Tarea 2 Estadística Actuarial II

Maria Carolina Navarro Monge C05513 Tábata Picado Carmona C05961
Jose Pablo Trejos Conejo C07862

Primeramente, se cargan las librerías necesarias.

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.3.1
```

```
## Warning: package 'ggplot2' was built under R version 4.3.1
```

```
library(readxl)
```

```
## Warning: package 'readxl' was built under R version 4.3.1
```

Ejercicio 2

Usando la metodología de Muestreo por Importancia, Si X~N(0.5,0.5) estime:

a. P(X<-5)

Primeramenre, mediante el metódo de integración por Montecarlo se obtiene el siguiente resultado:

```
#--Estimación de función de distribución mediante integración por Montecarlo--

set.seed(2901)
n <- 10^4 #tamaño de la muestra

X <- rnorm(n, 0.5, sqrt(0.5))

f <- dnorm(X, 0.5, sqrt(0.5))

valor_estimado_1 <- mean(f)

valor_real <- pnorm(-5, 0.5, sqrt(0.5))

comparacion <- data.frame("Estimación" = valor_estimado_1, "Valor real" = valor_real)

print(comparacion)
```

```
##   Estimación   Valor.real
## 1  0.3991698 3.678924e-15
```

Como se puede observar, la estimación resultante converge lento al valor real. Por lo tanto, mediante Muestreo por Importancia se puede acelelar la convergencia empleando una densidad auxiliar. Para este caso, la densidad auxiliar a utilizar es una exponencial truncada de la forma $\lambda e^{-\lambda(x-t)}$, con x>t.

La probabilidad a estimar es equivalente a P(X>6). Nos basaremos en esta para aproximarla mediante el Muestro por Importancia por medio de una exponencial truncada en 6 con $\lambda = 1$. El procedimiento se muestra en el siguiente algoritmo:

```
#--Estimación de función de distribución mediante Muestreo por Importancia--

A <- rexp(n)+6 #datos aleatorios mayores a 6 con distribución exponencial

w <- dnorm(A, 0.5, sqrt(0.5)) / dexp(A-6)

valor_estimado <- mean(w)

resumen <- data.frame("Estimación" = valor_estimado, "Valor real" = valor_real)

print(resumen)
```

```
##     Estimación   Valor.real
## 1 3.669418e-15 3.678924e-15
```

De tal manera, se obtiene una mejor aproximación de la probabilidad P(X<-5), pues, es muy similar al valor real.

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

El error absoluto de la estimación es:

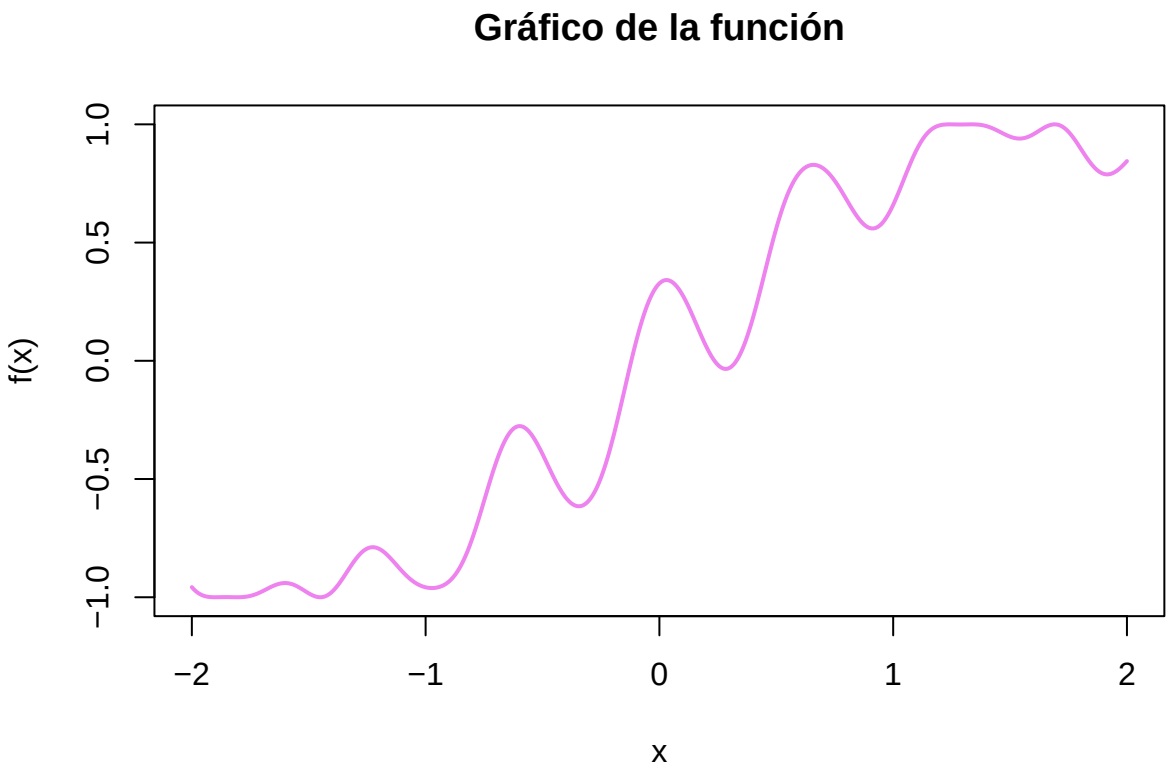
```
error_absoluto <- abs(valor_estimado-valor_real)
```

Ejercicio 4

Sea $f(x) = sen\left(x + \frac{cos(10x)}{3}\right)$ **para** $x \in [-2, 2]$ ## a.Utilizando el algoritmo de recalentamiento simulado estime el mínimo global en [−2,2], con valor inicial en 1.5.**

Primero definimos la función en R y la graficamos para darnos una idea de dónde puede estar ubicado el mínimo global de la función en el intervalo.

```
fx <- function(x){sin(x + (cos(10*x)/3))}
curve(fx,col="violet",lwd=2,from=-2,to = 2,n=1000,ylab="f(x)")
title("Gráfico de la función")
```



Ahora definimos la función que va a ejecutar el algoritmo de recalentamiento simulado la cual va a recibir la función definida anteriormente, el alpha que en este caso se establece en 0.1, el valor inicial en 1.5, número de iteraciones que para este ejercicio es de 1000 y por último, el mínimo y máximo del intervalo observado -2 y 2 respectivamente.

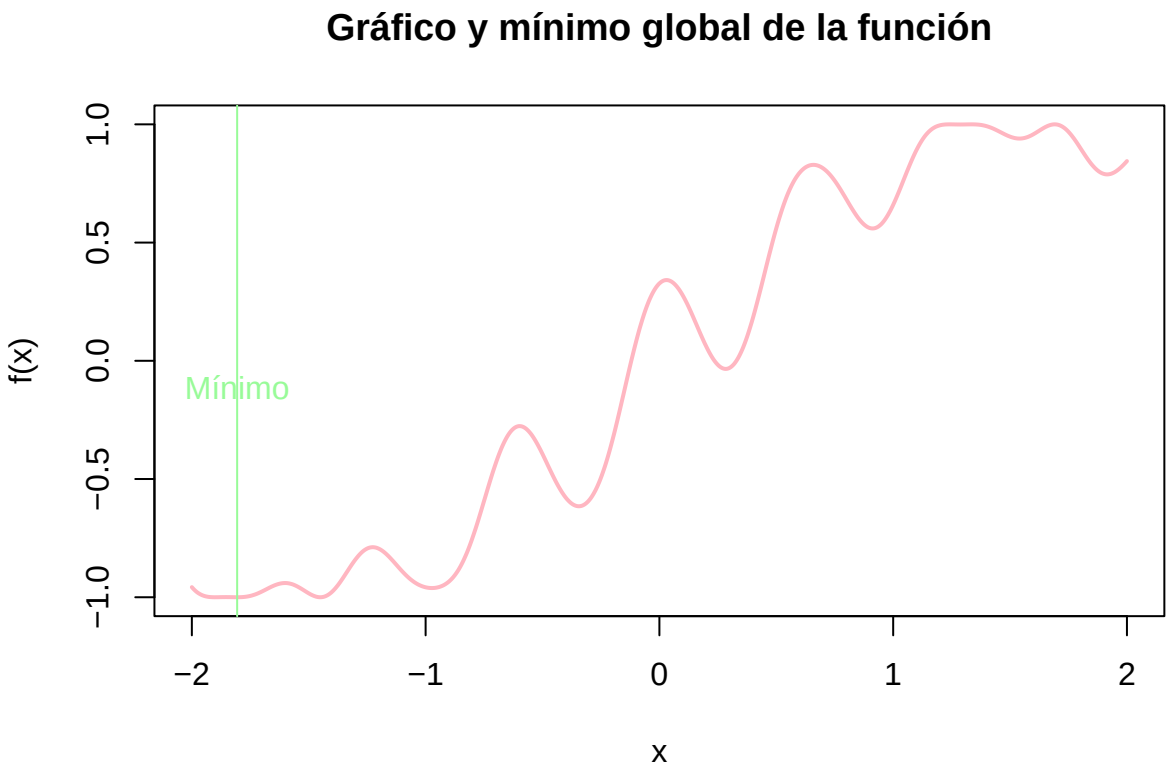
```
recalentamiento_simulado <- function(f,alpha=0.5,s0=0,niter,mini=-Inf,maxi=Inf){
  s_n <- s0
  estados <- rep(0,niter)
  iter_count <- 0

  for(k in 1:niter){
    estados[k]<-s_n
    T <- (1-alpha)^k
    s_new <- rnorm(1,s_n,1)

    if(s_new<mini){
      s_new <- mini
    }

    if(s_new>maxi){
      s_new <- maxi
    }

    dif <- f(s_new)-f(s_n)

    if(dif< 0){
```

```
      s_n <- s_new
    } else {
        random <- runif(1,0,1)

        if(random <exp(-dif/T)){
          s_n <- s_new
        }

    }

    iter_count <- iter_count +1

  }

  return(list(r=s_n,e=estados))
}

Resultado <- recalentamiento_simulado(fx,0.1,1.5,1000,-2,2)
Resultado$r
```

```
## [1] -1.806134
```

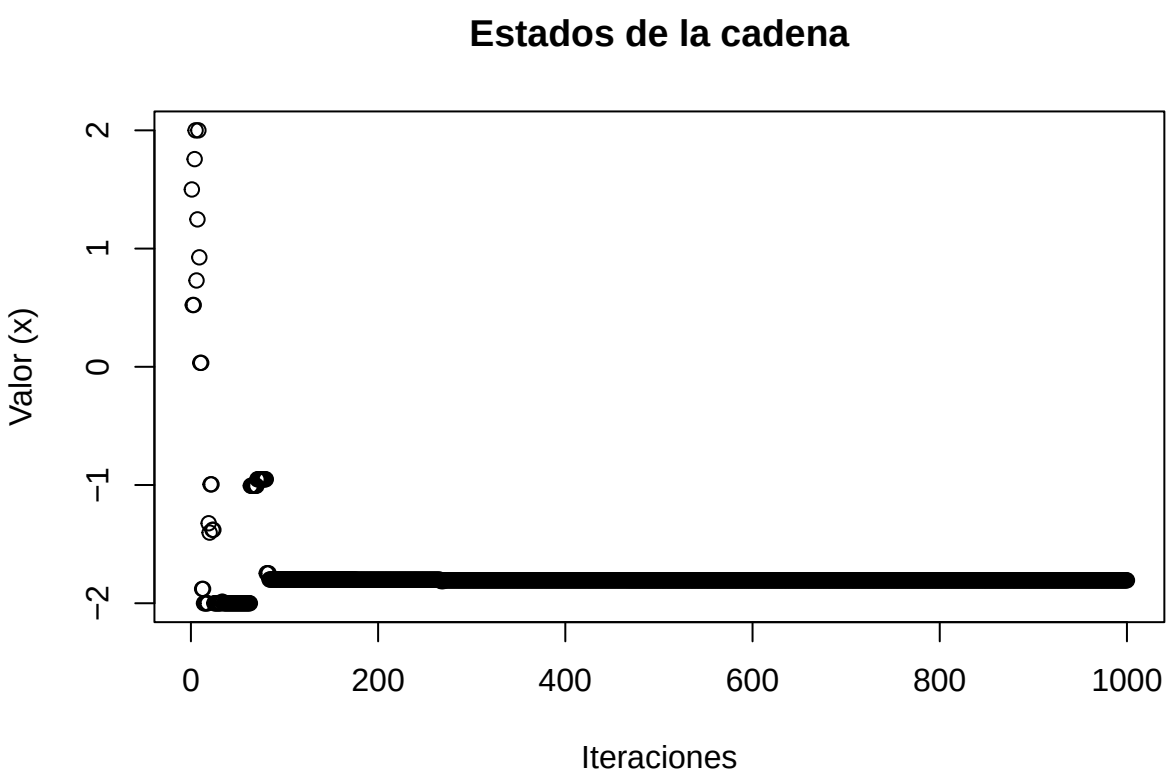
Una vez ejecutado el código arroja que el mínimo global de la función es -1.805683, para corrobar este resultado se decide graficar de nuevo la función junto con el mínimo.

```
curve(fx,col="#FFB6C1",lwd=2,from=-2,to = 2,n=1000,ylab="f(x)")
title("Gráfico y mínimo global de la función")
abline(v=Resultado$r,col="#98FB98")
text(Resultado$r, 0, "Mínimo", pos = 1, col = "#98FB98")
```



b.Grafique el resultado de los estados donde estuvo la cadena de la estimación del punto a.**

```
plot(Resultado$e, xlab = "Iteraciones", ylab = "Valor (x)",
     main = "Estados de la cadena")
```



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. Considerando:

a. Las tablas de vida dinámicas de la SUPEN (https://webapps.supen.fi.cr/tablasVida/Recursos/documentos/tavid2000-2150.xls)

b. Un cliente de 30 años, hombre con una suma asegurable de 1 000 000 colones.

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.

Primeramente, se carga la tabla de vida dinámica de la SUPEN y se filtra para obtener los datos correspondientes para un hombre que en el presente año (2024) tiene 30 años

```
#Se carga la base de datos
tabla_vida <- read_excel("tavid2000-2150.xls",
                         col_types = "numeric")

#Se filtra la base de datos para obtener los datos de un hombre nacido en 1994
#con edades mayor o igual a 30

datos <- subset(tabla_vida, sex == 1 & ynac == 1994 & edad >=30, select = c(edad,qx, year))
```

Además, se calculan las probabilidades de sobrevivencia necesarias para procesos posteriores

```
#Se obtienen las probabilidades de sobrevivencia

px <- 1- datos$qx

#Se añaden las probabilidades de sobrevivencia a la base datos
datos$px <- px
```

La construcción de la distribución de los pagos por año mediante MCMC se muestran en el siguiente código:

```
suma_asegurada_1 <- 10^6
suma_asegurada_2 <- 2*10^6

#--------------------MCMC---------------------------------|

#Se simulan diversas trayectorias de vida de la persona
set.seed(2901)
iteraciones=10^4
n=length(px)
pago <- rep(0, 86)

for (i in 1:iteraciones) {
  U <- runif(n)  # Se toman como probabilidades de muerte
  t <- 1
  cont <- 1

  #Determinación del año de fallecimiento
  while (t == 1) {
    if (U[cont] < px[cont]) {
      cont <- cont + 1
    } else {
      t <- 0
    }
  }
  año_fallecimiento <- cont - 1

  #Asignar los pagos correspondientes al año de fallecimiento
  if (año_fallecimiento < 30) {
    pago[año_fallecimiento + 1] <- pago[año_fallecimiento + 1] + suma_asegurada_2
  } else if (año_fallecimiento ==30) {
    pago[31] <- pago[31]+ suma_asegurada_1
  }else {
    pago[31] <- pago[31] + suma_asegurada_1
    pago[año_fallecimiento + 1] <- pago[año_fallecimiento + 1] + suma_asegurada_1
  }
}

resultado <- data.frame("Años pago"= datos$year, "Pago" = pago)
```

El histograma de los pagos esperados por año es el siguiente:

```r
ggplot(data = resultado, aes(x = Años.pago, y = Pago)) +
  geom_bar(stat = "identity", fill = "blue") +
  labs(title = "Histograma de Pagos Esperados por año", x = "Años", y = "Frecuencia")+
  theme_minimal()
```

Es evidente que la mayor cantidad de pagos se sitúan a los 60 años y posteriomente después de los 60, lo que indica que es más probable que el cliente fallezca después de los 60 años.

Se puede verificar el resultado obtenido por MCMC si lo comparamos con un histograma obtenido mediante un método determinista como el que se muestra a continuación:

```r
#----- Método determinista-----------------------------------------------------/

qx<- datos$qx

#Se crea una función que obtiene n_p_30 (probabilidad de sobrevivencia acumulada)
n_p_30 <- c(0)

n_p_30_function <- function(px) {

  for (i in 1:length(px)) {
    resultado <-1
    for(j in 1: i){
      resultado <- resultado*px[j]
    }
```

```r
    n_p_30[i] <- resultado
  }

  return(n_p_30)
}

n_p_30 <- n_p_30_function(px)

datos$n_p_30 <- n_p_30


#se calculan los pagos esperados para cada año
pago_esperado <- c(0)
pago_esperado[1] <- suma_asegurada_2*qx[1]

#caso fallecimiento antes de los 60 años
for (i in 2: 29 ) {
    pago_esperado[i] <- suma_asegurada_2*n_p_30[i-1]*qx[i]
}

#caso sobrevive a los 60 años

pago_esperado[30] <-suma_asegurada_1*n_p_30[30]

#caso fallecimiento después de los 60 años

for (i in 1: (length(px)-30)) {
  pago_esperado[30+i] <- suma_asegurada_1*n_p_30[30+i-1]*qx[30+i]
}

resultado_determinista <- data.frame("Años pago"= datos$year, "Pago" = pago_esperado)

ggplot(data = resultado_determinista, aes(x = Años.pago, y = Pago)) +
  geom_bar(stat = "identity", fill = "blue") +
  labs(title = "Histograma de Pagos Esperados por año", x = "Años", y = "Frecuencia")+
  theme_minimal()
```

Como se puede observar, los pagos esperados mediante el método determinista y el MCMC muestran distribuciones muy similares. Por tanto, se verifica que el resultado que se obtuvo por el método MCMC es aceptable.

Ejercicio 6

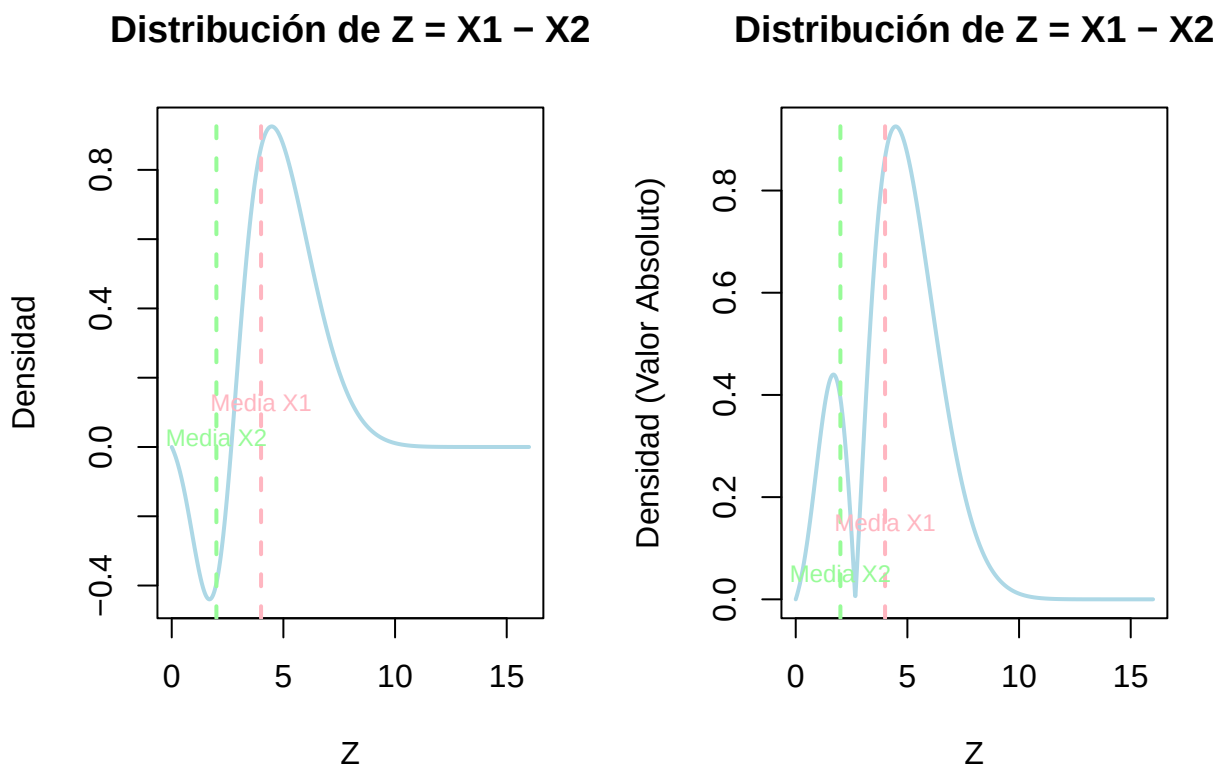
Usando el Algoritmo de Metropolis-Hastings construya una muestra de $Z = X1 - X2$ **donde** $X1$ $N(\mu, \sigma_2)$ **y** $X2$ $N(\mu/2, \sigma_2/4)$**, considere para este ejercicio = 2=4.** ## a.Gráfique la distribución de junto con las medias de 1, 2.

Primero se realiza la función de Z. Para ello se hace la resta de los núcleos de las distribuciones de X1 y X2.

```
fnormal <- function(x,mu1,mu2,sigma1, sigma2) {
  fx <- exp(-((x-mu1)^2/(2*(sigma1)))) - exp(-((x-mu2)^2/(2*(sigma2))))
  return(fx)
}

mu1 <- 4
mu2 <- 2
sigma1 <- 4
sigma2 <- 1

fZ <- function(x){return(fnormal(x,mu1,mu2,sigma1,sigma2))}
```

Ahora se realiza el gráfico de la distribución de Z.

```r
# Valores para el rango de la gráfica
x_values <- seq(0, 16, length.out = 1000)

par(mfrow = c(1, 2))

# Gráfico de la distribución de Z y las medias de X1 y X2
plot(x_values, fZ(x_values), type = "l", col = "#ADD8E6", lwd = 2,
     xlab = "Z", ylab = "Densidad", main = "Distribución de Z = X1 - X2")

# Líneas verticales para las medias de X1 y X2
abline(v = c(mu1, mu2), col = c("#FFB6C1", "#98FB98"), lty = c(2, 2), lwd = 2)

# Etiquetas para las medias
text(mu1, 0.20, "Media X1", pos = 1, col = "#FFB6C1", cex = 0.75)
text(mu2, 0.10, "Media X2", pos = 1, col = "#98FB98", cex = 0.75)

# Gráfico de la distribución en valor absoluto de Z y las medias de X1 y X2
plot(x_values, abs(fZ(x_values)), type = "l", col = "#ADD8E6", lwd = 2, xlab = "Z",
     ylab = "Densidad (Valor Absoluto)", main = "Distribución de Z = X1 - X2")

abline(v = c(mu1, mu2), col = c("#FFB6C1", "#98FB98"), lty = c(2, 2), lwd = 2)

text(mu1, 0.20, "Media X1", pos = 1, col = "#FFB6C1", cex = 0.75)
text(mu2, 0.10, "Media X2", pos = 1, col = "#98FB98", cex = 0.75)
```



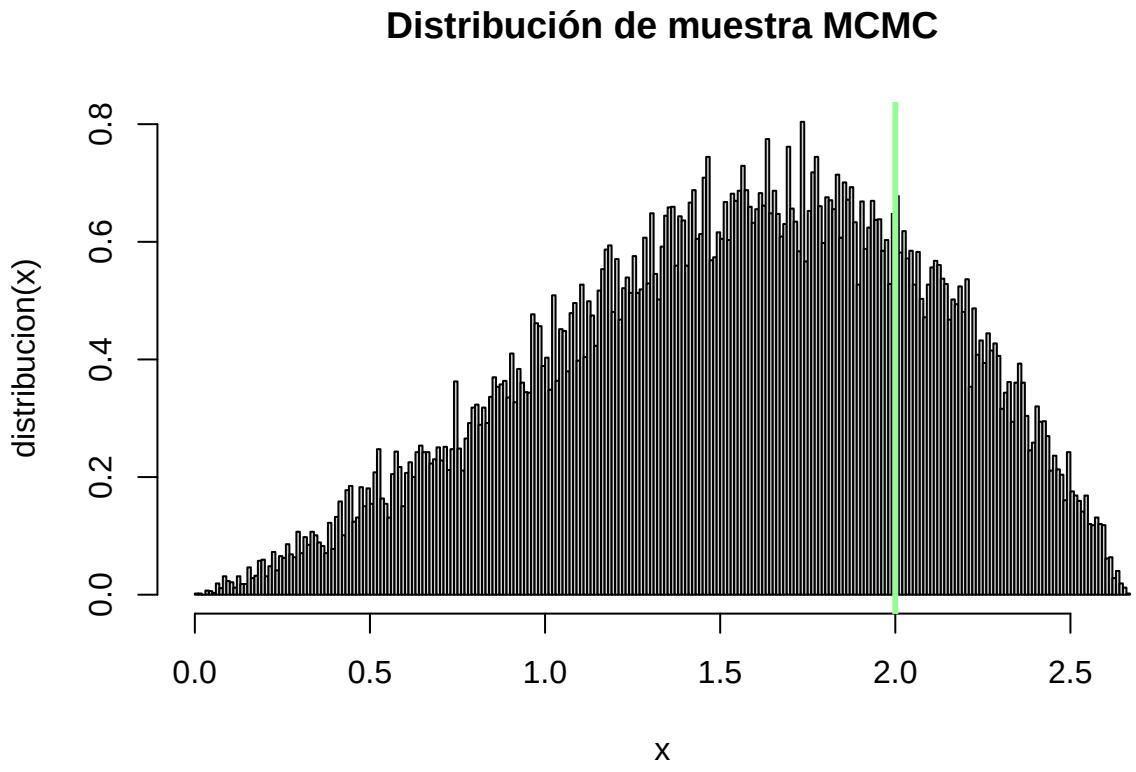
b.Gráfique la distribución (histograma) de la muestra MCMC del algoritmo junto con las medias de 1, 2.**

Antes de graficar el histograma se debe crear y ejecutar la función encargada del algoritmo MCMC.

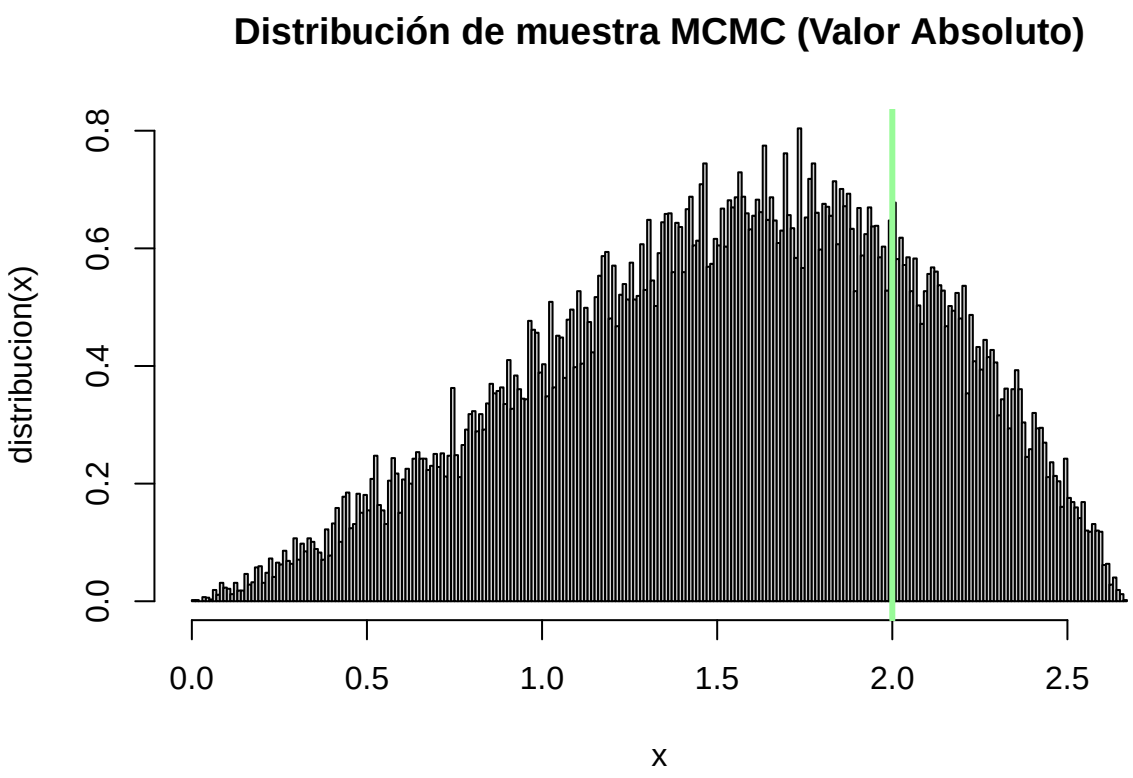
```
fpK <- function(x,y){
  pK <- dcauchy(y,location = x) #x es elcentro del pico de la distribución.
  return(pK)
}

N <- 10^5 #Número de Iteraciones
L <- 1000 #periodo quemado (burnin)
MCMC <- matrix(data=0,nrow=N,ncol=12)
colnames(MCMC) <- c("x","y","PIx","PIy","Kxy","Kyx","Rxy","Ryx","Mxy","Myx","Fxy",
                    "Salto")
#1.Iniciar 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 <- fZ(x)
  PIy <- fZ(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

}
```

```r
mcmc <- MCMC[(L+1):N,"x"]
```

```r
hist(mcmc, freq=FALSE, main="Distribución de muestra MCMC", xlab="x",
     ylab="distribucion(x)", breaks=200)
abline(v=mu1,col="#FFB6C1",lwd=3)
abline(v=mu2,col="#98FB98",lwd=3)
```




```r
hist(abs(mcmc), freq = FALSE,
     main = "Distribución de muestra MCMC (Valor Absoluto)",
     xlab = "x", ylab = "distribucion(x)", breaks = 200)
abline(v=mu1,col="#FFB6C1",lwd=3)
abline(v=mu2,col="#98FB98",lwd=3)
```



c.Estime la media de la distribución resultante de .**

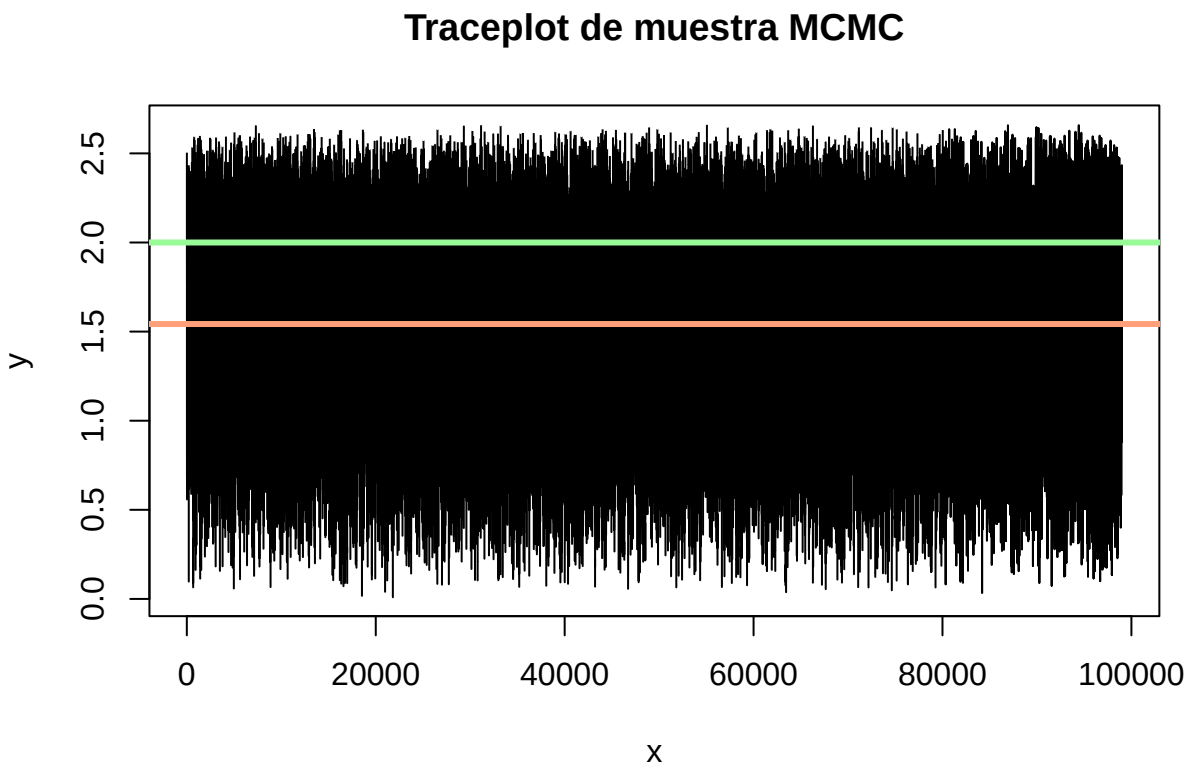
La media resultante de Z es de 5.064882

```
media <- mean(mcmc)
media
```

```
## [1] 1.542475
```

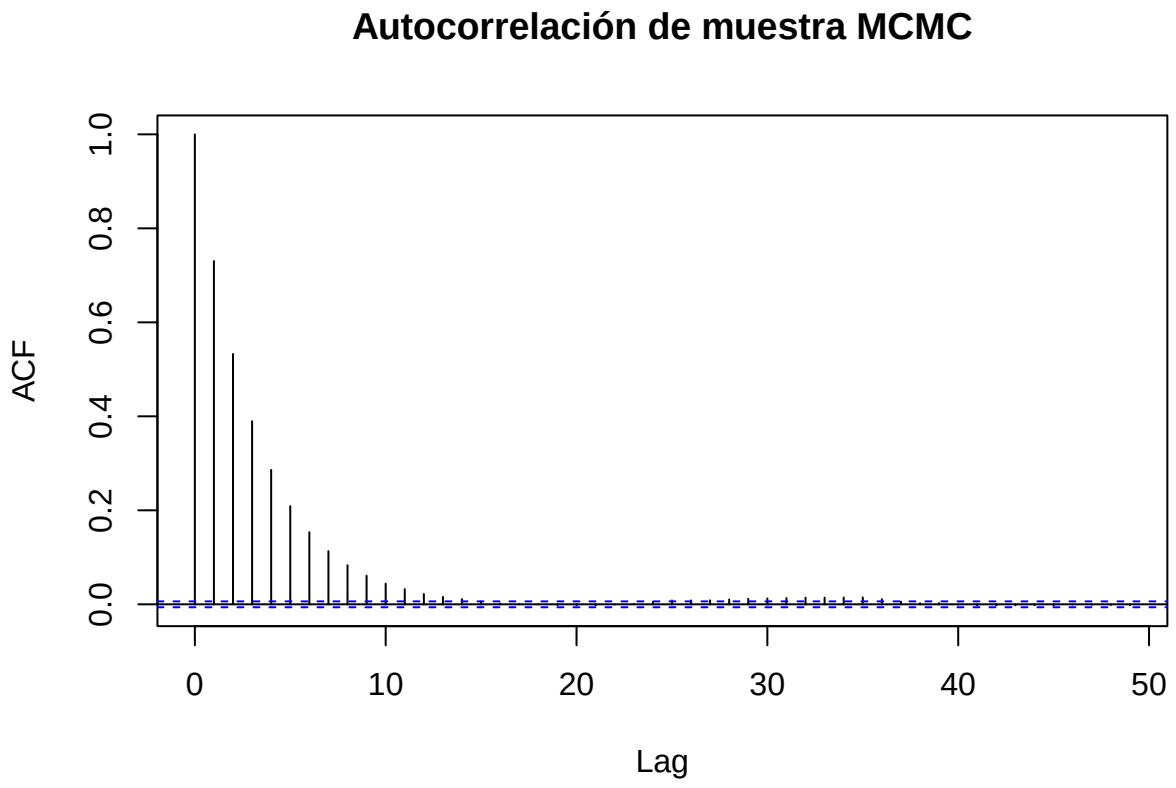
d.Gráfique el Traceplot de muestra MCMC del algoritmo junto con las medias de 1, 2, .**

```
options(scipen = 999)
par(mfrow = c(1, 1))

plot(mcmc,type="l",xlab="x",ylab ="y",main="Traceplot de muestra MCMC")
abline(h=mu1,col="#FFB6C1",lwd=3)
abline(h=mu2,col="#98FB98",lwd=3)
abline(h=media,col="#FFA07A",lwd=3)
```



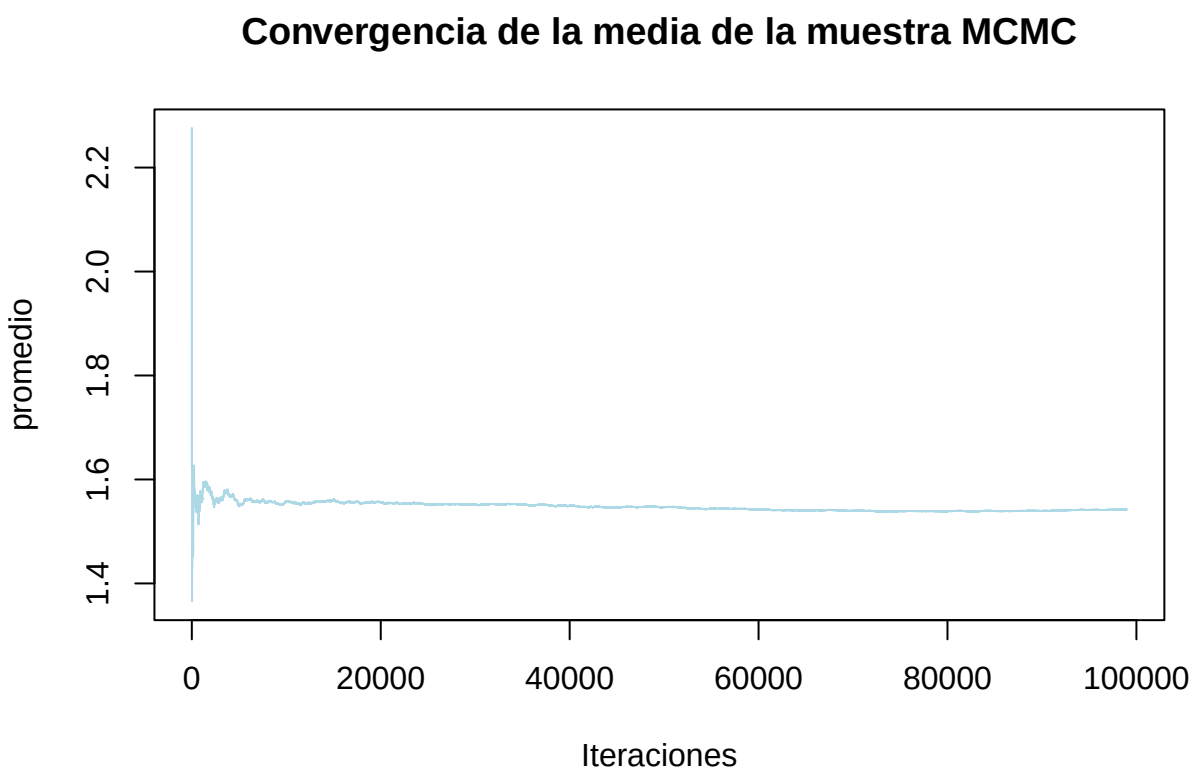
e.El gráfico de Autocorrelación de la muestra MCMC del algoritmo.**

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



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

```
m <- N-L
acumulado <- cumsum(mcmc)/(1:m)
plot(1:m,acumulado,col="#ADD8E6",type="l",ylab="promedio",xlab="Iteraciones",
     main="Convergencia de la media de la muestra MCMC")
```



g.La tasa de aceptación del algoritmo.**

La tasa de aceptación es de un 60.097%

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

```
## Tasa de aceptación: 0.40634
```

También este ejercicio se puede replantear como que Z al ser uan resta de normales también es una normal con media $\mu_1 - \mu_2$ y varianza $\sigma_1 + \sigma_2$. Por lo tanto, se puede ejcutar el código anterior pero definiendo la función a muestrar como una normal con los parámetros indicados.

```
fnormal <- function(x,mu1,mu2,sigma1, sigma2) {
  fx <- dnorm(x, mean = mu1-mu2, sd = sigma1+sigma2)
  return(fx)
}

mu1 <- 4
mu2 <- 2
sigma1 <- 4
sigma2 <- 1

fZ <- function(x){return(fnormal(x,mu1,mu2,sigma1,sigma2))}
```