

# Tarea 2

## Estadística Actuarial II

José Ignacio Rojas Zárate, C16911      Montserrat Beirute Abarca, C10997  
Valeria Vásquez Venegas, C18373

08 de febrero de 2024

### Índice

1. Ejercicio 1	2
2. Ejercicio 2	5
3. Ejercicio 5	6
4. Ejercicio 6	9

## 1. Ejercicio 1

Usando un algoritmo de integración por Montecarlo estime  $\ln(2)$  con un error absoluto de  $10^{-3}$

Considere la siguiente integral:

$$\int_0^1 \frac{\ln(x+1)}{2} + \frac{1}{2} dx$$

.

*Solución:*

Empezamos separando la integral en dos partes. Para  $\int \frac{\ln(x+1)}{2} dx$ : Realizamos la sustitución  $u = x + 1$  y  $du = dx$ . Note que si  $x \rightarrow 0 \Rightarrow u \rightarrow 1$  y si  $x \rightarrow 1 \Rightarrow u \rightarrow 2$ . Esta primera integral se convierte en:  $\frac{1}{2} \int_1^2 \ln(u) du$

Ahora, se aplica la regla de integración por partes, donde  $m = \ln(u) \Rightarrow dm = \frac{1}{u} du$  y  $dv = 1 \Rightarrow v = u$ :

$$\frac{1}{2} \left( u \ln(u) \Big|_1^2 - \int_1^2 u \frac{1}{u} du \right)$$

Esto se simplifica a:  $\frac{1}{2} (\ln(2) - 1) = \frac{\ln(2)}{2} - \frac{1}{2}$ .

La otra parte de la integral original,  $\int_0^1 \frac{1}{2} dx = \frac{1}{2}$ :

Por lo tanto, la solución a la integral es  $\frac{\ln(2)}{2} - \frac{1}{2} + \frac{1}{2} = \ln(2)$ .

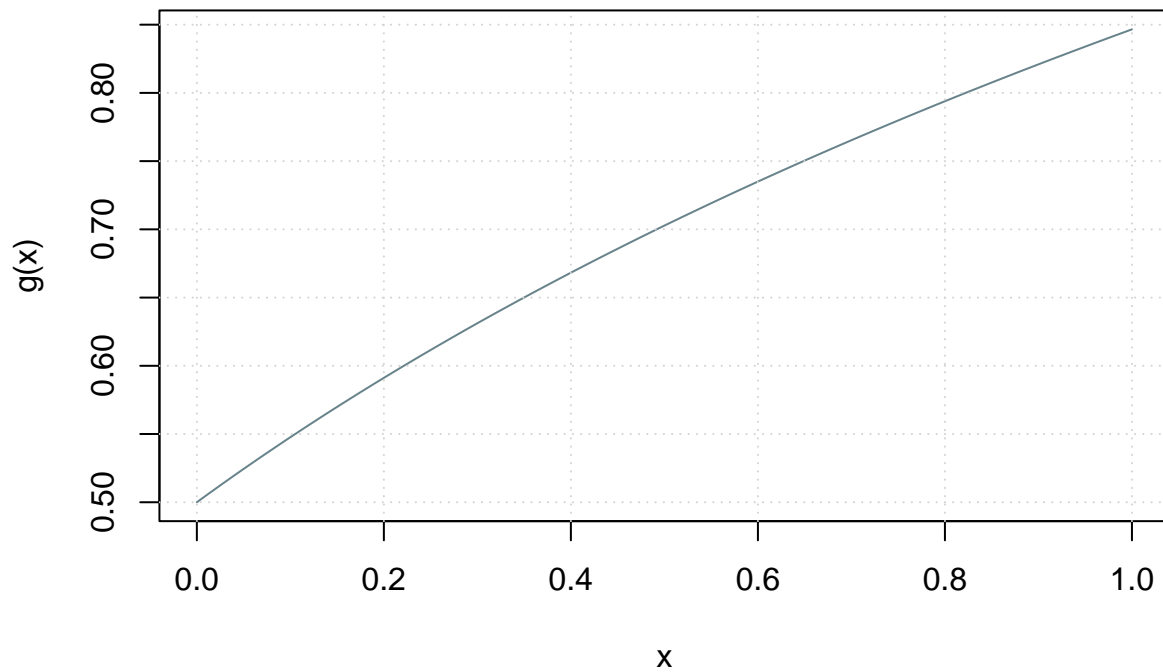
Ahora, procedemos a aplicar el algoritmo de Montecarlo:

```
set.seed(147) # definimos una semilla
n <- 10^4 # tamaño de la muestra
U <- runif(n) # genera un vector con distribución uniforme

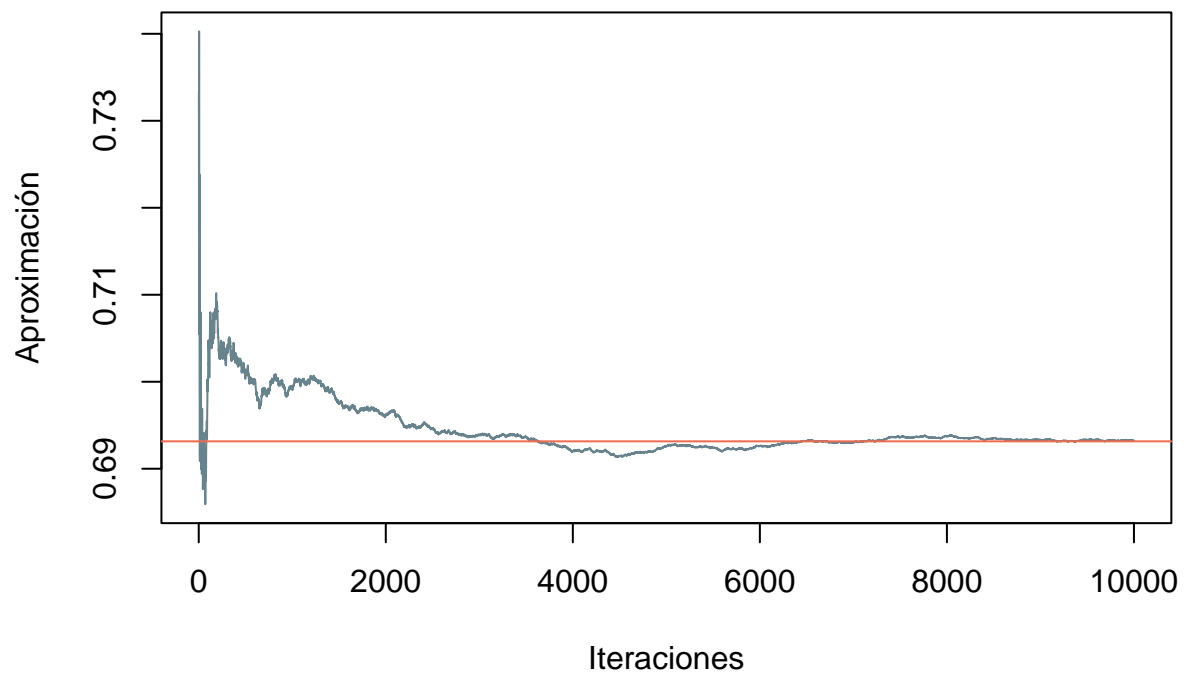
g <- Vectorize(function(x) (log(x + 1)/2) + (1/2) ) # construimos la función g

curve(g,0,1,col="lightblue",lwd=1,main="Gráfico de g(X)")
grid()
```

## Gráfico de $g(X)$



```
Y <- g(U) #genera el vector para cada observación
acumulado<-cumsum(Y)/(1:n)
plot(1:n,acumulado,col="lightblue4",type="l",ylab="Aproximación",xlab="Iteraciones")
abline(h=integrate(g,0,1)$value,col="coral2",lwd=1)
```



```
P_est1=mean(Y)
```

```
e1 <- (abs(P_est1-log(2)))
```

```
e1
```

```
## [1] 4.587881e-05
```

Concluya que con la función considerada, el algoritmo de integración por Montecarlo arroja un error absoluto de  $4.587881e-05$  al estimar  $\ln(2)$ .

## 2. Ejercicio 2

Usando la metodología de Muestreo por Importancia, si  $X \sim N(0,5,0,5)$  estime:

a.  $P(X \leq -5)$

```
pnorm(6,mean = 0.5,sd = (0.5^(1/2)),lower.tail = F)
```

```
## [1] 3.678924e-15
```

Concluya que la  $P(X \leq -5) = 3,678924e - 15$ .

b. Estime el error absoluto de la estimación del punto a:

```
n <-10^4 # Tamaño de la muestra
```

```
A <-rexp(n) + 6
```

```
g <-Vectorize(function(x) ((1/pi) * exp(-(x-0.5)^2)/(exp(-x+6))))
```

```
Y2<-g(A)
```

```
M<-matrix(c(mean(Y2),pnorm(6,mean = 0.5,sd = (0.5^(1/2)), lower.tail = F),  
            abs(mean(Y2)-pnorm(6,mean = 0.5,sd = (0.5^(1/2)), lower.tail = F)),  
            sqrt(var(Y2)/sqrt(10^4))),ncol=4, byrow=TRUE)
```

```
colnames(M) <- c("Estimacion", "Valor Real","Error Absoluto","Error Estandar")
```

```
M
```

```
##          Estimacion  Valor Real Error Absoluto Error Estandar
```

```
## [1,] 2.056013e-15 3.678924e-15  1.622911e-15  4.619617e-16
```

Concluya que el error absoluto es de  $1.622911e-15$  al utilizar la metodología de Muestreo por Importancia.

### 3. Ejercicio 5

Una aseguradora tiene un producto llamado Doble Seguro de Vida (DSV) el cual paga 2 veces la suma asegurada si la persona fallece antes de los 60 años, paga 1 suma asegurada cuando la persona cumple los 60 años (si no ha fallecido) y paga 1 suma asegurada si fallece después de los 60 años.

Construya con la ayuda de un MCMC la distribución de los pagos por año de que se espera de este seguro. Use al menos 10 000 iteraciones. Y muestre Histograma.

Primero abrimos la base de datos de mortalidad, luego filtramos para obtener los datos de un hombre nacido en 1994.

```
tabla_vida_original <- read_excel("tabla.xls")

#filtrando por año de nacimiento y edad
tabla_vida <- tabla_vida_original[(tabla_vida_original$ynac == 1994) &
                                   (tabla_vida_original$sex == 1),]
```

Para realizar la simulación primero definimos algunos aspectos como semilla, numero de iteraciones, vector de mortalidad (filtrado para mortalidades de 30 años en adelante), longitud del vector y un vector vacío donde almacenar los resultados.

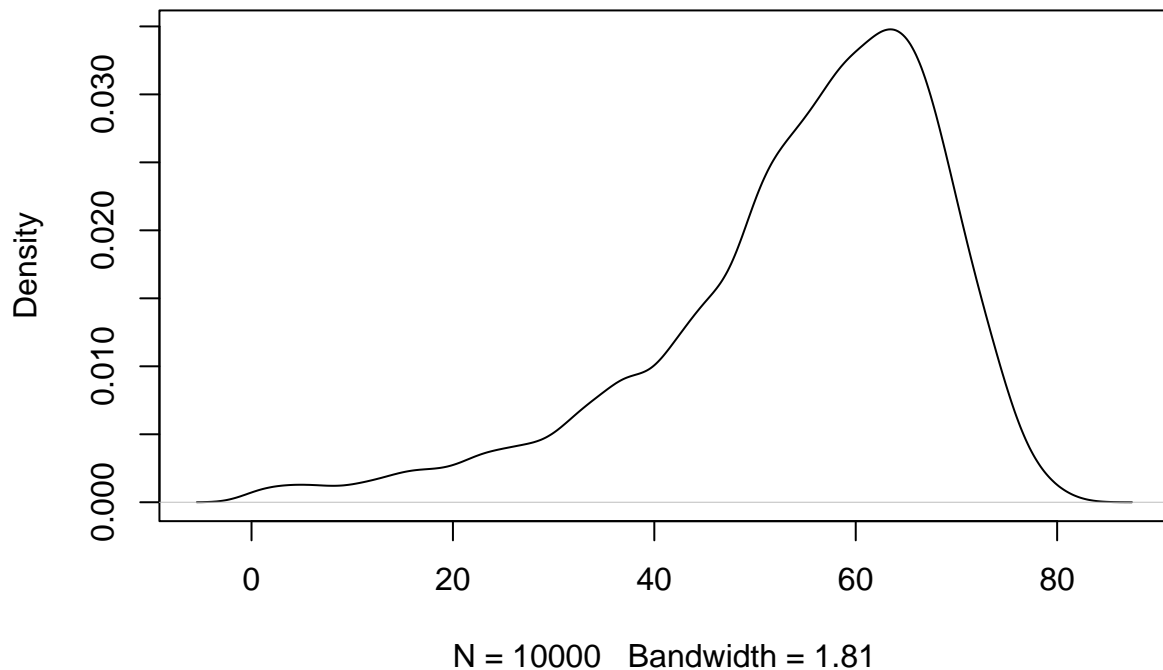
```
set.seed(2024)
iteraciones <- 10^4
vec_qx <- tabla_vida$qx[25:length(tabla_vida$qx)]
n <- length(vec_qx)
vec_resultados <- rep(0, iteraciones)

for(i in c(1:iteraciones)){
  U <- runif(n)
  t <- 1
  cont <- 1
  while(t == 1){
    if(U[cont] > vec_qx[cont]) #Aquí se decide si la persona sobrevive o muere
      {cont <- cont + 1}
    else{
      t <- 0
    }
  }
  vec_resultados[i] <- cont - 1
}
```

Ahora podemos observar la distribución de muertes:

```
plot(density(vec_resultados))
```

**density.default(x = vec\_resultados)**



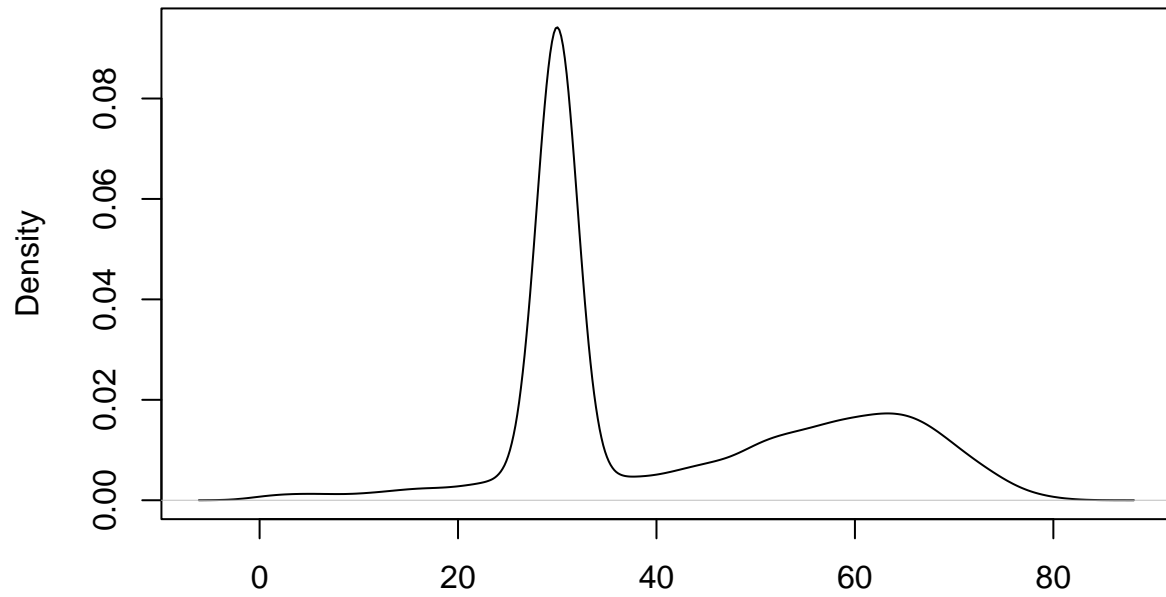
Para poder obtener la distribución de pagos por año, lo que hacemos es considerar que cada muerte después de 30 años (60 años del individuo) representa un pago, pero si la persona muere antes de los próximos 30 años se consideran 2 pagos, por lo que añadimos una observación extra en ese caso. Luego, consideramos que si la persona sobrevive a los 60 años, se le otorga un pago en esa fecha.

```
for (i in 1:length(vec_resultados)){  
  if(vec_resultados[i] < (60-30)){  
    vec_resultados <- c(vec_resultados, vec_resultados[i])  
  }  
  if(vec_resultados[i] >= 30){  
    vec_resultados <- c(vec_resultados, 30)  
  }  
}
```

Graficando el resultado:

```
plot(density(vec_resultados))
```

**density.default(x = vec\_resultados)**



N = 20000 Bandwidth = 2.041



## 4. Ejercicio 6

Usando el Algoritmo de Metropolis-Hastings construya una muestra de  $Z = X_1 - X_2$  donde  $X_1 \sim N(\mu, \sigma^2)$  y  $X_2 \sim N(\mu/2, \sigma^2/4)$ , considere para este ejercicio  $\mu = \sigma = 4$ .

- a. Gráfique la distribución de  $Z$  junto con las medias de  $X_1, X_2$ .

Primero definiremos los parámetros  $\mu = \sigma = 4$ , además, como  $Z = X_1 - X_2$ , entonces  $Z$  sigue una distribución  $Z \sim N(\mu - \mu/2, \sigma^2 + \sigma^2/4)$ . Luego definimos la función fPI.

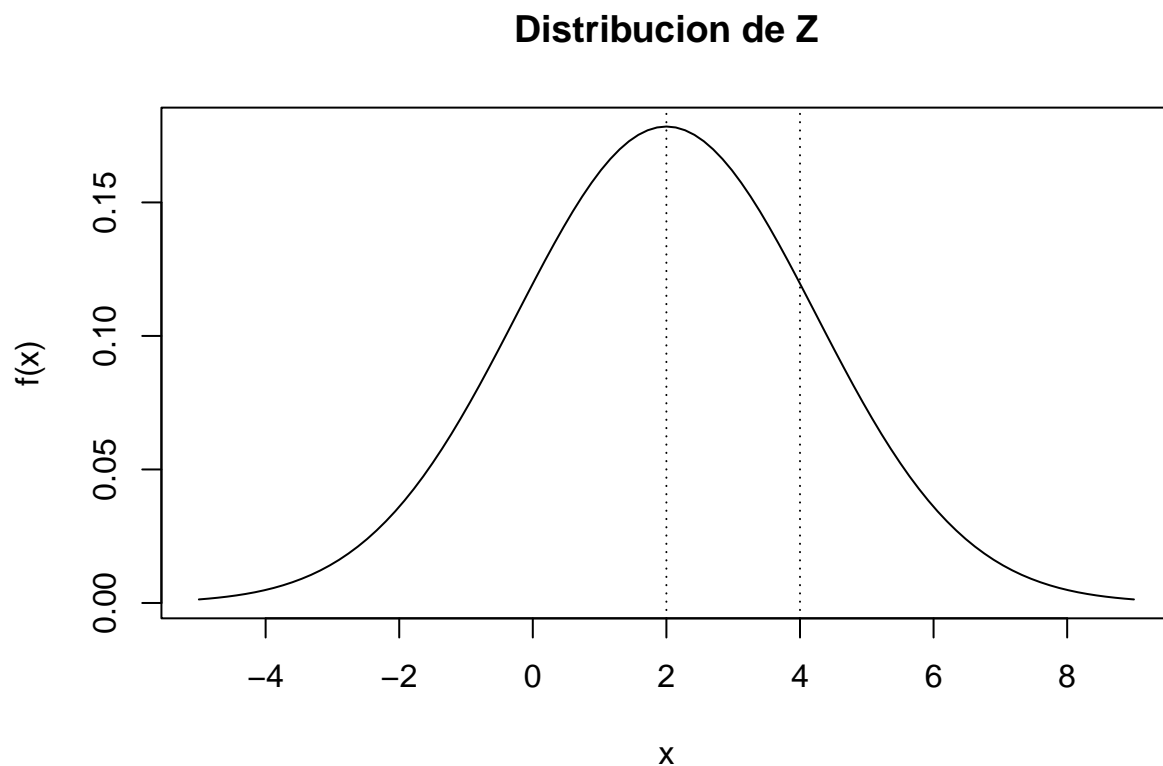
```
mu <- 4
sigma <- 4

mu_z <- 4-(4/2)
sigma_z <- 4+(4/4) #En realidad es sigma^2

fPI = function(x)
{
  fx = exp( -((x - mu_z)^2/(2*(sigma_z))))/(sqrt(2*pi*sigma_z))
  return(fx)
}
```

Ahora podemos mostrar la distribución de  $Z$ :

```
curve(fPI(x), from=-5,to=9, main="Distribucion de Z",xlab="x",ylab="f(x)")
abline(v = 4, lty = 3) #Media de x1
abline(v = 2, lty = 3) #Media de x2
```



- b. Gráfique la distribución (histograma) de la muestra MCMC del algoritmo junto con las medias de

$X_1, X_2$ .

Primero definimos la funcion fpK y luego aplicamos el algoritmo

```
#Usaremos dist cauchy para el proceso de markov
fpK = function(x,y){
  pK = dcauchy(y,location = x) #x es el centro del pico de la distribución.
  return(pK)
}

#Aplicando el algoritmo
N = 10^5 # Número de Iteraciones
L = 1000 # periodo quemado (burn in)

MCMC = matrix(data = 0, nrow = N, ncol = 12)
colnames(MCMC) =
  c("x","y","PIx","PIy","Kxy","Kyx","Rxy","Ryx","Mxy","Myx","Fxy","Salto")
# 1. Inicial con un valor arbitrario de x del dominio de distribución
x = runif(1,-50,50)

for (i in 1:N){
  # 2. Generamos la propuesta con una distribucion arbitraria
  y = rcauchy(1,location = x) #Valor aleatorio según X

  #3. Tasa de Aceptación
  PIx = fPI(x)
  PIy = fPI(y)
  Kxy = fpK(x,y)
  Kyx = fpK(y,x)
  Rxy = (PIy*Kyx) / (PIx*Kxy)
  Ryx = (PIx*Kxy) / (PIy*Kyx)
  # Matriz estocástica de los estados de la distribución estacionaria
  if (x!=y){
    Mxy = Kxy*min(1,Rxy)
    Myx = Kyx*min(1,Ryx)
  }
  else
  { Mxy = -1
    Myx = -1
  }

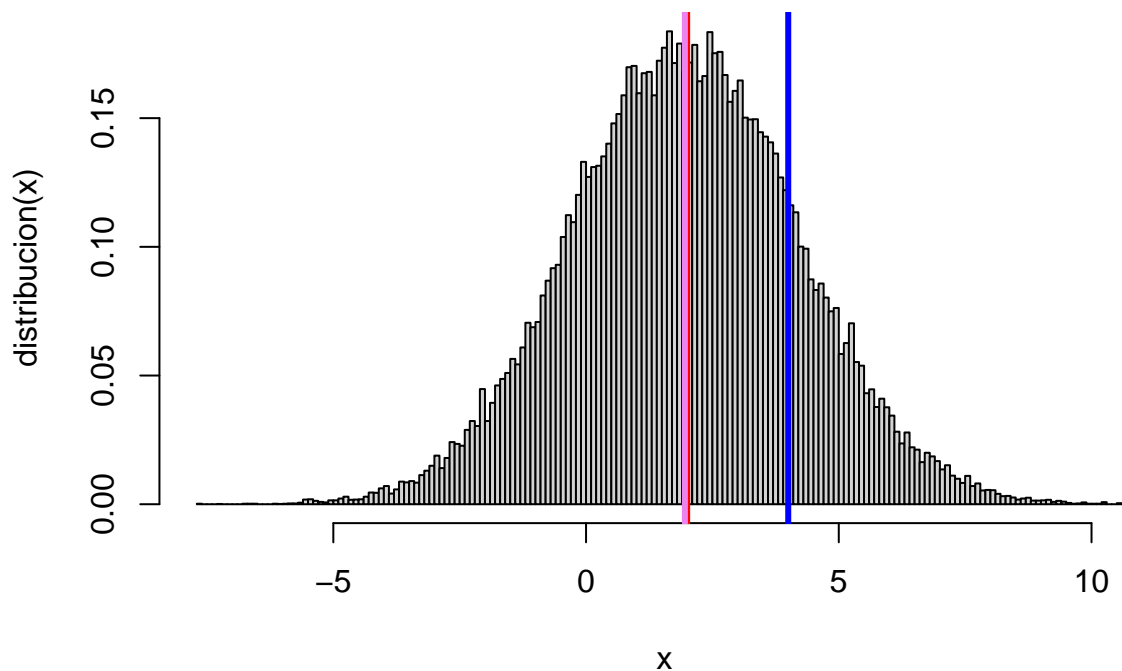
  #4. Criterio de Aceptacion o Rechazo
  #Probabilidad de aceptación,runif(1)
  Fxy = runif(1)
  MCMC[i,] = c(x,y,PIx,PIy,Kxy,Kyx,Rxy,Ryx,Mxy,Myx,Fxy,0)
  if (Fxy < Rxy)
  {x = y
   lsalto = 1
  }
  else
  { lsalto = 0
  }
  MCMC[i,12] = lsalto
}
mcmc = MCMC[(L+1):N,"x"]
```

Ahora para mostrar el histograma:

```
media=mean(mcmc)

hist(mcmc,
     freq = FALSE,
     main = "Distribucion de muestra MCMC",
     xlab = "x",
     ylab = "distribucion(x)",
     breaks = 200)
abline(v=mu, col='blue', lwd=3)      #Media de X1
abline(v=mu/2, col='red', lwd=3)     #Media de X2
abline(v=media, col='violet', lwd=3) #Media de Z
```

## Distribucion de muestra MCMC



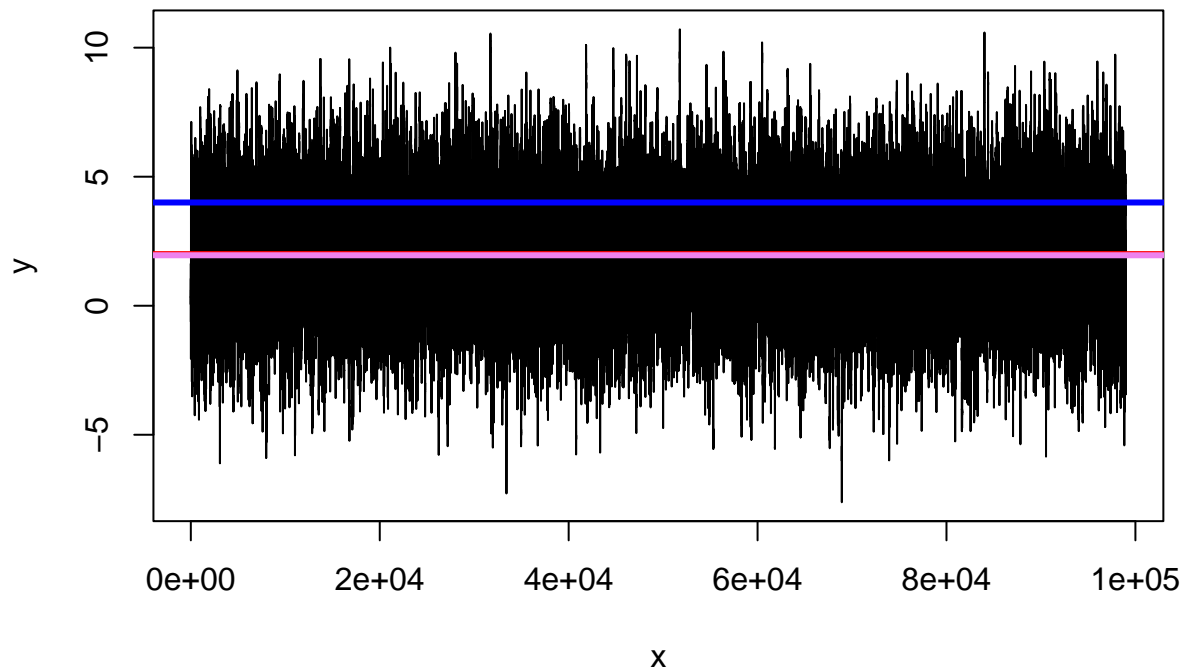
c. Estime la media de la distribución resultante de  $Z$ .

La media de mcmc es: 1.9555262

d. Gráfique el Traceplot de muestra MCMC del algoritmo junto con las medias de  $X_1, X_2, Z$ .

```
#Traceplot
plot(mcmc,type="l",xlab = "x", ylab = "y", main = "Traceplot de muestra MCMC")
abline(h=mu, col='blue', lwd=3)      #Media de X1
abline(h=mu/2, col='red', lwd=3)     #Media de X2
abline(h=media, col='violet', lwd=3) #Media de Z
```

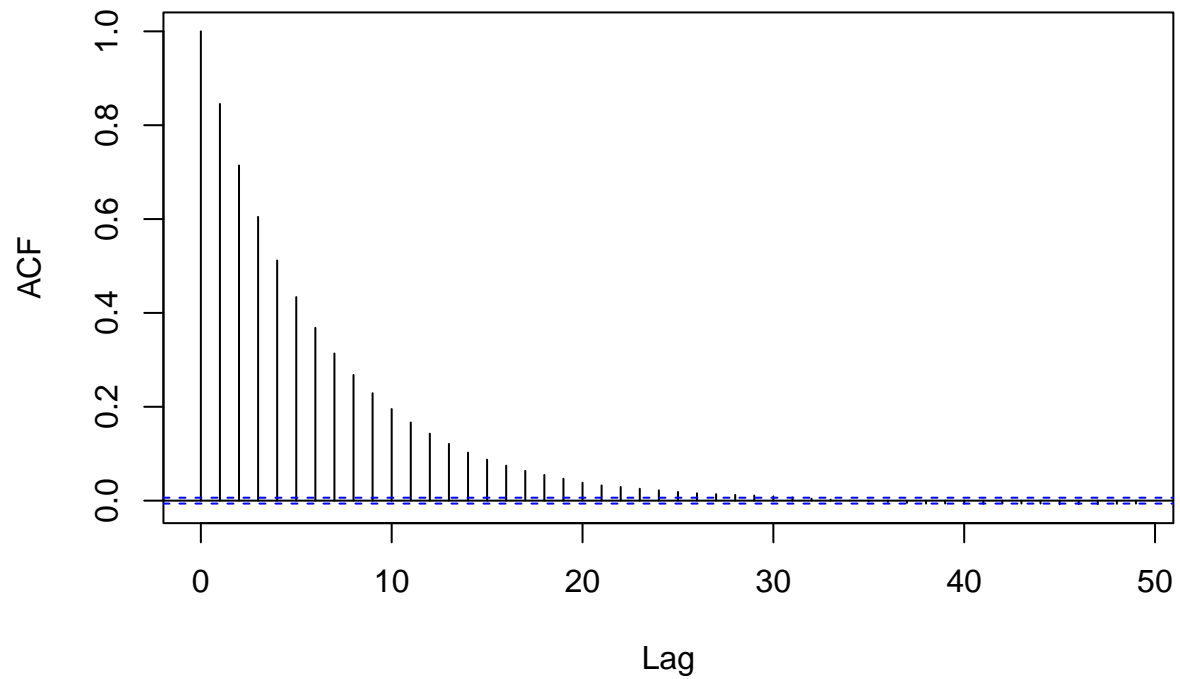
### Traceplot de muestra MCMC



e. El gráfico de Autocorrelación de la muestra MCMC del algoritmo.

```
#Grafico de Autocorrelación  
acf(mcmc,main = "Autocorrelacion de muestra MCMC")
```

## Autocorrelacion de muestra MCMC



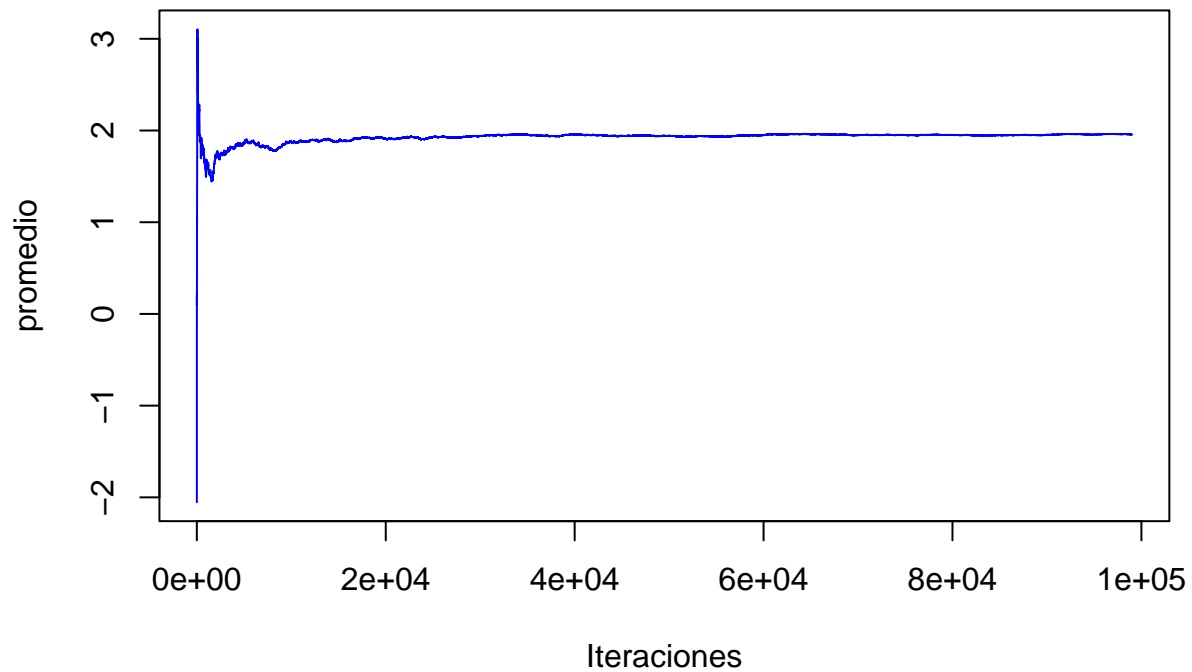
f. El gráfico de la convergencia de la media de la muestra MCMC del algoritmo.

```
#Grafico convergencia de la media
```

```
m=N-L
```

```
acumulado<-cumsum(mcmc)/(1:m)
```

```
plot(1:m,acumulado,col="blue",type="l",ylab="promedio",xlab="Iteraciones")
```



g. La tasa de aceptación del algoritmo.

```
cat("Tasa de aceptación \n",
    "NumeroSaltos/TotalIteraciones :", mean(MCMC[, "Salto"]) , "\n")
```

```
## Tasa de aceptación
## NumeroSaltos/TotalIteraciones : 0.70926
```