

Tarea 2.

La fecha de entrega es el lunes **21 de septiembre de 2020**.

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. **Problema de captura-recaptura** Un estadístico está interesado en el número N de peces que hay 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 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.

Este procedimiento de captura y recaptura se conoce como el método de Lincoln-Petersen, y se basa en una medición de la concentración de peces marcados. La idea del procedimiento es la siguiente: Para medir el volumen de agua en una tina, una forma fácil es vaciar un volumen v de líquido de prueba coloreada y dejar que se

mezcle. Luego se toma un volumen del agua mezclada y se mide la concentración $c = \frac{v}{v+V}$ del líquido de prueba. Y de aquí se despeja $V = \frac{v}{c} - v$.

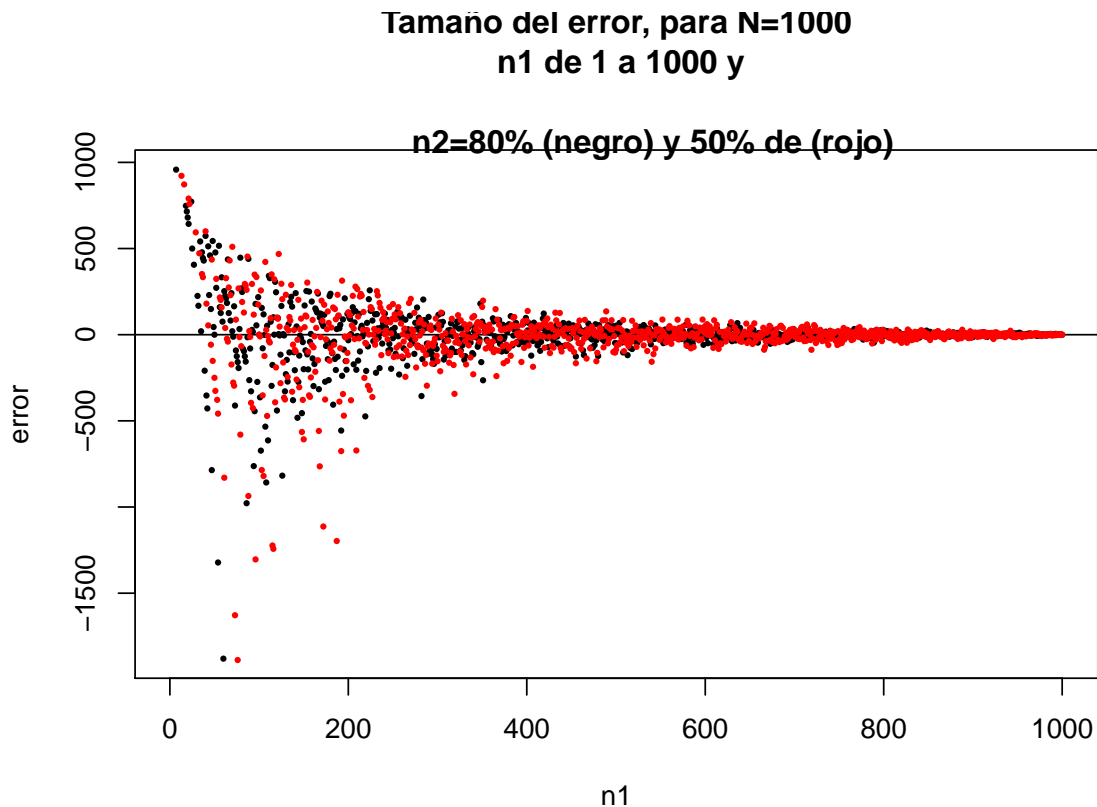
En el caso de los peces, no se agrega un conjunto de peces externos, sino que se hace una primera captura. La primera captura es $n_1 = 250$ y la segunda muestra es $n_2 = 174$ y los marcados son $r = 50$. La estimación de N es entonces $N = \frac{n_1 n_2}{r} = 870$, donde r/n_2 es el equivalente a la concentración c .

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 a los que se enfrentan las compañías de seguros: calcular la probabilidad de quebrar y estimar cuánto dinero podrán hacer. Los parámetros se dan en conjuntos, se pide simular combinaciones de los diferentes parámetros (probabilidad de reclamo, horizonte de tiempo).

Supongan que una compañía de seguros tiene activos por \$ 10^7 de pesos. 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 $p \in \{0.01, 0.1, 0.15, 0.2\}$ 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 $T \in$

$\{5, 10, 20\}$ años. Sea $Z(t)$ los activos de la compañía al final del año t , así que $Z(0) = 10,000,000$, 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 bancarota, 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 en el horizonte de tiempo T y estimen: (1) la probabilidad de que la compañía se vaya a bancarota, 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 $Z(t) > 10,000,000$ entonces $Z(t) - 10,000,000$ se le paga a los accionistas. Si $Z(t) \leq 10,000,000$ 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 T años, y (3) Las ganancias totales esperadas después de T años.

Solución.

El primer ejercicio es estándar en probabilidad. Me concentraré en los últimos dos incisos.

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 , 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.

Para generar distribuciones Pareto, usaré la función en el paquete `actuar`, aunque podrían usar su propio generador.

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

Attaching package: 'actuar'

The following object is masked from 'package:grDevices':

    cm

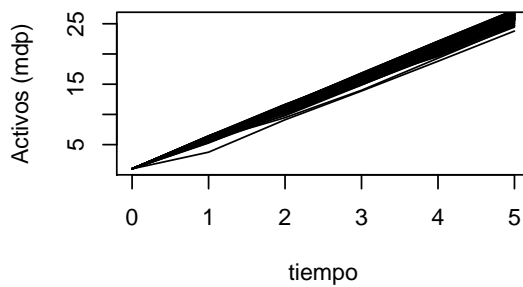
Proceso <- function(ActIni = 1e6, clientes = 1000, t = 5, prima = 5500, p=0.1){
  Z <- NULL
  Z0 <- ActIni
  for(i in 1:t){
    # clientes que generan reclamos, binomial
    reclamos <- sum(rbinom(clientes,1,p))
    Z[i] <- max(ifelse(i==1,Z0,Z[i-1]) + clientes*prima - sum(rpareto(reclamos,3,100000)),0)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

```

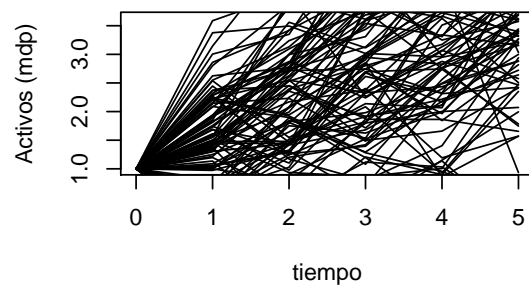
        return(c(Z0,Z))
    }
    for(t0 in c(5,10,20)){
        par(mfrow=c(2,2))
        for(p0 in c(0.01,0.1,0.15,0.20)){
            y <- NULL
            plot(0:t0, Proceso(t=t0,p=p0)/1e6, type = "l",
                main = paste("Corrida para T=",t0," y p=",p0), xlab = "tiempo", ylab = "Activos (mdp)")
            for(i in 1:100){y[[i]] <- Proceso(t=t0,p=p0)/1e6; lines(0:t0,y[[i]])}
            Trayectorias <- matrix(unlist(y), ncol = t0+1, byrow = T)
        }
        head(Trayectorias)
    }
}

```

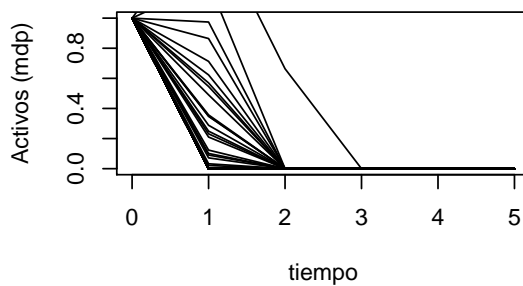
Corrida para T= 5 y p= 0.01



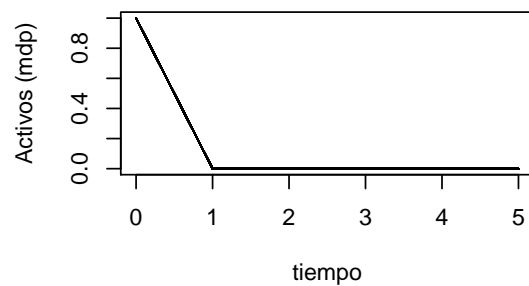
Corrida para T= 5 y p= 0.1



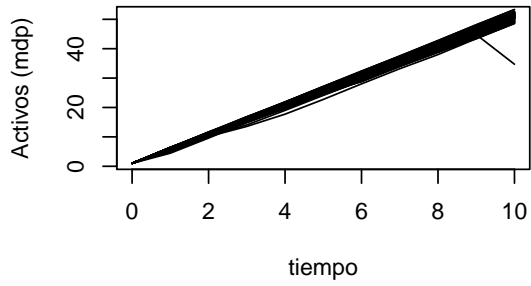
Corrida para T= 5 y p= 0.15



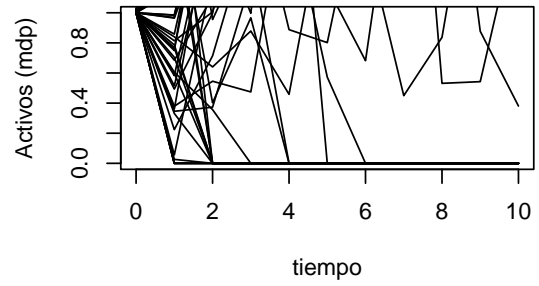
Corrida para T= 5 y p= 0.2



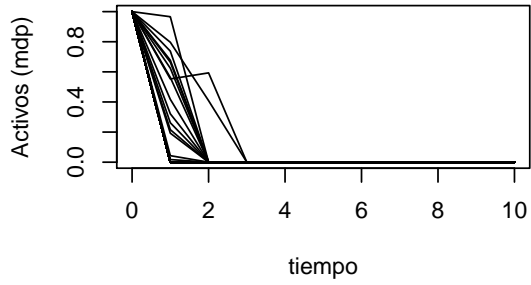
Corrida para $T=10$ y $p=0.01$



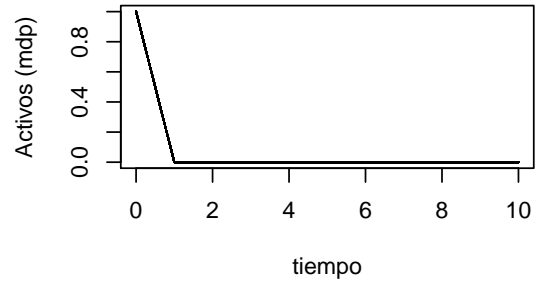
Corrida para $T=10$ y $p=0.1$

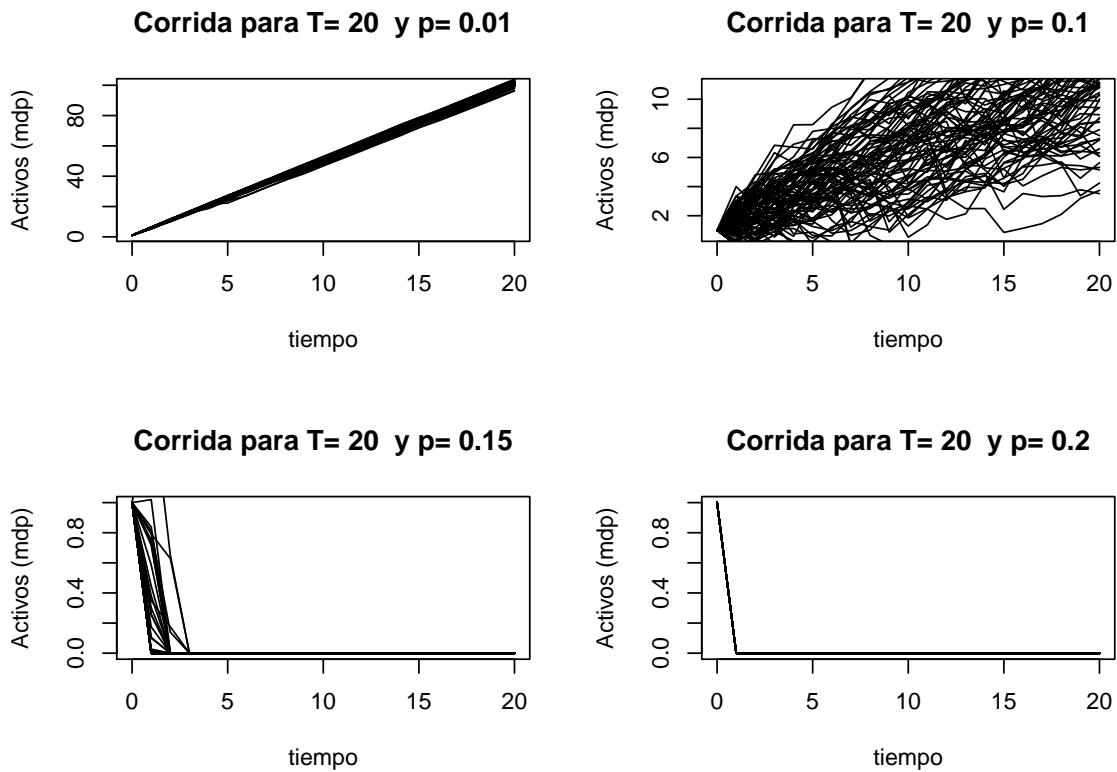


Corrida para $T=10$ y $p=0.15$



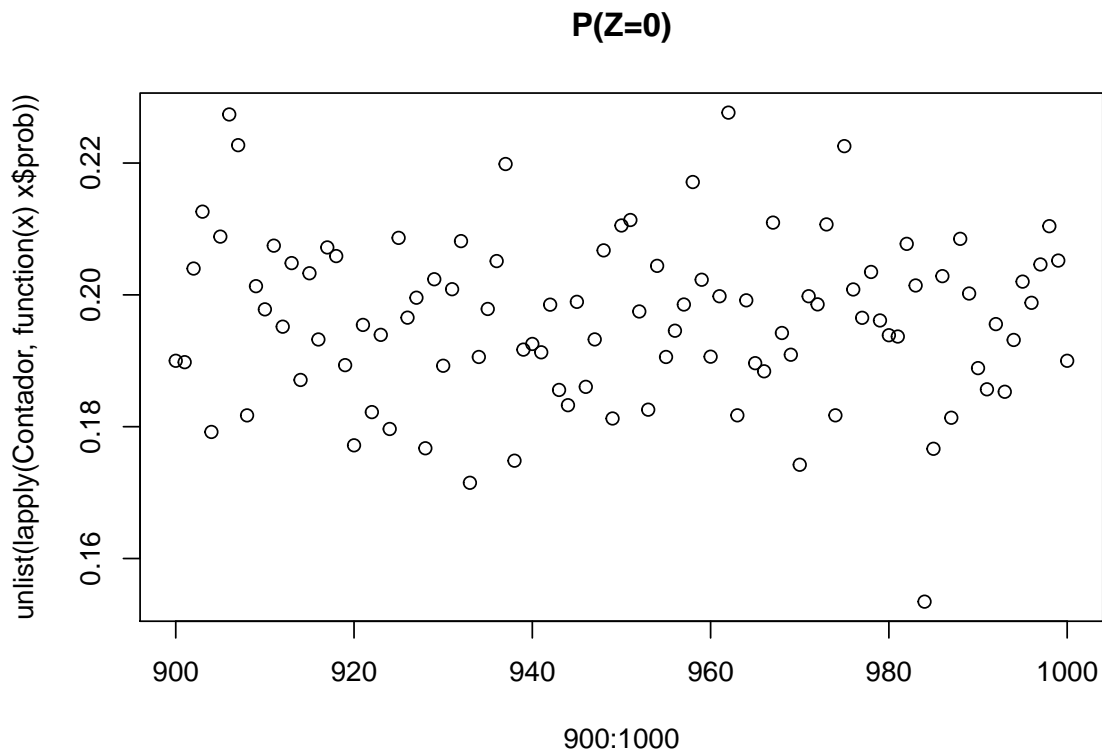
Corrida para $T=10$ y $p=0.2$





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 $T = 20$ años y $p = 0.1$ (habría que hacer todas las combinaciones que se pidieron). 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(t = 20, p = 0.1)
  Tr <- matrix(unlist(Tr), ncol = 21, byrow = T)
  probab <- sum(apply(Tr, 1, function(x) 0 %in% x))/N
  acti <- mean(Tr[,21])
  return(list(probab = probab, activos = acti))
}
Contador <- list(NULL)
for(n in 900:1000) Contador[[n-899]] <- simula(n)
plot(900:1000, unlist(lapply(Contador, function(x)x$probab)), main="P(Z=0) ")
```



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

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
9081614	9388238	9538935	9531896	9645982	10141689

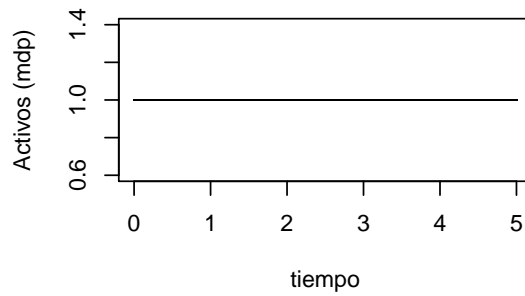
Ahora modificamos el ejercicio para toma de ganancias.

```
#Considera ahora la toma de ganancias.
Proceso2 <- function(ActIni = 1e6, clientes = 1000, t = 5, p = 0.1, prima = 5500){
  Z <- NULL
  G <- NULL #Ganancias
  Z0 <- ActIni
  for(i in 1:t){
    reclamos <- sum(rbinom(clientes,1,p)) #clientes que generan reclamos, binomial
    Z[i] <- max(ifelse(i==1,Z0,Z[i-1]) + clientes*prima - sum(rpareto(reclamos,3,100000)),0)
    #verifica si se alcanzó 0, y todos los consecutivos 0.
    if(Z[i] == 0){Z[i:t] <-0;break}
    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))
}

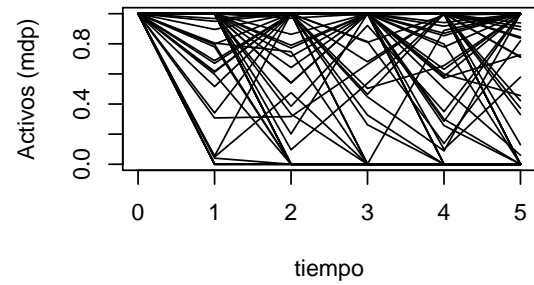
for(t0 in c(5,10,20)){
  par(mfrow = c(2,2))
  for(p0 in c(0.01,0.1,0.15,0.20)){
    y <- NULL
    plot(0:t0, Proceso2(t = t0, p = p0)$Z/1e6, type = "l", xlab = "tiempo", ylab = "Activos (mdp)",
         main = paste("Corrida para T=",t0," y p=",p0))
    for(i in 1:100){y[[i]] <- Proceso2(t = t0, p = p0)$Z/1e6}
    lines(0:t0,y[[i]])
  }
  Trayectorias <- matrix(unlist(y),ncol=t0+1,byrow=T)
  head(Trayectorias)
```


)

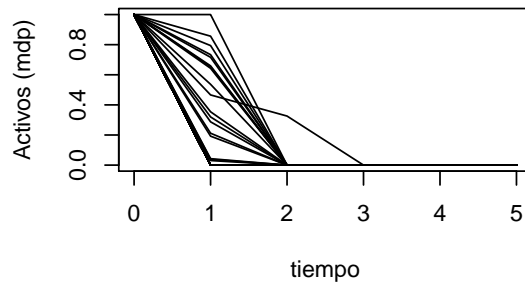
Corrida para $T= 5$ y $p= 0.01$



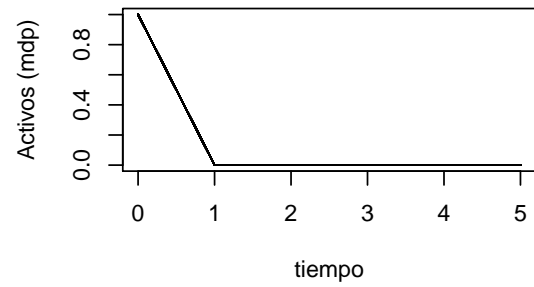
Corrida para $T= 5$ y $p= 0.1$



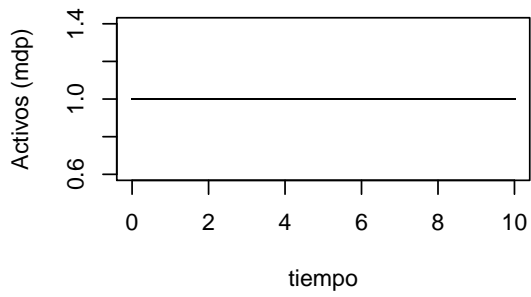
Corrida para $T= 5$ y $p= 0.15$



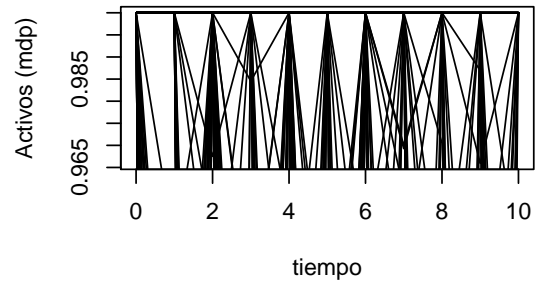
Corrida para $T= 5$ y $p= 0.2$



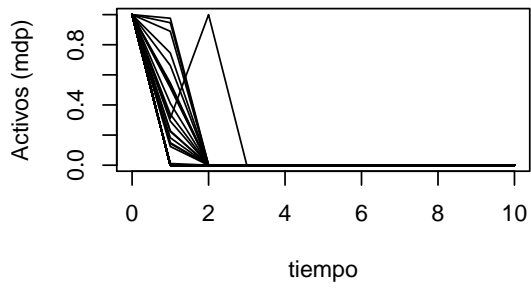
Corrida para $T=10$ y $p=0.01$



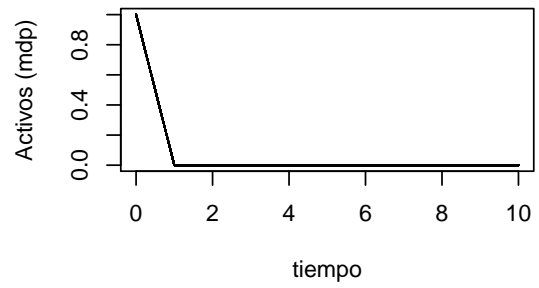
Corrida para $T=10$ y $p=0.1$

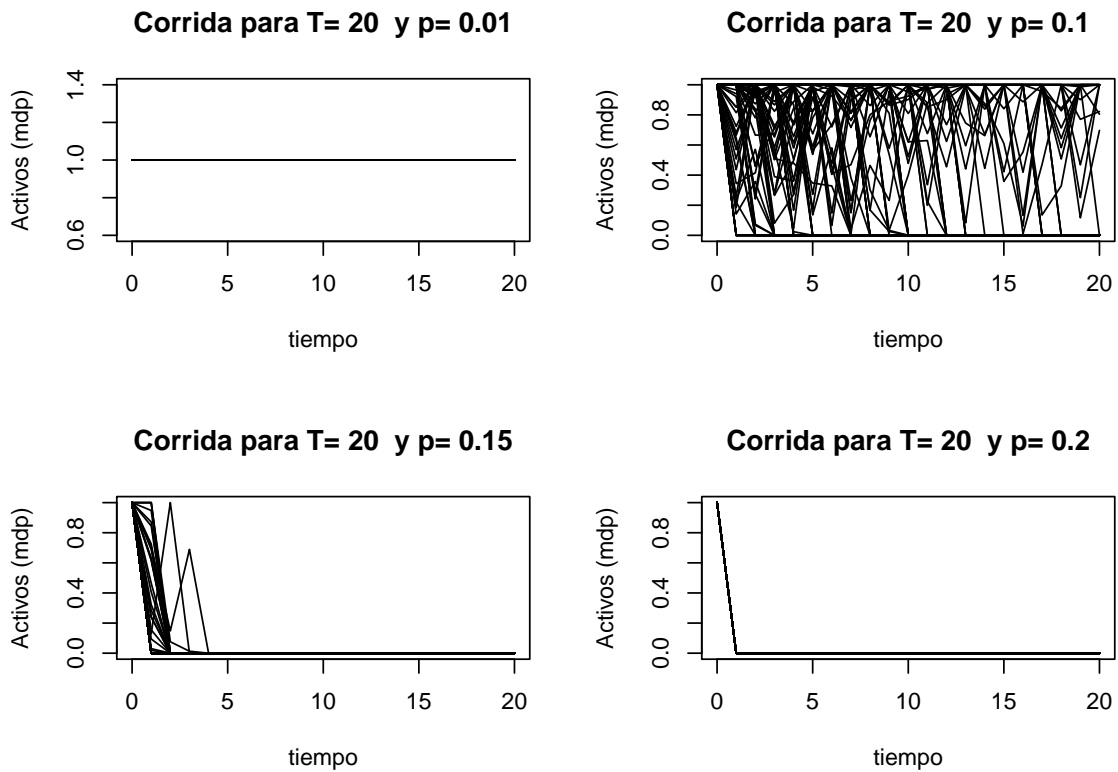


Corrida para $T=10$ y $p=0.15$



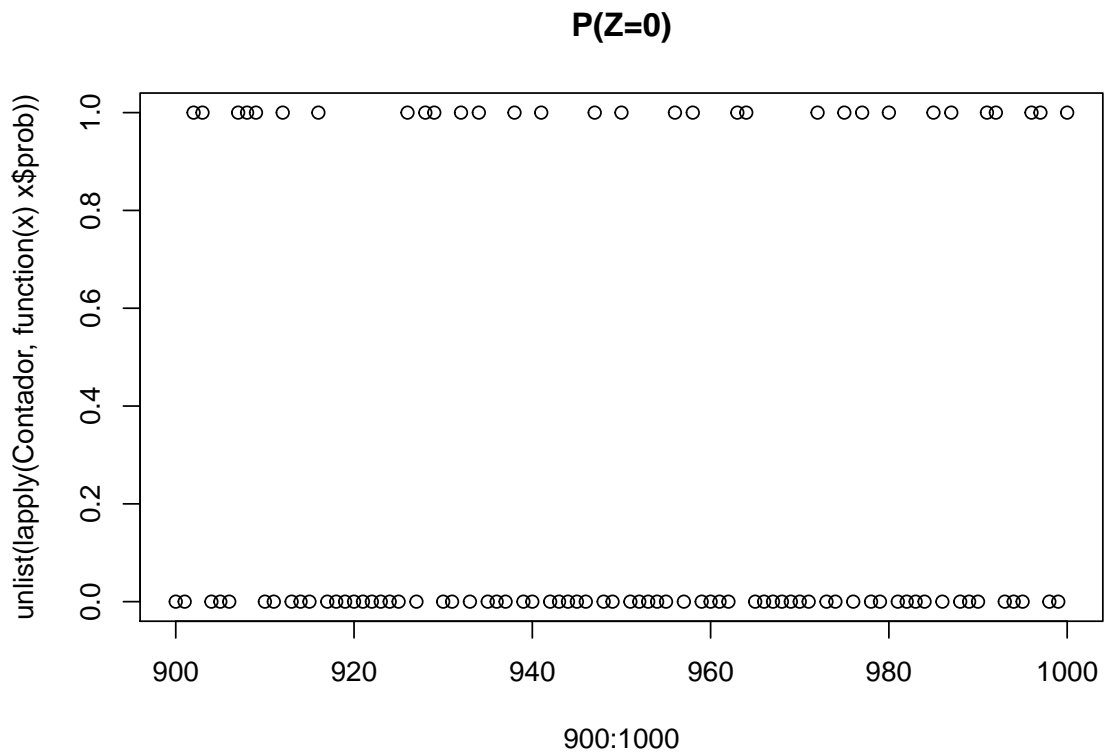
Corrida para $T=10$ y $p=0.2$





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
                                )
            )
  )
}
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) ")
```



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

  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    0      0  698036  564431 1000000 1000000

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

  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    0 1550289 2987848 2887359 4016714 7957074
```

□

3. Proponer algoritmos (método y 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 $\gamma, x \in \mathbb{R}, \beta > 0$.

(b) Gumbel (o de valor extremo)

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

donde $\gamma, x \in \mathbb{R}, \beta > 0$.

(c) Logística

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

donde $\gamma, x \in \mathbb{R}, \beta > 0$.

(d) Pareto

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

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

(e) Vasicek

$$f_{p,\rho}(x) = \sqrt{\frac{1-\rho}{\rho}} \exp \left(\frac{1}{2} \left\{ \Phi^{-1}(x)^2 - \left(\frac{\sqrt{1-\rho}\Phi^{-1}(x) - \Phi^{-1}(p)}{\sqrt{\rho}} \right)^2 \right\} \right)$$

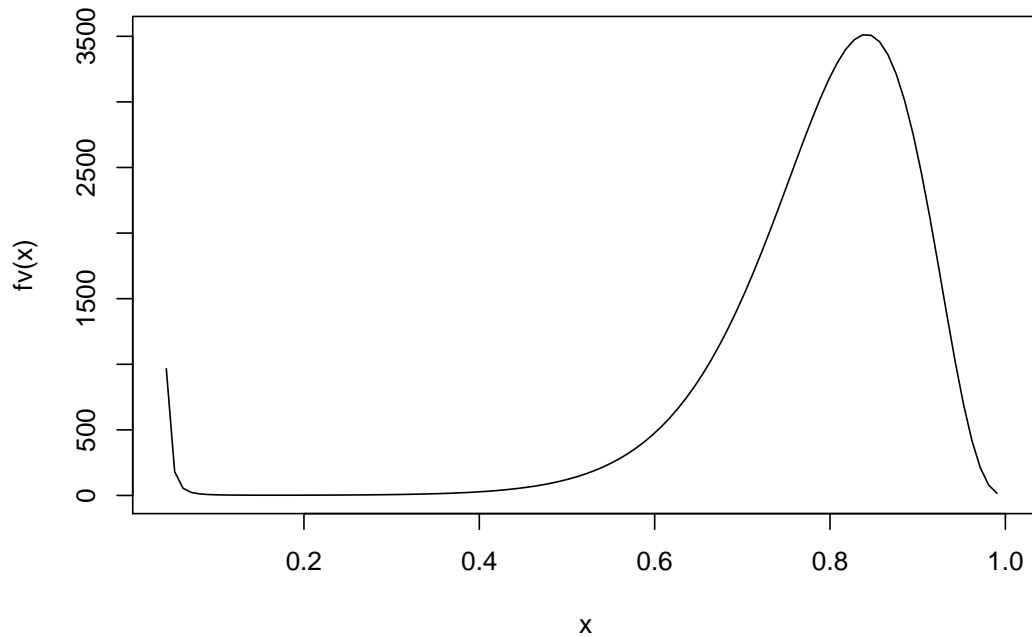
donde $p, \rho \in (0, 1)$ y $\Phi(x)$ es la distribución normal estándar.

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 $n = 50, 100, 150, \dots, 5000$ 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 $X = \frac{c}{(1-u)^{1/\alpha_2}}$.
- (e) La función de Vasicek es muy sensible a los valores de ρ y de p , pero su forma funcional se puede muestrear con aceptación y rechazo en el intervalo (0,1).

```
fv <- function(x, p = 0.1, rho = 0.2)
  sqrt((1-rho)/rho) * exp(0.5 * (qnorm(x)^2 - (sqrt(1-rho) * qnorm(x) - qnorm(p)) / sqrt(rho) )
curve(fv(x), from = 0.043, to = 1)
```



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)

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

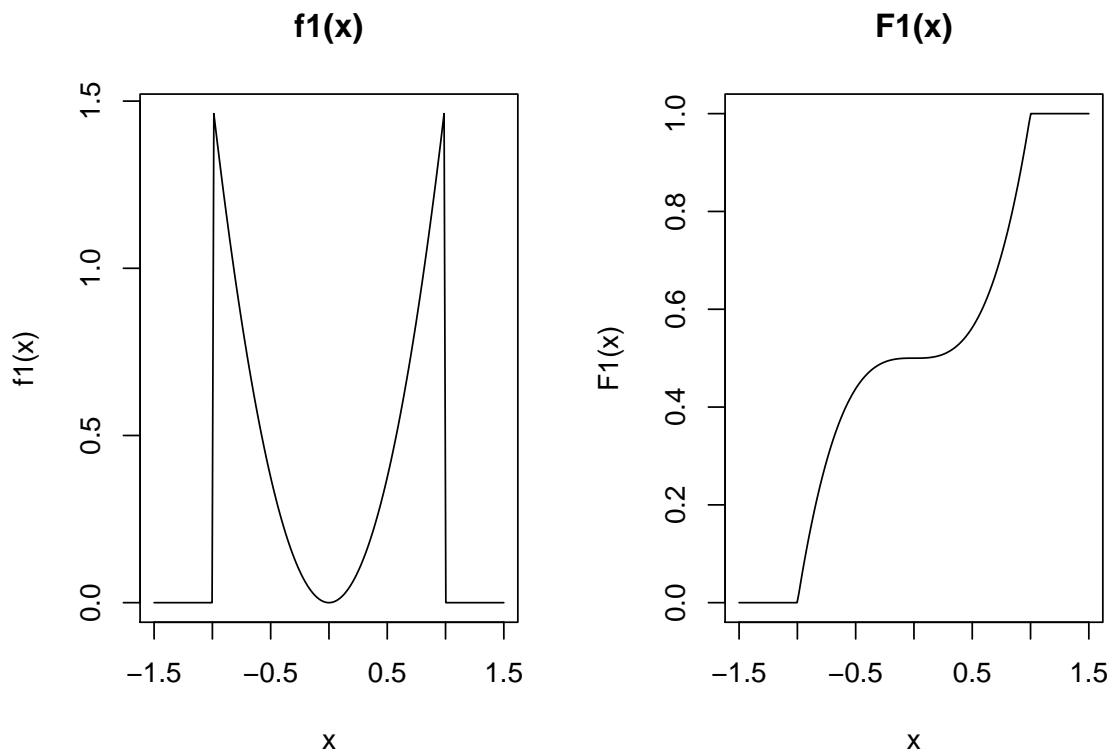
(b) Para $0 < a < \frac{1}{2}$,

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

Solución.

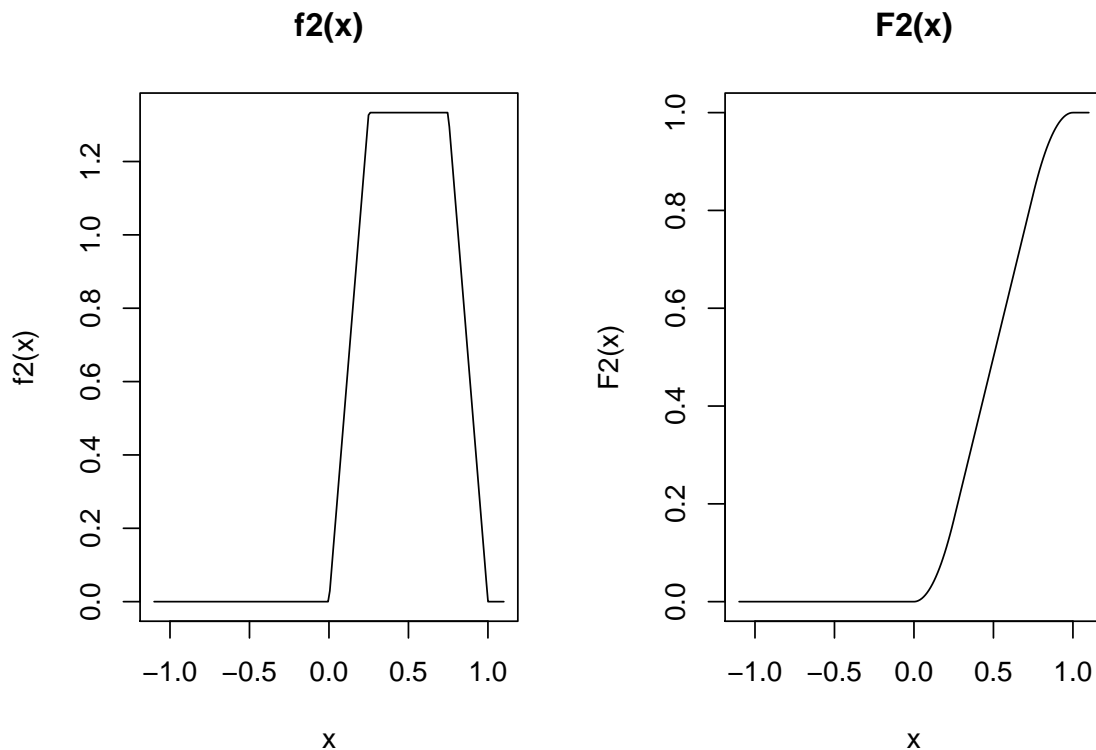
Para el inciso (a), consideremos las funciones de densidad y distribución dadas a continuación:

```
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)

```
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 $X \sim \text{geom}(\pi/4)$. 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:

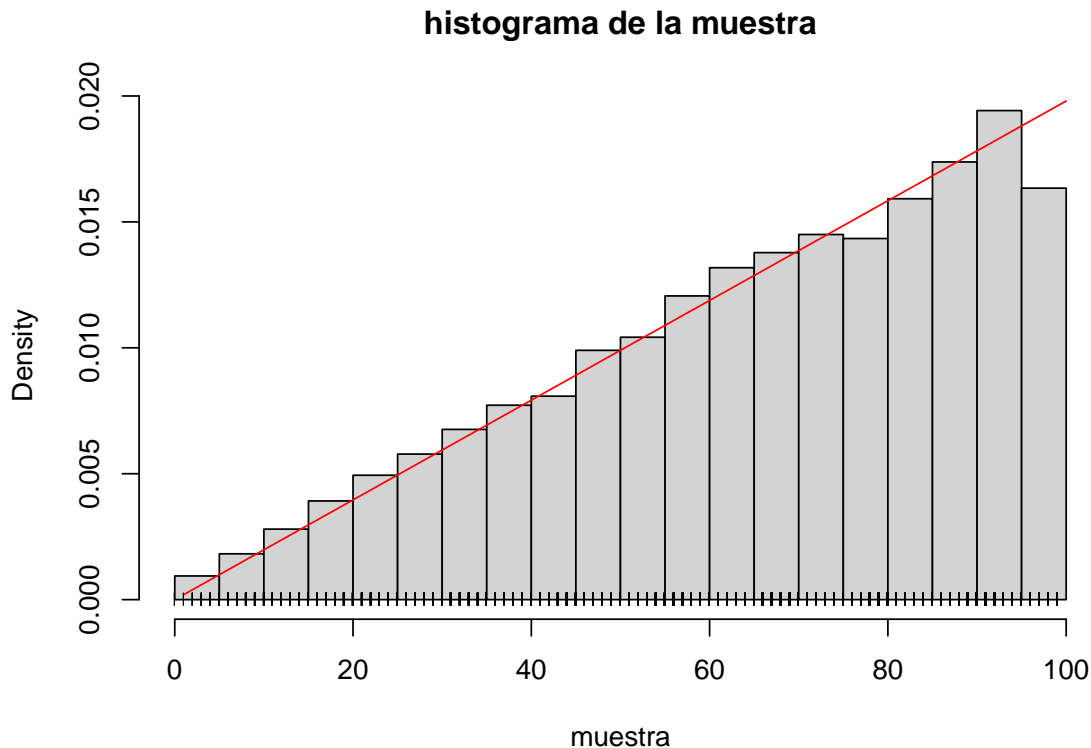
$$p(x) = \frac{2x}{k(k+1)}, x = 1, 2, \dots, k$$

para $k = 100$.

Solución.

Noten que la función de distribución tiene la forma: $F(j) = P(X \leq j) = \frac{2}{k(k+1)} \sum_{i=1}^j i = \frac{2}{k(k+1)} \times \frac{j(j+1)}{2}$ 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
```



□

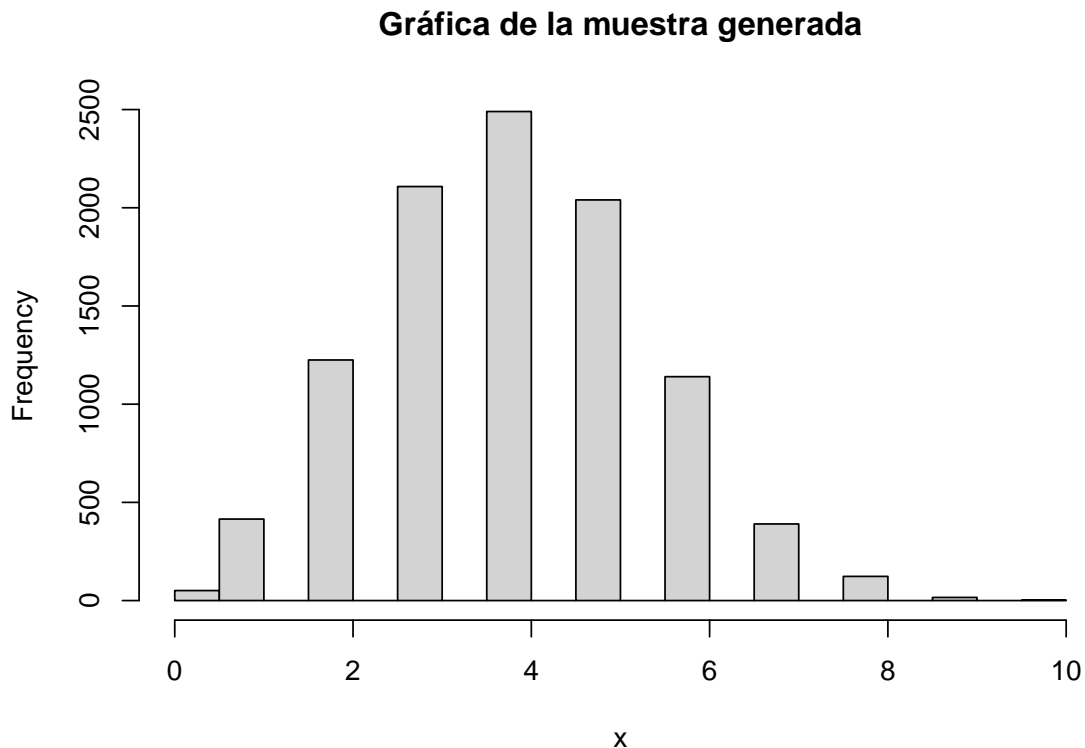
7. ■ Desarrollen un algoritmo para generar una variable aleatoria binomial, usando la técnica de convolución (Hint: ¿cuál es la relación entre binomiales y Bernoullis?) Generar una muestra de 100,000 números. ¿Qué método es más eficiente, el de convoluciones o la función `rbinom` en R?

- Ahora supongan que tienen Y como una suma de $n = 100$ variables Bernoulli con parámetro $p = 0.6$ correlacionadas, con correlación $\rho = 0.3$. Obtener por simulación la estimación de la densidad de Y .

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
0.008   0.000   0.007
```

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

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

Claramente, el método de R es mucho más eficiente.

Para la segunda parte del ejercicio, en clase desarrollamos el ajuste que teníamos que hacer para resolver el problema de la dependencia. Aquí se aplica el método NORTA. Suponemos que $I_i = I(X_i \leq u) \sim \text{Bernoulli}(p)$, donde $X_i \sim \mathcal{N}(0, 1)$. Entonces podemos escoger $u = \Phi^{-1}(p)$, ya que

$$p = P(I_i = 1) = P(X_i \leq u) = \Phi(u)$$

Por otro lado, $\Psi = \text{Cor}(I_i, I_j) = \frac{\text{Cov}(I_i, I_j)}{p(1-p)}$, por lo que

$$\text{Cov}(I_i, I_j) = \Psi p(1-p)$$

Por otro lado, tomando la definición de covarianza en el caso Bernoulli,

$$\text{Cov}(I_i, I_j) = E(I_i I_j) - p^2 = P(I_i = 1, I_j = 1) - p^2 = \Phi_2(u, u; \rho) - p^2$$

y

$$P(I_i = 1, I_j = 1) = p^2 + \Psi p(1-p)$$

Por lo anterior, podemos encontrar el valor de la correlación en la distribución normal bivariada de tal manera que

$$p^2 + \Psi p(1-p) = \Phi_2(u, u; \rho)$$

```
library(pbv) # normal bivariada
p <- 0.6
u <- qnorm(p)
psi <- 0.3
G <- function(x, p = 0.6, u = qnorm(p)) p^2 + psi*p*(1-p) - pbvnorm(u, u, x)
(rho <- uniroot(G, interval=c(-1,1))$root)

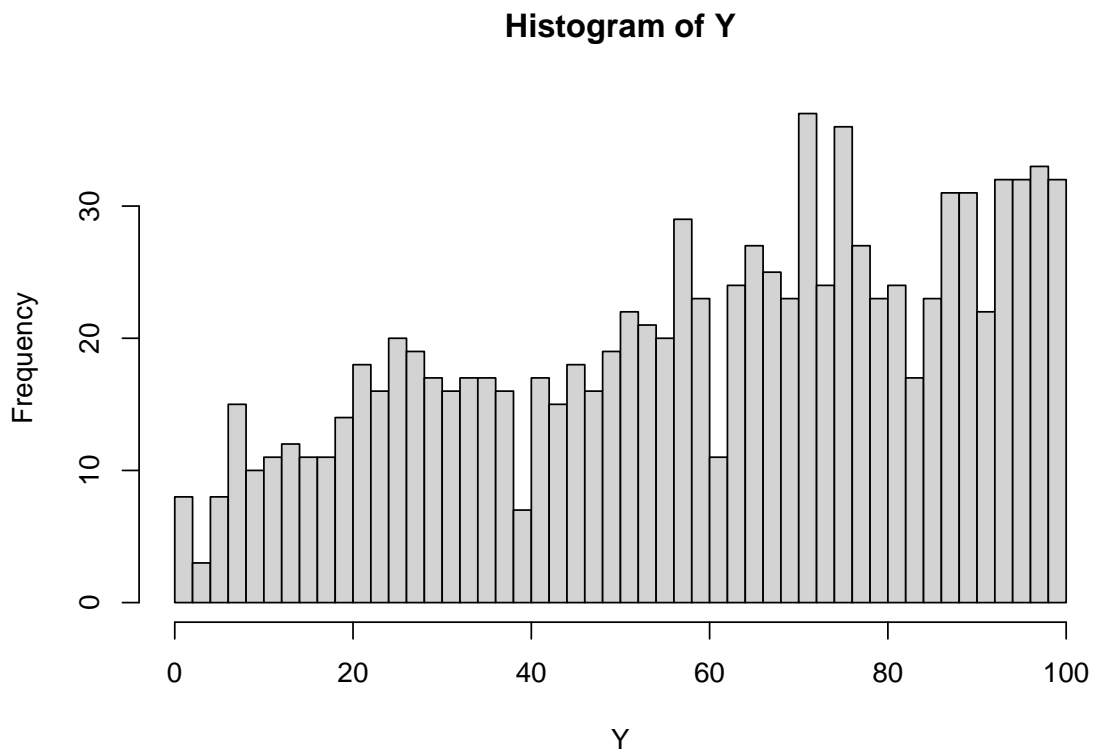
[1] 0.4589874
```

□

hora generamos $n = 100$ normales dependientes con correlación dada por $\rho = 0.4589874$ y creamos las Bernoullis con el umbral $u = 0.2533471$.

```
rmvn <- function(n,mu,Sigma){
  d <- length(mu)
  Q <- chol(Sigma)
  Z <- matrix(rnorm(n*d), nrow = n)
  X <- Z %*% Q + matrix(mu, n, d, byrow = T)
  X
}
# Creamos la matriz de correlaciones
S <- matrix(rho, nrow = 100, ncol = 100)
diag(S) <- 1

# Generamos una muestra de vectores normal con la correlación deseada
X <- rmvn(n = 1000, mu = rep(0,100), Sigma = S)
# Generamos las Bernoullis
I <- apply(X, c(1,2), function(x) ifelse(x < u, 1, 0))
# Una muestra de las sumas de Bernoullis dependientes para calcular su histograma
Y <- apply(I, 1, sum)
hist(Y, breaks = 50)
```



8. 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.

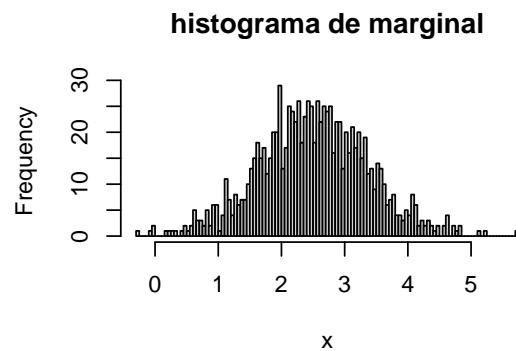
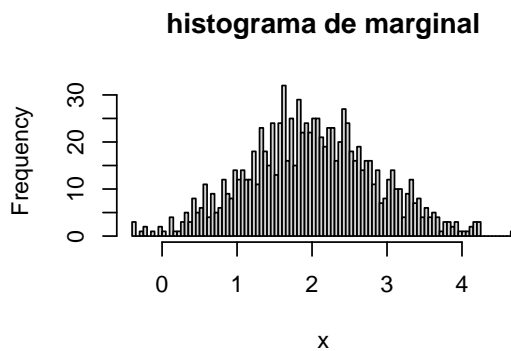
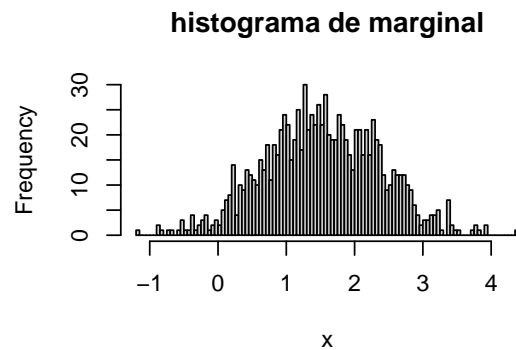
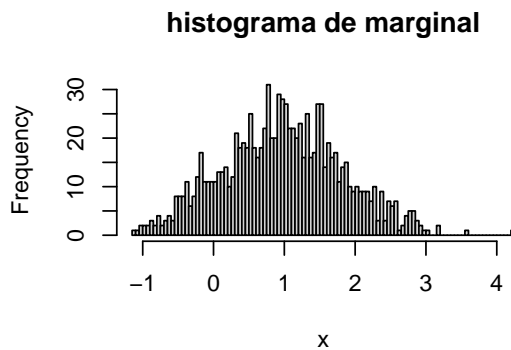
El siguiente script hace lo solicitado. Aquí estoy usando la función `mvrnorm` del paquete MASS, pero también se puede usar el método de Box-Müller para generar los valores normales de los vectores.

```
library(MASS)

mezclaNor <- function(n,mu1,mu2,S1,S2,p){
  #Esta función genera una mezcla de dos normales  $p \cdot N(\mu_1, S_1) + (1-p) \cdot N(\mu_2, S_2)$ 
  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] -1.15 -1.10 -1.05 -1.00 -0.95 -0.90 -0.85 -0.80 -0.75 -0.70 -0.65 -0.60
 [13] -0.55 -0.50 -0.45 -0.40 -0.35 -0.30 -0.25 -0.20 -0.15 -0.10 -0.05  0.00
 [25]  0.05  0.10  0.15  0.20  0.25  0.30  0.35  0.40  0.45  0.50  0.55  0.60
 [37]  0.65  0.70  0.75  0.80  0.85  0.90  0.95  1.00  1.05  1.10  1.15  1.20
 [49]  1.25  1.30  1.35  1.40  1.45  1.50  1.55  1.60  1.65  1.70  1.75  1.80
 [61]  1.85  1.90  1.95  2.00  2.05  2.10  2.15  2.20  2.25  2.30  2.35  2.40
 [73]  2.45  2.50  2.55  2.60  2.65  2.70  2.75  2.80  2.85  2.90  2.95  3.00
 [85]  3.05  3.10  3.15  3.20  3.25  3.30  3.35  3.40  3.45  3.50  3.55  3.60
 [97]  3.65  3.70  3.75  3.80  3.85  3.90  3.95  4.00  4.05  4.10  4.15  4.20
[109]  4.25

$counts
  [1]  1  1  2  2  2  3  2  4  2  3  4  3  8  8  8 11  6  8 12 17 11 11 11 11 13
 [26] 13 14 10 12 21 18 19 18 25 18 16 18 22 31 20 20 29 28 27 22 22 20 23 16 25
 [51] 16 17 27 27 14 19 16 17 11 14 15 10  9 10  9  9  9  7 10  3  9  3  7  6  7
 [76]  1  2  4  5  5  3  2  1  1  0  0  2  0  0  0  0  0  0  0  1  0  0  0  0  0
[101]  0  0  0  0  0  0  0  0  1

$density
  [1] 0.02 0.02 0.04 0.04 0.04 0.06 0.04 0.08 0.04 0.06 0.08 0.06 0.16 0.16 0.16
 [16] 0.22 0.12 0.16 0.24 0.34 0.22 0.22 0.22 0.22 0.26 0.26 0.28 0.20 0.24 0.42
 [31] 0.36 0.38 0.36 0.50 0.36 0.32 0.36 0.44 0.62 0.40 0.40 0.58 0.56 0.54 0.44
 [46] 0.44 0.40 0.46 0.32 0.50 0.32 0.34 0.54 0.54 0.28 0.38 0.32 0.34 0.22 0.28
 [61] 0.30 0.20 0.18 0.20 0.18 0.18 0.18 0.14 0.20 0.06 0.18 0.06 0.14 0.12 0.14
 [76] 0.02 0.04 0.08 0.10 0.10 0.06 0.04 0.02 0.02 0.00 0.00 0.04 0.00 0.00 0.00
 [91] 0.00 0.00 0.00 0.00 0.02 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
[106] 0.00 0.00 0.02

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

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

$sequidist
[1] TRUE

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

[[2]]
$breaks
  [1] -1.20 -1.15 -1.10 -1.05 -1.00 -0.95 -0.90 -0.85 -0.80 -0.75 -0.70 -0.65
 [13] -0.60 -0.55 -0.50 -0.45 -0.40 -0.35 -0.30 -0.25 -0.20 -0.15 -0.10 -0.05
 [25]  0.00  0.05  0.10  0.15  0.20  0.25  0.30  0.35  0.40  0.45  0.50  0.55
 [37]  0.60  0.65  0.70  0.75  0.80  0.85  0.90  0.95  1.00  1.05  1.10  1.15
 [49]  1.20  1.25  1.30  1.35  1.40  1.45  1.50  1.55  1.60  1.65  1.70  1.75
 [61]  1.80  1.85  1.90  1.95  2.00  2.05  2.10  2.15  2.20  2.25  2.30  2.35
 [73]  2.40  2.45  2.50  2.55  2.60  2.65  2.70  2.75  2.80  2.85  2.90  2.95
 [85]  3.00  3.05  3.10  3.15  3.20  3.25  3.30  3.35  3.40  3.45  3.50  3.55
 [97]  3.60  3.65  3.70  3.75  3.80  3.85  3.90  3.95  4.00  4.05  4.10  4.15
[109]  4.20  4.25  4.30  4.35  4.40

```

```

$counts
 [1] 1 0 0 0 0 0 2 1 0 1 1 0 1 3 1 1 4 1 2 3 4 1 2 3 2
[26] 5 7 8 14 4 10 9 13 12 11 10 15 13 18 11 18 16 21 24 22 15 19 25 17 30
[51] 21 24 22 26 22 28 20 19 19 24 22 19 16 13 21 21 16 21 16 23 19 18 12 9 10
[76] 13 12 12 12 10 9 6 4 2 3 3 4 4 5 1 0 7 2 1 1 0 0 0 1 2
[101] 1 0 2 0 0 0 0 0 0 0 0 0 0 1

$density
 [1] 0.02 0.00 0.00 0.00 0.00 0.00 0.04 0.02 0.00 0.02 0.02 0.00 0.02 0.06 0.02
[16] 0.02 0.08 0.02 0.04 0.06 0.08 0.02 0.04 0.06 0.04 0.10 0.14 0.16 0.28 0.08
[31] 0.20 0.18 0.26 0.24 0.22 0.20 0.30 0.26 0.36 0.22 0.36 0.32 0.42 0.48 0.44
[46] 0.30 0.38 0.50 0.34 0.60 0.42 0.48 0.44 0.52 0.44 0.56 0.40 0.38 0.38 0.48
[61] 0.44 0.38 0.32 0.26 0.42 0.42 0.32 0.42 0.32 0.46 0.38 0.36 0.24 0.18 0.20
[76] 0.26 0.24 0.24 0.24 0.20 0.18 0.12 0.08 0.04 0.06 0.06 0.08 0.08 0.10 0.02
[91] 0.00 0.14 0.04 0.02 0.02 0.00 0.00 0.00 0.02 0.04 0.02 0.00 0.04 0.00 0.00
[106] 0.00 0.00 0.00 0.00 0.00 0.00 0.02

$mids
 [1] -1.175 -1.125 -1.075 -1.025 -0.975 -0.925 -0.875 -0.825 -0.775 -0.725
[11] -0.675 -0.625 -0.575 -0.525 -0.475 -0.425 -0.375 -0.325 -0.275 -0.225
[21] -0.175 -0.125 -0.075 -0.025 0.025 0.075 0.125 0.175 0.225 0.275
[31] 0.325 0.375 0.425 0.475 0.525 0.575 0.625 0.675 0.725 0.775
[41] 0.825 0.875 0.925 0.975 1.025 1.075 1.125 1.175 1.225 1.275
[51] 1.325 1.375 1.425 1.475 1.525 1.575 1.625 1.675 1.725 1.775
[61] 1.825 1.875 1.925 1.975 2.025 2.075 2.125 2.175 2.225 2.275
[71] 2.325 2.375 2.425 2.475 2.525 2.575 2.625 2.675 2.725 2.775
[81] 2.825 2.875 2.925 2.975 3.025 3.075 3.125 3.175 3.225 3.275
[91] 3.325 3.375 3.425 3.475 3.525 3.575 3.625 3.675 3.725 3.775
[101] 3.825 3.875 3.925 3.975 4.025 4.075 4.125 4.175 4.225 4.275
[111] 4.325 4.375

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

$sequidist
[1] TRUE

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

[[3]]
$breaks
 [1] -0.40 -0.35 -0.30 -0.25 -0.20 -0.15 -0.10 -0.05 0.00 0.05 0.10 0.15
[13] 0.20 0.25 0.30 0.35 0.40 0.45 0.50 0.55 0.60 0.65 0.70 0.75
[25] 0.80 0.85 0.90 0.95 1.00 1.05 1.10 1.15 1.20 1.25 1.30 1.35
[37] 1.40 1.45 1.50 1.55 1.60 1.65 1.70 1.75 1.80 1.85 1.90 1.95
[49] 2.00 2.05 2.10 2.15 2.20 2.25 2.30 2.35 2.40 2.45 2.50 2.55
[61] 2.60 2.65 2.70 2.75 2.80 2.85 2.90 2.95 3.00 3.05 3.10 3.15
[73] 3.20 3.25 3.30 3.35 3.40 3.45 3.50 3.55 3.60 3.65 3.70 3.75
[85] 3.80 3.85 3.90 3.95 4.00 4.05 4.10 4.15 4.20 4.25 4.30 4.35
[97] 4.40 4.45 4.50 4.55 4.60 4.65 4.70

$counts
 [1] 3 0 1 2 0 1 0 2 1 0 4 1 1 3 5 3 8 5 6 11 4 9 5 6 12
[26] 9 8 14 12 14 12 12 18 11 23 18 15 24 13 24 32 16 25 15 29 22 24 22 25 25
[51] 21 19 23 23 16 20 27 24 18 16 19 14 16 16 11 14 7 8 12 14 10 10 4 9 12
[76] 7 8 4 6 4 5 4 1 3 3 2 3 1 1 1 2 3 3 0 0 0 0 0 0 0
[101] 0 1

$density
 [1] 0.06 0.00 0.02 0.04 0.00 0.02 0.00 0.04 0.02 0.00 0.08 0.02 0.02 0.06 0.10
[16] 0.06 0.16 0.10 0.12 0.22 0.08 0.18 0.10 0.12 0.24 0.18 0.16 0.28 0.24 0.28
[31] 0.24 0.24 0.36 0.22 0.46 0.36 0.30 0.48 0.26 0.48 0.64 0.32 0.50 0.30 0.58
[46] 0.44 0.48 0.44 0.50 0.50 0.42 0.38 0.46 0.46 0.32 0.40 0.54 0.48 0.36 0.32

```

```

[61] 0.38 0.28 0.32 0.32 0.22 0.28 0.14 0.16 0.24 0.28 0.20 0.20 0.08 0.18 0.24
[76] 0.14 0.16 0.08 0.12 0.08 0.10 0.08 0.02 0.06 0.06 0.04 0.06 0.02 0.02 0.02
[91] 0.04 0.06 0.06 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.02

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

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

$sequidist
[1] TRUE

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

[[4]]
$breaks
 [1] -0.30 -0.25 -0.20 -0.15 -0.10 -0.05  0.00  0.05  0.10  0.15  0.20  0.25
[13]  0.30  0.35  0.40  0.45  0.50  0.55  0.60  0.65  0.70  0.75  0.80  0.85
[25]  0.90  0.95  1.00  1.05  1.10  1.15  1.20  1.25  1.30  1.35  1.40  1.45
[37]  1.50  1.55  1.60  1.65  1.70  1.75  1.80  1.85  1.90  1.95  2.00  2.05
[49]  2.10  2.15  2.20  2.25  2.30  2.35  2.40  2.45  2.50  2.55  2.60  2.65
[61]  2.70  2.75  2.80  2.85  2.90  2.95  3.00  3.05  3.10  3.15  3.20  3.25
[73]  3.30  3.35  3.40  3.45  3.50  3.55  3.60  3.65  3.70  3.75  3.80  3.85
[85]  3.90  3.95  4.00  4.05  4.10  4.15  4.20  4.25  4.30  4.35  4.40  4.45
[97]  4.50  4.55  4.60  4.65  4.70  4.75  4.80  4.85  4.90  4.95  5.00  5.05
[109] 5.10  5.15  5.20  5.25  5.30  5.35  5.40  5.45  5.50  5.55  5.60  5.65
[121] 5.70  5.75

$counts
 [1]  1  0  0  0  1  2  0  0  0  1  1  1  1  0  1  2  1  2  5  3  3  2  5  2  6
[26]  6  1  4 11  7  4  8  6  7  7 10 13 15 18 15 17 12 15 20 20 29 13 17 25 24
[51] 22 26 18 23 26 25 18 26 22 25 24 25 16 22 22 13 20 16 21 17 20 15 19 12 13
[76]  9 14 13 10  6  7  8  4  4  3  5  4  8  6  2  3  2  3  2  1  2  0  2  4  0
[101]  2  1  2  0  0  0  0  0  1  0  1  0  0  0  0  0  0  0  0  0  0  0  1

$density
 [1] 0.02 0.00 0.00 0.00 0.00 0.02 0.04 0.00 0.00 0.00 0.02 0.02 0.02 0.02 0.00 0.02
[16] 0.04 0.02 0.04 0.10 0.06 0.06 0.04 0.10 0.04 0.12 0.12 0.02 0.08 0.22 0.14
[31] 0.08 0.16 0.12 0.14 0.14 0.20 0.26 0.30 0.36 0.30 0.34 0.24 0.30 0.40 0.40
[46] 0.58 0.26 0.34 0.50 0.48 0.44 0.52 0.36 0.46 0.52 0.50 0.36 0.52 0.44 0.50
[61] 0.48 0.50 0.32 0.44 0.44 0.26 0.40 0.32 0.42 0.34 0.40 0.30 0.38 0.24 0.26
[76] 0.18 0.28 0.26 0.20 0.12 0.14 0.16 0.08 0.08 0.06 0.10 0.08 0.16 0.12 0.04
[91] 0.06 0.04 0.06 0.04 0.02 0.04 0.00 0.04 0.08 0.00 0.04 0.02 0.04 0.00 0.00
[106] 0.00 0.00 0.00 0.02 0.00 0.02 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
[121] 0.02

$mids
 [1] -0.275 -0.225 -0.175 -0.125 -0.075 -0.025  0.025  0.075  0.125  0.175
[11]  0.225  0.275  0.325  0.375  0.425  0.475  0.525  0.575  0.625  0.675
[21]  0.725  0.775  0.825  0.875  0.925  0.975  1.025  1.075  1.125  1.175
[31]  1.225  1.275  1.325  1.375  1.425  1.475  1.525  1.575  1.625  1.675
[41]  1.725  1.775  1.825  1.875  1.925  1.975  2.025  2.075  2.125  2.175

```



```

[51] 2.225 2.275 2.325 2.375 2.425 2.475 2.525 2.575 2.625 2.675
[61] 2.725 2.775 2.825 2.875 2.925 2.975 3.025 3.075 3.125 3.175
[71] 3.225 3.275 3.325 3.375 3.425 3.475 3.525 3.575 3.625 3.675
[81] 3.725 3.775 3.825 3.875 3.925 3.975 4.025 4.075 4.125 4.175
[91] 4.225 4.275 4.325 4.375 4.425 4.475 4.525 4.575 4.625 4.675
[101] 4.725 4.775 4.825 4.875 4.925 4.975 5.025 5.075 5.125 5.175
[111] 5.225 5.275 5.325 5.375 5.425 5.475 5.525 5.575 5.625 5.675
[121] 5.725

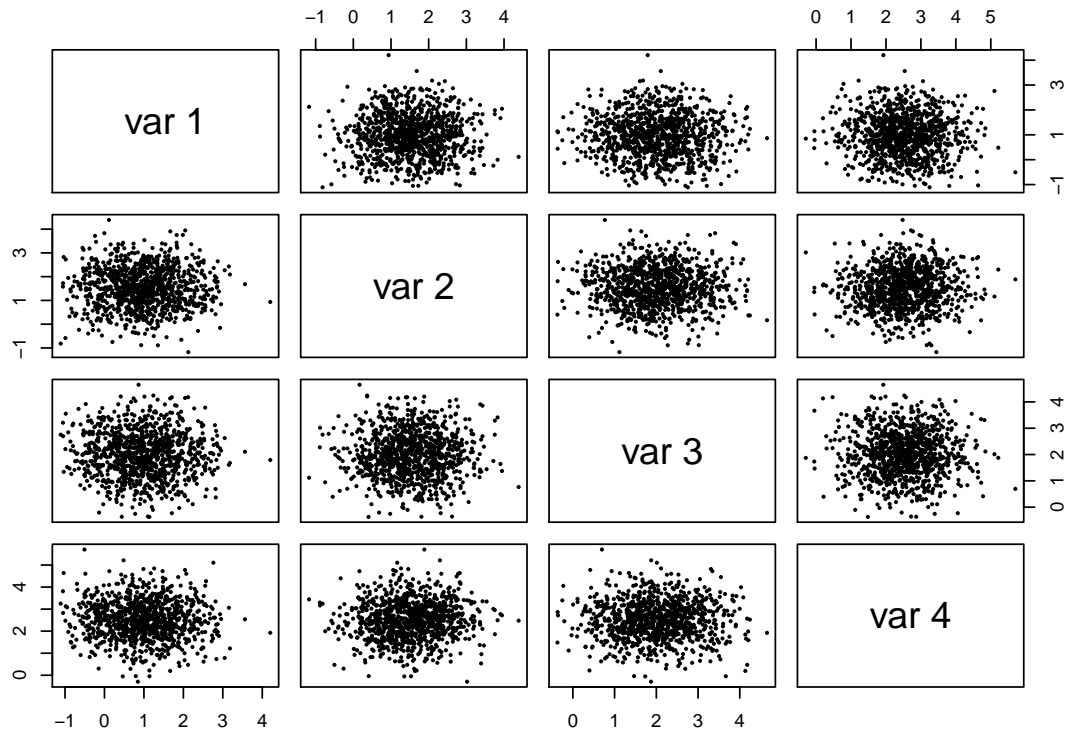
$хname
[1] "newX[, i]"

$sequidist
[1] TRUE

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

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

```



□

9. **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
 - $T_{ij} \sim \mathcal{N}(0, 1)$ independientes, para $i > j$.
 - $T_{ii} \sim \sqrt{\chi^2_{(n-i+1)}}$, $i = 1, \dots, d$.
 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 distribución 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,] 15.095261 -1.8939129  6.9672392 -4.0914122
[2,] -1.893913  9.8630367  0.4929256  0.9295381
[3,]  6.967239  0.4929256  9.9260737 -1.9394035
[4,] -4.091412  0.9295381 -1.9394035  4.3828629

[[2]]
      [,1]      [,2]      [,3]      [,4]
[1,] 12.131446293 -1.8131569 -0.002590887  2.972412
[2,] -1.813156930  5.9333469 -0.487477099 -0.892165
[3,] -0.002590887 -0.4874771  9.472478629 -5.612645
[4,]  2.972411525 -0.8921650 -5.612645470 11.839582
```

```

[[3]]
      [,1]      [,2]      [,3]      [,4]
[1,] 8.1111539 0.8194673 4.584820 2.234849
[2,] 0.8194673 2.6250779 1.518938 1.023942
[3,] 4.5848204 1.5189384 11.120520 2.354045
[4,] 2.2348491 1.0239425 2.354045 8.376634

[[4]]
      [,1]      [,2]      [,3]      [,4]
[1,] 4.753926 1.5580564 2.5384296 1.7329682
[2,] 1.558056 2.9143963 0.4461518 -0.5638284
[3,] 2.538430 0.4461518 5.8404546 -2.5043780
[4,] 1.732968 -0.5638284 -2.5043780 6.7305530

[[5]]
      [,1]      [,2]      [,3]      [,4]
[1,] 8.185551 -5.0643213 -5.5753738 -2.088836
[2,] -5.064321 10.1109102 0.7998443 1.156694
[3,] -5.575374 0.7998443 11.5945235 2.202376
[4,] -2.088836 1.1566945 2.2023757 4.824407

```

Para la segunda parte, usamos la transformación sugerida:

```

W2 <- function(k,n,mu1,S1){
  #Esta función genera k muestras de la distribución Wishart utilizando la
  #descomposición de Bartlett.
  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,] 5.5613919 -1.134494 -0.6837185 4.140644
[2,] -1.1344945 7.585988 -2.4920997 -1.546949
[3,] -0.6837185 -2.492100 5.0586525 -2.134487
[4,] 4.1406436 -1.546949 -2.1344871 4.979573

[[2]]
      [,1]      [,2]      [,3]      [,4]
[1,] 4.15953646 -0.05835086 0.4667704 -0.4676370
[2,] -0.05835086 3.88096990 -0.6588826 0.1902958
[3,] 0.46677037 -0.65888259 5.0478448 -0.7847228
[4,] -0.46763701 0.19029580 -0.7847228 3.7506283

[[3]]
      [,1]      [,2]      [,3]      [,4]
[1,] 7.64734798 -0.01737887 -2.594284 0.2477977
[2,] -0.01737887 14.79076360 -4.640821 -2.8886973
[3,] -2.59428372 -4.64082147 8.829083 -1.9997361
[4,] 0.24779772 -2.88869735 -1.999736 5.5126079

[[4]]

```

```

      [,1]      [,2]      [,3]      [,4]
[1,]  4.59409190 -0.07302467 -1.391884  1.413785
[2,] -0.07302467  3.27436112 -2.738061  1.940043
[3,] -1.39188421 -2.73806130  6.668267 -5.636206
[4,]  1.41378541  1.94004349 -5.636206 10.653166

[[5]]
      [,1]      [,2]      [,3]      [,4]
[1,]  1.63640852  0.497325  0.03224523  1.6513851
[2,]  0.49732499 16.364041 -2.36677840 -1.4713491
[3,]  0.03224523 -2.366778  3.86220732  0.5002321
[4,]  1.65138514 -1.471349  0.50023212  5.0198294

```

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
      0.532   0.000   0.533

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

      user  system elapsed
      0.240   0.000   0.239

```

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`.

□

10.
 - Construyan un vector de 100 números crecientes y espaciados regularmente entre 0.1 y 20. Lláménlo `SIG2`. Ahora construyan otro vector de longitud 21 empezando en -1 y terminando en 1. Lláménlo `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 , lláménlo `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.

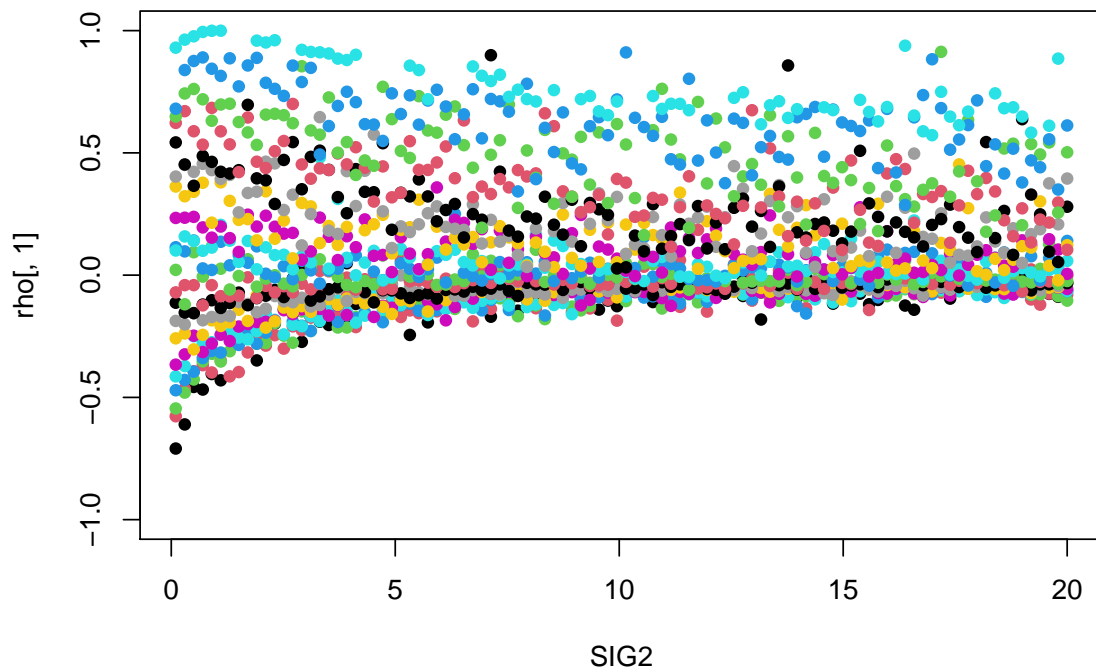
```
normalBM <- function(n) {
  #genera una muestra de pares de normales independientes de tamaño n.
  u1 <- runif(n)
  u2 <- runif(n)
  R <- sqrt(-2*log(u1))
  z1 <- R*cos(2*pi*u2)
  z2 <- R*sin(2*pi*u2)
  return(cbind(z1,z2))
}
```

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, es que la transformación no es lineal para los valores de la correlación. y que hay combinaciones de la varianza y la correlación que no se pueden dar en la distribución lognormal.

□