -0.225 -0.175 -0.125 -0.075
 [31] -0.025  0.025  0.075  0.125  0.175  0.225  0.275  0.325  0.375  0.425
 [41]  0.475  0.525  0.575  0.625  0.675  0.725  0.775  0.825  0.875  0.925
 [51]  0.975  1.025  1.075  1.125  1.175  1.225  1.275  1.325  1.375  1.425
 [61]  1.475  1.525  1.575  1.625  1.675  1.725  1.775  1.825  1.875  1.925
 [71]  1.975  2.025  2.075  2.125  2.175  2.225  2.275  2.325  2.375  2.425
 [81]  2.475  2.525  2.575  2.625  2.675  2.725  2.775  2.825  2.875  2.925
 [91]  2.975  3.025  3.075  3.125  3.175  3.225  3.275  3.325  3.375  3.425
[101]  3.475  3.525  3.575  3.625  3.675  3.725  3.775  3.825  3.875
```

```
$xname
[1] "newX[, i]"

$equidist
[1] TRUE

attr(,"class")
[1] "histogram"

[[2]]
$breaks
  [1] -1.80 -1.75 -1.70 -1.65 -1.60 -1.55 -1.50 -1.45 -1.40 -1.35 -1.30
 [12] -1.25 -1.20 -1.15 -1.10 -1.05 -1.00 -0.95 -0.90 -0.85 -0.80 -0.75
 [23] -0.70 -0.65 -0.60 -0.55 -0.50 -0.45 -0.40 -0.35 -0.30 -0.25 -0.20
 [34] -0.15 -0.10 -0.05  0.00  0.05  0.10  0.15  0.20  0.25  0.30  0.35
 [45]  0.40  0.45  0.50  0.55  0.60  0.65  0.70  0.75  0.80  0.85  0.90
 [56]  0.95  1.00  1.05  1.10  1.15  1.20  1.25  1.30  1.35  1.40  1.45
 [67]  1.50  1.55  1.60  1.65  1.70  1.75  1.80  1.85  1.90  1.95  2.00
 [78]  2.05  2.10  2.15  2.20  2.25  2.30  2.35  2.40  2.45  2.50  2.55
 [89]  2.60  2.65  2.70  2.75  2.80  2.85  2.90  2.95  3.00  3.05  3.10
[100]  3.15  3.20  3.25  3.30  3.35  3.40  3.45  3.50  3.55  3.60  3.65
[111]  3.70  3.75  3.80  3.85  3.90  3.95  4.00  4.05  4.10  4.15  4.20
[122]  4.25

$counts
  [1]  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0
 [24]  0  1  0  2  0  3  2  2  4  2  3  2  4  7  9 10  6  9  3  9 14  6 11
 [47] 15 10 11 18 11  9 15 22 20 37 19 15 20 22 18 25 18 24 26 24 24 26 20
 [70] 26 25 24 25 17 20 16 21 13 17 19 21 20  8 12 15 17  7 14 10  8  6 12
 [93]  8  8  6  3  6  4  7  2  3  2  2  4  0  3  1  1  1  0  1  1  0  2  1
[116]  0  0  0  0  0  1

$density
  [1] 0.02 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
 [15] 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.02 0.00 0.00 0.02 0.00 0.04 0.00
 [29] 0.06 0.04 0.04 0.08 0.04 0.06 0.04 0.08 0.14 0.18 0.20 0.12 0.18 0.06
 [43] 0.18 0.28 0.12 0.22 0.30 0.20 0.22 0.36 0.22 0.18 0.30 0.44 0.40 0.74
 [57] 0.38 0.30 0.40 0.44 0.36 0.50 0.36 0.48 0.52 0.48 0.48 0.52 0.40 0.52
 [71] 0.50 0.48 0.50 0.34 0.40 0.32 0.42 0.26 0.34 0.38 0.42 0.40 0.16 0.24
 [85] 0.30 0.34 0.14 0.28 0.20 0.16 0.12 0.24 0.16 0.16 0.12 0.06 0.12 0.08
 [99] 0.14 0.04 0.06 0.04 0.04 0.08 0.00 0.06 0.02 0.02 0.02 0.00 0.02 0.02
[113] 0.00 0.04 0.02 0.00 0.00 0.00 0.00 0.00 0.02

$mids
  [1] -1.775 -1.725 -1.675 -1.625 -1.575 -1.525 -1.475 -1.425 -1.375 -1.325
 [11] -1.275 -1.225 -1.175 -1.125 -1.075 -1.025 -0.975 -0.925 -0.875 -0.825
 [21] -0.775 -0.725 -0.675 -0.625 -0.575 -0.525 -0.475 -0.425 -0.375 -0.325
 [31] -0.275 -0.225 -0.175 -0.125 -0.075 -0.025  0.025  0.075  0.125  0.175
 [41]  0.225  0.275  0.325  0.375  0.425  0.475  0.525  0.575  0.625  0.675
 [51]  0.725  0.775  0.825  0.875  0.925  0.975  1.025  1.075  1.125  1.175
 [61]  1.225  1.275  1.325  1.375  1.425  1.475  1.525  1.575  1.625  1.675
 [71]  1.725  1.775  1.825  1.875  1.925  1.975  2.025  2.075  2.125  2.175
 [81]  2.225  2.275  2.325  2.375  2.425  2.475  2.525  2.575  2.625  2.675
 [91]  2.725  2.775  2.825  2.875  2.925  2.975  3.025  3.075  3.125  3.175
[101]  3.225  3.275  3.325  3.375  3.425  3.475  3.525  3.575  3.625  3.675
[111]  3.725  3.775  3.825  3.875  3.925  3.975  4.025  4.075  4.125  4.175
[121]  4.225

$xname
[1] "newX[, i]"

$equidist
[1] TRUE

attr(,"class")
[1] "histogram"

[[3]]
$breaks
  [1] -0.60 -0.55 -0.50 -0.45 -0.40 -0.35 -0.30 -0.25 -0.20 -0.15 -0.10
 [12] -0.05  0.00  0.05  0.10  0.15  0.20  0.25  0.30  0.35  0.40  0.45
 [23]  0.50  0.55  0.60  0.65  0.70  0.75  0.80  0.85  0.90  0.95  1.00
 [34]  1.05  1.10  1.15  1.20  1.25  1.30  1.35  1.40  1.45  1.50  1.55
 [45]  1.60  1.65  1.70  1.75  1.80  1.85  1.90  1.95  2.00  2.05  2.10
 [56]  2.15  2.20  2.25  2.30  2.35  2.40  2.45  2.50  2.55  2.60  2.65
 [67]  2.70  2.75  2.80  2.85  2.90  2.95  3.00  3.05  3.10  3.15  3.20
 [78]  3.25  3.30  3.35  3.40  3.45  3.50  3.55  3.60  3.65  3.70  3.75
 [89]  3.80  3.85  3.90  3.95  4.00  4.05  4.10  4.15  4.20  4.25  4.30
[100]  4.35  4.40  4.45  4.50  4.55  4.60  4.65  4.70  4.75

$counts
  [1]  1  0  0  0  0  1  1  3  0  0  3  1  3  4  4  5  3  2  4  3  5  6  7
 [24]  6  6  7  7 11 11  7 11 11 13  9 10 12 14 23 15 11 22 23 22 23 17 33
 [47] 22 25 21 19 24 29 17 24 34 19 25 21 26 26 19 18 18 13 16 18 18 17 10
 [70] 13 11 12  9 12  7  8  3  7  5  8  6  2  4  5  6  6  0  2  2  0  0  0
 [93]  1  2  0  2  1  1  0  2  1  0  1  1  0  0  1
```

```
$density
  [1] 0.02 0.00 0.00 0.00 0.00 0.02 0.02 0.06 0.00 0.00 0.06 0.02 0.06 0.08
 [15] 0.08 0.10 0.06 0.04 0.08 0.06 0.10 0.12 0.14 0.12 0.12 0.14 0.14 0.22
 [29] 0.22 0.14 0.22 0.22 0.26 0.18 0.20 0.24 0.28 0.46 0.30 0.22 0.44 0.46
 [43] 0.44 0.46 0.34 0.66 0.44 0.50 0.42 0.38 0.48 0.58 0.34 0.48 0.68 0.38
 [57] 0.50 0.42 0.52 0.52 0.38 0.36 0.36 0.26 0.32 0.36 0.36 0.34 0.20 0.26
 [71] 0.22 0.24 0.18 0.24 0.14 0.16 0.06 0.14 0.10 0.16 0.12 0.04 0.08 0.10
 [85] 0.12 0.12 0.00 0.04 0.04 0.00 0.00 0.00 0.02 0.04 0.00 0.04 0.02 0.02
 [99] 0.00 0.04 0.02 0.00 0.02 0.02 0.00 0.00 0.02

$mids
  [1] -0.575 -0.525 -0.475 -0.425 -0.375 -0.325 -0.275 -0.225 -0.175 -0.125
 [11] -0.075 -0.025  0.025  0.075  0.125  0.175  0.225  0.275  0.325  0.375
 [21]  0.425  0.475  0.525  0.575  0.625  0.675  0.725  0.775  0.825  0.875
 [31]  0.925  0.975  1.025  1.075  1.125  1.175  1.225  1.275  1.325  1.375
 [41]  1.425  1.475  1.525  1.575  1.625  1.675  1.725  1.775  1.825  1.875
 [51]  1.925  1.975  2.025  2.075  2.125  2.175  2.225  2.275  2.325  2.375
 [61]  2.425  2.475  2.525  2.575  2.625  2.675  2.725  2.775  2.825  2.875
 [71]  2.925  2.975  3.025  3.075  3.125  3.175  3.225  3.275  3.325  3.375
 [81]  3.425  3.475  3.525  3.575  3.625  3.675  3.725  3.775  3.825  3.875
 [91]  3.925  3.975  4.025  4.075  4.125  4.175  4.225  4.275  4.325  4.375
[101]  4.425  4.475  4.525  4.575  4.625  4.675  4.725

$xname
[1] "newX[, i]"

$equidist
[1] TRUE

attr(,"class")
[1] "histogram"

[[4]]
$breaks
  [1] -0.25 -0.20 -0.15 -0.10 -0.05  0.00  0.05  0.10  0.15  0.20  0.25
 [12]  0.30  0.35  0.40  0.45  0.50  0.55  0.60  0.65  0.70  0.75  0.80
 [23]  0.85  0.90  0.95  1.00  1.05  1.10  1.15  1.20  1.25  1.30  1.35
 [34]  1.40  1.45  1.50  1.55  1.60  1.65  1.70  1.75  1.80  1.85  1.90
 [45]  1.95  2.00  2.05  2.10  2.15  2.20  2.25  2.30  2.35  2.40  2.45
 [56]  2.50  2.55  2.60  2.65  2.70  2.75  2.80  2.85  2.90  2.95  3.00
 [67]  3.05  3.10  3.15  3.20  3.25  3.30  3.35  3.40  3.45  3.50  3.55
 [78]  3.60  3.65  3.70  3.75  3.80  3.85  3.90  3.95  4.00  4.05  4.10
 [89]  4.15  4.20  4.25  4.30  4.35  4.40  4.45  4.50  4.55  4.60  4.65
[100]  4.70  4.75  4.80  4.85  4.90  4.95  5.00  5.05  5.10  5.15  5.20
[111]  5.25  5.30  5.35  5.40  5.45  5.50  5.55  5.60  5.65  5.70  5.75
[122]  5.80

$counts
  [1]  2  0  0  0  0  0  0  1  0  1  3  0  0  1  3  0  1  0  1  7  3  4  3
 [24]  2  2  4  5 13  8  8  7  6  9  9 13 19 16 12 18 15 13 17 18 20 18 16
 [47] 23 23 19 20 28 15 27 22 27 23 22 25 18 23 25 20 23 13 21 17 28 21 14
 [70] 13 19 15  8 15 13  9 12  9  8  6  6  7  9  5  5  2  1  3  4  5  2  2
 [93]  1  3  4  5  2  0  2  3  2  1  1  0  0  1  1  0  0  0  0  0  0  0  0
[116]  0  0  0  0  0  1

$density
  [1] 0.04 0.00 0.00 0.00 0.00 0.00 0.00 0.02 0.00 0.02 0.06 0.00 0.00 0.02
 [15] 0.06 0.00 0.02 0.00 0.02 0.14 0.06 0.08 0.06 0.04 0.04 0.08 0.10 0.26
 [29] 0.16 0.16 0.14 0.12 0.18 0.18 0.26 0.38 0.32 0.24 0.36 0.30 0.26 0.34
 [43] 0.36 0.40 0.36 0.32 0.46 0.46 0.38 0.40 0.56 0.30 0.54 0.44 0.54 0.46
 [57] 0.44 0.50 0.36 0.46 0.50 0.40 0.46 0.26 0.42 0.34 0.56 0.42 0.28 0.26
 [71] 0.38 0.30 0.16 0.30 0.26 0.18 0.24 0.18 0.16 0.12 0.12 0.14 0.18 0.10
 [85] 0.10 0.04 0.02 0.06 0.08 0.10 0.04 0.04 0.02 0.06 0.08 0.10 0.04 0.00
 [99] 0.04 0.06 0.04 0.02 0.02 0.00 0.00 0.02 0.02 0.00 0.00 0.00 0.00 0.00
[113] 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.02

$mids
  [1] -0.225 -0.175 -0.125 -0.075 -0.025  0.025  0.075  0.125  0.175  0.225
 [11]  0.275  0.325  0.375  0.425  0.475  0.525  0.575  0.625  0.675  0.725
 [21]  0.775  0.825  0.875  0.925  0.975  1.025  1.075  1.125  1.175  1.225
 [31]  1.275  1.325  1.375  1.425  1.475  1.525  1.575  1.625  1.675  1.725
 [41]  1.775  1.825  1.875  1.925  1.975  2.025  2.075  2.125  2.175  2.225
 [51]  2.275  2.325  2.375  2.425  2.475  2.525  2.575  2.625  2.675  2.725
 [61]  2.775  2.825  2.875  2.925  2.975  3.025  3.075  3.125  3.175  3.225
 [71]  3.275  3.325  3.375  3.425  3.475  3.525  3.575  3.625  3.675  3.725
 [81]  3.775  3.825  3.875  3.925  3.975  4.025  4.075  4.125  4.175  4.225
 [91]  4.275  4.325  4.375  4.425  4.475  4.525  4.575  4.625  4.675  4.725
[101]  4.775  4.825  4.875  4.925  4.975  5.025  5.075  5.125  5.175  5.225
[111]  5.275  5.325  5.375  5.425  5.475  5.525  5.575  5.625  5.675  5.725
[121]  5.775

$xname
[1] "newX[, i]"

$equidist
[1] TRUE
```

```
attr(,"class")
[1] "histogram"


pairs(Z,pch=16,cex=0.5)
```



□

12. **Distribución de Wishart**. Suponer que $M = X'X$, donde X es una matriz $n \times d$ de una muestra aleatoria de una distribución $\mathcal{N}_d(\mu, \Sigma)$. Entonces M tiene una distribución Wishart con matriz de escala Σ y n grados de libertad, y se denota $W \sim W_d(\Sigma, n)$. Cuando $d = 1$, los elementos de X son una muestra aleatoria de una $\mathcal{N}(\mu, \sigma^2)$, por lo que $W_1(\sigma^2, n) \sim \sigma^2 \chi^2_{(n)}$.

    - Una forma de generar observaciones de una distribución Wishart, es generar muestras de multivariadas normales y calcular la matriz producto XX'. Programar este método. Noten que este método es muy costoso porque se tienen que generar nd valores aleatorios normales para determinar las $d(d+1)/2$ diferentes entradas de M.
    - Un método más eficiente se basa en la descomposición de Bartlett: sea $T = (T_{ij})$ una matriz triangular inferior de $d \times d$ con entradas independientes que satisfacen

a) $T_{ij} \sim \mathcal{N}(0,1)$ independientes, para $i > j$.

b) $T_{ii} \sim \sqrt{\chi^2_{(n-i+1)}}$, $i = 1, \ldots, d$.

Entonces la matriz $A = TT'$ tiene una distribución Wishart $W_d(I_d, n)$. Para generar variables $W_d(\Sigma, n)$, obtener la descomposición de Choleski $\Sigma = LL'$, donde L es triangular inferior. Entonces $LAL' \sim W_d(\Sigma, n)$. Implementar esta versión.

- Comparar en tiempo de ejecución de ambas versiones.

Solución.

Para la primera parte del ejercicio, definimos la función $W1$ de la siguiente manera:

```
W1 <- function(k, n, mu1, S1){
        #Esta función genera k muestras de la distrbución Wishart de  la forma ineficiente
        #mu1 es un vector de dimensión d, y S1 es una matriz definida positiva de dxd
        require(MASS)
        W <- list(NULL)
        for(i in 1:k){
                X <- mvrnorm(n, mu=mu1, Sigma=S1)
                W[[i]] <- t(X)%*%X
        }
        return(W)
}

mu1 <- rep(0,4)
S1 <- diag(4)
W1(k = 5, n = 10, mu1 = mu1, S1 = S1)
```

```
[[1]]
          [,1]     [,2]     [,3]       [,4]
[1,] 15.9526475 5.599819 4.978384  0.2333088
[2,]  5.5998186 4.935713 2.712445  2.5464027
[3,]  4.9783840 2.712445 7.210953  1.9613178
[4,]  0.2333088 2.546403 1.961318 10.7995699

[[2]]
          [,1]       [,2]      [,3]       [,4]
[1,]  3.323783  1.2325207 1.9423716 -1.1836331
[2,]  1.232521  6.0705111 0.1660781 -0.9505133
[3,]  1.942372  0.1660781 8.4208267  0.5371871
[4,] -1.183633 -0.9505133 0.5371871 10.9399980

[[3]]
          [,1]       [,2]       [,3]      [,4]
[1,] 11.414420 -2.1175969  3.7647759 1.0244707
[2,] -2.117597 11.2647404  0.3459151 3.8404754
[3,]  3.764776  0.3459151 18.9778644 0.9598537
[4,]  1.024471  3.8404754  0.9598537 9.5187788

[[4]]
          [,1]       [,2]       [,3]       [,4]
[1,] 14.7242583 -0.6288252  7.5584778 -0.1947632
[2,] -0.6288252  2.1755701 -1.5024643 -0.3549534
[3,]  7.5584778 -1.5024643 10.5412651  0.6595003
[4,] -0.1947632 -0.3549534  0.6595003  3.6210601

[[5]]
           [,1]      [,2]        [,3]      [,4]
[1,]  2.72487860 -1.040602  0.02814361  1.463719
[2,] -1.04060198  5.418556  3.61758861 -1.129107
[3,]  0.02814361  3.617589  8.47872453 -6.726310
[4,]  1.46371870 -1.129107 -6.72630959 24.719848
```

Para la segunda parte, usamos la transformación sugerida:

```
W2 <- function(k,n,mu1,S1){
        #Esta función genera k muestras de la distribuciń Wishart utilizando la
        #descomposición de Barttlet.
        W <- list(NULL)
        d <- length(mu1)
        M <- matrix(0,nrow=d,ncol=d)
        for(i in 1:k){
                M[lower.tri(matrix(0,nrow=d,ncol=d))] <- rnorm(d*(d+1)/2-d)
                diag(M) <- sqrt(rchisq(d,n-(1:d)-1))
                L <- chol(S1)
                W[[i]] <- L%*%M%*%t(M)%*%t(L)
        }
        return(W)
}

W2(k = 5, n = 10, mu1 = mu1, S1 = S1)
```

```
[[1]]
           [,1]       [,2]       [,3]      [,4]
[1,]  7.2583575 -4.172511 -0.6999384 -2.459920
[2,] -4.1725114 20.003988  4.8819700  1.339806
[3,] -0.6999384  4.881970  5.6885892  1.048807
[4,] -2.4599201  1.339806  1.0488073  3.955921

[[2]]
            [,1]       [,2]        [,3]       [,4]
[1,]  2.87713915 -0.5398107 -0.09680835 -2.8640951
[2,] -0.53981067  4.1324666  0.62368046 -0.9407599
[3,] -0.09680835  0.6236805  7.75373356 -0.9359603
[4,] -2.86409514 -0.9407599 -0.93596028  7.8032390

[[3]]
           [,1]        [,2]       [,3]        [,4]
[1,]  6.5134335 -2.13748063 -0.8123395 -4.89517804
[2,] -2.1374806  4.58031741  2.7404815 -0.01049475
[3,] -0.8123395  2.74048155  7.2666969 -2.72898219
[4,] -4.8951780 -0.01049475 -2.7289822  6.79998788

[[4]]
           [,1]      [,2]      [,3]       [,4]
[1,]  3.9068025 -2.774106  2.098574 -0.7426185
[2,] -2.7741059  6.306793 -1.606789  2.3558365
[3,]  2.0985739 -1.606789  3.462772  1.7850178
[4,] -0.7426185  2.355836  1.785018  9.1288954

[[5]]
          [,1]      [,2]      [,3]      [,4]
[1,] 2.8526699 0.6355580 0.9956038 0.4749397
[2,] 0.6355580 2.7525058 0.2722013 1.2643961
[3,] 0.9956038 0.2722013 8.5509924 0.7002981
[4,] 0.4749397 1.2643961 0.7002981 4.1122706
```

Para comparar los tiempos, generamos una muestra de tamaño grande, digamos $k = 10,000$ (idealmente para hacer una comparación adecuada, hay que hacer una matriz, variando tanto k como n y hasta de d).

```
k <- 10000
system.time(W <- W1(k,10,c(0,0,0,0),diag(4)))
```

```
   user  system elapsed
  0.670   0.000   0.671
```

```
system.time(W <- W2(k,10,c(0,0,0,0),diag(4)))
```

```
   user  system elapsed
  0.304   0.000   0.305
```

Noten que el tiempo es menor, aun cuando tuvimos que construir la matriz T (que yo llamé M) y hacer el producto de varias matrices, y esto es aún cuando estamos usando funciones optimizadas como `mvrnorm`.

□

13. Las ocurrencias de huracanes que tocan tierra durante el fenómeno meteorológico "el Niño" se modelan como un proceso Poisson (ver Bove et al (1998)). Los autores aseguran que "Durante un año de 'El Niño', la probabilidad de dos o más huracanes haciendo contacto con tierra en los estados Unidos es 28 %". Encontrar la tasa del proceso Poisson.

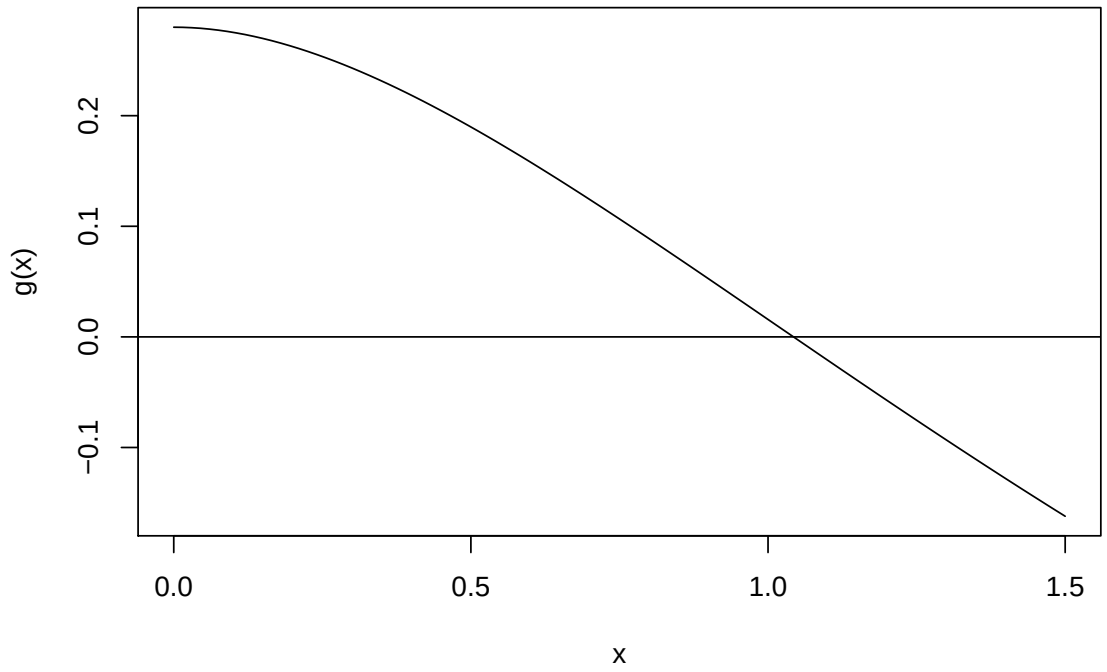
Solución.

Este problema dice que un proceso Poisson con parámetro λ cumple la siguiente condición: $P(N_2 \geq 2) = 0.28$. Entonces tenemos que encontrar λ.

$$\begin{aligned} P(N_2 \geq 2) &= 1 - \sum_{i=0}^{1} \frac{e^{-\lambda}\lambda^i}{i!} \\ &= 1 - e^{-\lambda}(1+\lambda) \\ &= 0.28 \end{aligned}$$

Tenemos que resolver la ecuación: $e^{-\lambda}(1+\lambda) = 0.78$

```
g <- function(x) exp(-x)*(1+x)-0.72
curve(g,0,1.5)
abline(h=0)
```



```
a <- uniroot(g,interval=c(1,1.5))
a


$root
[1] 1.042851

$f.root
[1] -3.173615e-07

$iter
[1] 3

$init.it
[1] NA

$estim.prec
[1] 6.103516e-05
```

Entonces $\lambda = 1.0428507824926$

□

14. Comenzando a mediodía, los comensales llegan a un restaurante de acuerdo a un proceso Poisson a una tasa de 5 clientes por minuto. El tiempo que cada cliente pasa comiendo en el restaurante tiene una distribución exponencial con media de 40 minutos, independiente de otros clientes e independiente de los tiempos de arribo. Encuentra la distribución así como la media y varianza, del número de comensales en el restaurante a las 2:00pm. Simular el restaurante para verificar los resultados obtenidos.

Solución.

En este problema utilizamos un resultado que obtuvimos en relación a las estadísticas de orden de la distribución uniforme y los tiempos de llegada exponenciales con el proceso Poisson. Denotemos con R_t el número de clientes en el restaurante y A_t el proceso Poisson de la llegada de los clientes al tiempo t, Entonces, expresamos la distribución de R_t condicional en A_t:

$$\begin{aligned} P(R_t = k) &= \sum_{i=k}^{\infty} P(R_t = k | A_t = i) P(A_t = i) \\ &= \sum_{i=k}^{\infty} P(R_t = k | A_t = i) \frac{e^{-\lambda t}(\lambda t)^i}{i!}. \end{aligned}$$

Suponemos que hay i comensales en el restaurante al tiempo i que llegaron en los tiempos $S_1, S_2, \ldots, S_i$ Sea L_k el tiempo que pasa el comensal k en el restaurante, para $1 \leq k \leq i$. Entonces de acuerdo al ejercicio, estos tiempos tiene distribución exponencial con parámetro $\beta = 40$ minutos, y los clientes dejan el restaurante en los tiempos $S_1 + L_1, \ldots, S_i + L_i$.

Habrá k clientes en el restaurante en el tiempo t si y sólo si k de los tiempos de salida $S_1 + L_1, \ldots, S_i + L_i$ exceden t. Esto nos da:

$$\begin{aligned} P(R_t = k | A_t = i) &= P(k \text{ de los tiempos } S_1 + L_1, \ldots, S_i + L_i \text{ exceden } t | A_t = i) \\ &= P(k \text{ de } U_{(1)} + L_1, \ldots, U_{(i)} + L_i \text{ exceden } t) \\ &= P(k \text{ de } U_1 + L_1, \ldots, U_i + L_i \text{ exceden } t) \\ &= \binom{i}{k} p^k (1-p)^{i-k} \end{aligned}$$

donde $p = P(U_1 + L_1 > t) = \frac{1}{t}\int_0^t P(L_1 > t - x)dx = \int_0^t (1 - F(x))dx$
Conjuntando los resultados, esto nos da:

$$\begin{aligned} P(R_t = k) &= \sum_{i=k}^{\infty} P(R_t = k | A_t = i) \frac{e^{-\lambda t}(\lambda t)^i}{i!} \\ &= \sum_{i=k}^{\infty} \binom{i}{k} p^k (1-p)^{i-k} \frac{e^{-\lambda t}(\lambda t)^i}{i!} \\ &= \frac{p^k (\lambda t)^k}{k!} \sum_{i=k}^{\infty} \frac{(1-p)^{i-k}(\lambda t)^{i-k}}{(i-k)!} \\ &= \frac{p^k (\lambda t)^k}{k!} e^{\lambda(1-p)t} \\ &= \frac{e^{-\lambda p t}(\lambda p t)^k}{k!} \end{aligned}$$

Así que R_t tiene distribución Poisson con parámetro λp con p como se dijo antes.

En particular, para nuestro problema, y considerando como unidad de tiempo minutos, se tienen los siguientes parámetros: la distribución de Z_t es exponencial con parámetro $\beta = 40$, por lo que $p = -\frac{1}{\beta}(e^{-t/\beta} - 1)$, $t = 120$ (dos horas a partir de mediodía), $\lambda = 5$. Una vez identficado el proceso, es fácil hacer la simulación. La otra manera de pensar la simulación es generar los dos procesos.

□

15.
- Construyan un vector de 100 números crecientes y espaciados regularmente entre 0.1 y 20. Llámenlo `SIG2` . Ahora construyan otro vector de longitud 21 empezando en -1 y terminando en 1. Llámenlo `RHO`.
- Para cada entrada σ^2 de `SIG2` y cada entrada de `RHO`:
  - Generar una muestra de tamaño $N = 500$ de una distribución bivariada normal $Z = (X, Y)$ donde $X \sim \mathcal{N}(0, 1)$ y $Y \sim \mathcal{N}(0, \sigma^2)$ y el coeficiente de correlación de X y Y es ρ. Z es una matriz de dimensiones 500×2.

- Crear una matriz de 500×2, llámenlo `EXPZ`, con las exponenciales de las entradas de Z. ¿Qué distribución tienen estas variables transformadas?
- Calculen el coeficiente de correlación, $\tilde{\rho}$ de las columnas de `EXPZ`. Grafiquen los puntos $(\sigma^2, \tilde{\rho})$ y comenten sobre lo que obtuvieron.

Solución.

Para el primer inciso,

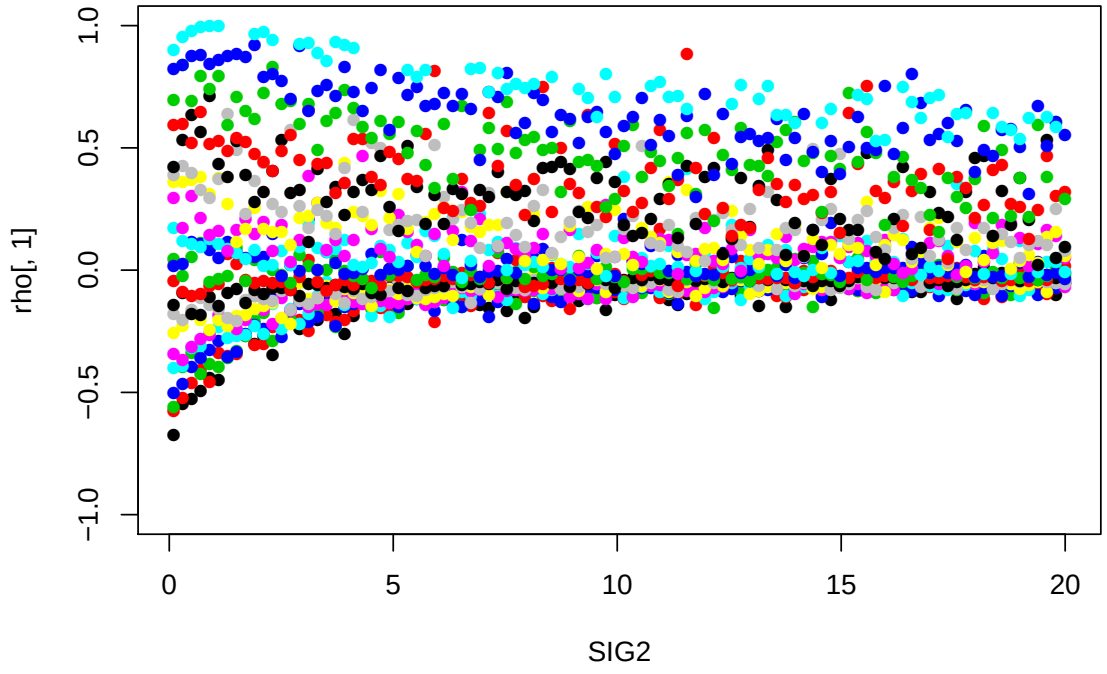
```
SIG2 <- seq(0.1,20,length=100)
RHO <- seq(-1,1,length=21)
```

Para el segundo inciso, generamos las normales utilizando el método de Box-Müller, que de construye con la siguiente función (pueden usar cualquier función que genere normales). Z tiene una distribución lognormal.

```
normalBM <- function(n){
        #genera una muestra de pares de normales independientes de tamaño n.
        u1 <- runif(n)
        u2 <- runif(n)
        R  <- sqrt(-2*log(u1))
        z1 <- R*cos(2*pi*u2)
        z2 <- R*sin(2*pi*u2)
        return(cbind(z1,z2))
}
```

Una vez obtenidos los pares de variables normales, multiplicamos por la matriz B tal que $BB' = \Sigma$, donde $\Sigma = \begin{pmatrix} 1, \rho\sigma \\ \rho\sigma, \sigma^2 \end{pmatrix}$.

```
rho <- matrix(numeric(),nrow=100,ncol=21)

suppressWarnings(
 for(i in SIG2){
        for(j in RHO){
                Sigma <- matrix(c(1,sqrt(i)*j,sqrt(i)*j,i),nrow=2,byrow=T)
                e <- eigen(Sigma)
                B <- e$vectors %*% diag(sqrt(e$values)) %*% t(e$vectors)
                ZEXP <- exp(normalBM(500) %*% B)
                rho[match(i,SIG2),match(j,RHO)] <- cor(ZEXP[,1],ZEXP[,2])
        }
}
) #Algunos puntos no están definidos, para no listar todos los casos.

plot(SIG2,rho[,1],pch=16,col=1,ylim=c(-1,1))

for(i in 2:21)points(SIG2,rho[,i],pch=16,col=i)
```



Lo que se puede observar de la gráfica, son los siguientes características:

- La relación entre correlación y varianza en variables lognormales no es lineal
- Hay valores del coefciente de correlación que no se pueden alcanzar para ciertos valores de la varianza. Por ejemplo, prácticamente para ningún valor de la varianza se puede tomar correlación negativa cercana a -0.5, y conforme la varianza crece, la correlación no puede ser negativa.

□