

## 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)
library(readxl)
```

### Ejercicio 2

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

#### a. $P(X < -5)$

Inicialmente se analiza mediante el método de integración por Montecarlo, con el cual 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 acelerar 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

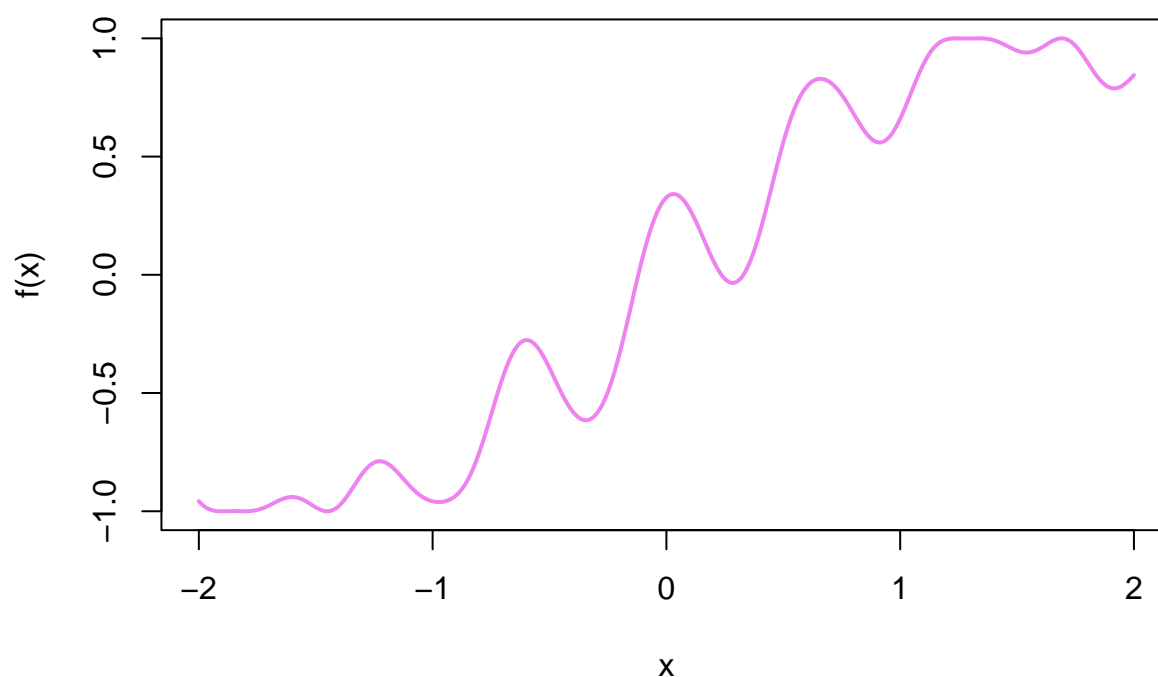
Sea  $f(x) = \text{sen}\left(x + \frac{\cos(10x)}{3}\right)$  para  $x \in [-2, 2]$

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")
```

## 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
```

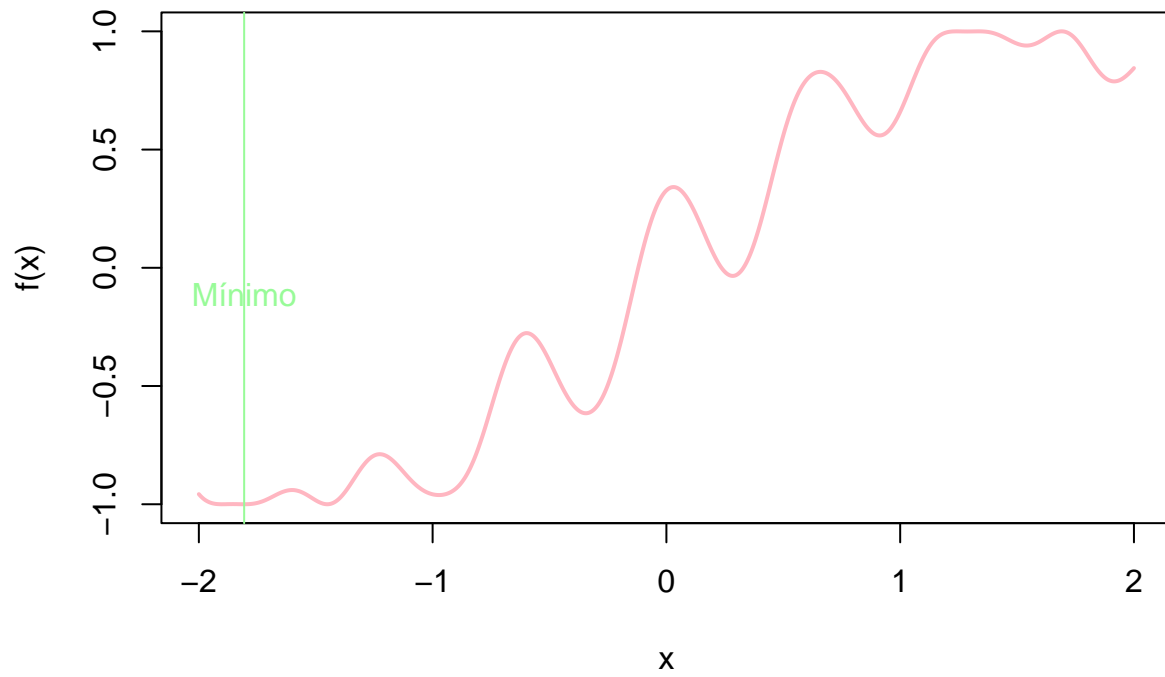
Una vez ejecutado el código arroja que el mínimo global de la función es -1.805683, para corroborar este resultado se decide graficar de nuevo la función junto con el mínimo.

```

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")

```

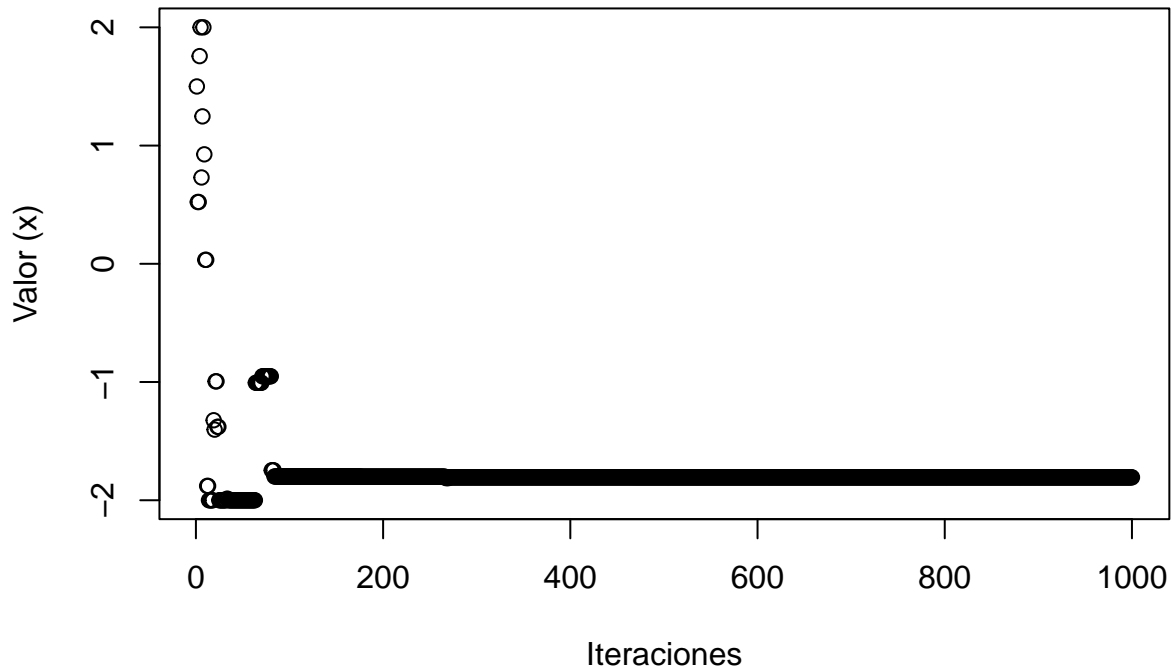
## Gráfico y mínimo global de la función



##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")
```

## 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:

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

```
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 posteriormente 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:

```
#----- 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]
    }
  }
}
```



```

    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.